Bijlage D: De Levenscyclus en Beveiliging van Plugins binnen het S1mpleTrader V2 Ecosysteem

Versie 3.0 · Status: Definitief Architectuurvoorstel

Inleiding: Een Modulair Framework voor Gelaagde Beveiliging

Dit document presenteert een diepgaand architectuurvoorstel voor de beveiliging van de plugin-levenscyclus binnen het S1mpleTrader V2 ecosysteem. De context van een financieel handelsplatform stelt unieke en niet-onderhandelbare eisen aan de beveiliging, integriteit en prestaties van softwarecomponenten van derden. Plugins, die per definitie (indirect) toegang hebben tot kapitaal, marktdata en strategische algoritmes, introduceren een aanzienlijk risico dat met een robuust, gelaagd en verifieerbaar beveiligingsmodel beheerst moet worden.

De voorgestelde architectuur is gebaseerd op een modulaire, tweeledige strategie die is ontworpen om zowel onmiddellijke, pragmatische beveiligingsgaranties te bieden als een pad te effenen naar een toekomst van cryptografisch afdwingbare en gedecentraliseerde betrouwbaarheid. Deze twee sporen zijn complementair en ontworpen voor een stapsgewijze implementatie:

1. **Zero Trust Architectuur (Praktisch en Onmiddellijk Toepasbaar):** Dit spoor vormt de fundering van ons beveiligingsmodel. Gebaseerd op de principes van NIST SP 800-207, gaan we uit van het adagium "vertrouw nooit, verifieer altijd".[1] Elke plugin, elke afhankelijkheid, en elke runtime-actie wordt als potentieel onbetrouwbaar beschouwd en moet expliciet worden geverifieerd. Dit wordt gerealiseerd door een combinatie van strikt afhankelijkheidsbeheer, cryptografische ondertekening van code, proactieve statische analyse, en rigoureuze runtime-isolatie (sandboxing). Deze maatregelen zijn direct implementeerbaar en bieden een sterke basisverdediging tegen de meest voorkomende en impactvolle dreigingen.
2. **Blockchain & Moderne Cryptografie (Toekomstbestendig en Verifieerbaar):** Dit spoor bouwt voort op de Zero Trust-basis door vertrouwensmechanismen te verankeren in cryptografische waarheden in plaats van in centrale autoriteiten. Door gebruik te maken van technologieën zoals gedecentraliseerde identiteiten (DIDs), verifieerbare claims (Verifiable Credentials), onveranderlijke grootboeken (blockchains) en zelf-uitvoerende contracten (smart contracts), creëren we een ecosysteem waarin de integriteit en herkomst van software-artefacten onafhankelijk en wiskundig verifieerbaar

zijn. Dit spoor introduceert tevens economische prikkels en boetes (slashing) om goed gedrag te stimuleren en kwaadwillende actoren financieel te bestraffen.

De modulaire opzet stelt S1mpleTrader V2 in staat om te starten met een robuust Zero Trust-model dat de primaire risico's afdekt, en naarmate het ecosysteem groeit en de behoefte aan hogere garanties toeneemt, stapsgewijs de cryptografische en gedecentraliseerde bouwstenen te integreren. Deze aanpak maximaliseert de veiligheid op korte termijn zonder de ontwikkelingssnelheid onnodig te belemmeren, terwijl het een strategisch pad biedt naar een toekomst van aantoonbare en gedecentraliseerde betrouwbaarheid.

D.0. Dreigingsmodel en Risicoanalyse in de Financiële Context

Een effectief beveiligingsmodel begint met een grondige en realistische analyse van de dreigingen. In de context van S1mpleTrader, waar plugins direct of indirect interageren met financieel kapitaal, zijn de risico's niet abstract, maar concreet en potentieel catastrofaal. De aanname is dat elke plugin een potentiële vector is voor financieel verlies, datamanipulatie of diefstal van intellectueel eigendom (handelsstrategieën).

Uitgebreide Analyse van Risico's

De belangrijkste risico's worden hieronder gedetailleerd, ondersteund door reële voorbeelden die de ernst en waarschijnlijkheid van deze dreigingen onderstrepen.

Supply-Chain Compromise

Een supply-chain-aanval richt zich op de minder beveiligde elementen in de toeleveringsketen om een beter beveiligd hoofddoel te compromitteren.[2] Voor software betekent dit dat niet de applicatie zelf, maar een van haar afhankelijkheden (dependencies), bouwtools of distributiekanalen wordt aangevallen. De financiële sector is een primair doelwit voor dergelijke aanvallen, die vaak een cascade-effect hebben.[3] De wereldwijde jaarlijkse kosten

van deze aanvallen worden geschat op $60 miljard in 2025, oplopend tot $138 miljard in 2031.[4]

- **Case Study: De Target-hack (2013):** Een van de meest illustratieve voorbeelden is de datalek bij de Amerikaanse retailer Target. Aanvallers kregen toegang tot het netwerk van Target niet door Target direct aan te vallen, maar via een gecompromitteerde derde partij: een leverancier van HVAC-systemen. Via deze ogenschijnlijk onbelangrijke toeleveringsschakel werd malware geïntroduceerd in de kassasystemen, wat leidde tot de diefstal van de gegevens van 40 miljoen creditcards. De totale kosten voor Target bedroegen meer dan $61 miljoen, exclusief de enorme reputatieschade.[2] Dit toont aan dat elke externe afhankelijkheid, hoe triviaal ook, een potentiële aanvalsvector is.
- **Case Study: SolarWinds (2020):** Deze geavanceerde aanval toont het risico van een "midstream" compromittering. Aanvallers wisten het bouwproces van SolarWinds' Orion-software te infiltreren en voegden een backdoor toe aan een legitieme software-update. Deze kwaadaardige update werd vervolgens gedistribueerd naar ongeveer 18.000 klanten, waaronder Amerikaanse overheidsinstanties en grote bedrijven, wat leidde tot een van de meest significante spionage-incidenten van de afgelopen decennia.[4] Dit benadrukt de noodzaak om niet alleen de code zelf, maar het gehele bouw- en distributieproces te beveiligen.
- **Specifieke Risico's in het Python Ecosysteem: Typosquatting:** Deze dreiging is bijzonder relevant voor S1mpleTrader, aangezien plugins in Python worden ontwikkeld. Typosquatting is een techniek waarbij aanvallers kwaadaardige pakketten publiceren op de Python Package Index (PyPI) met namen die sterk lijken op populaire, legitieme pakketten (bijv. reqjuests in plaats van requests of python-binance in plaats van python-binance).[7] Recente, geautomatiseerde campagnes hebben specifiek populaire bibliotheken in de financiële en cryptocurrency-handelssector als doelwit gehad. Bibliotheken zoals
  ccxt, freqtrade, yfinance en bitcoinlib werden het slachtoffer van honderden kwaadaardige variaties die binnen enkele uren werden gepubliceerd.[8] De payload van deze pakketten is vaak ontworpen om omgevingsvariabelen, authenticatiegegevens en de inhoud van cryptocurrency-wallets te stelen.[7] Dit toont aan dat het S1mpleTrader-ecosysteem een direct en aantrekkelijk doelwit is voor dit type geautomatiseerde aanval.

Malware en Logic Bombs

Een logic bomb is een stuk kwaadaardige code dat opzettelijk in een softwaresysteem wordt ingevoegd en een schadelijke functie activeert wanneer aan specifieke voorwaarden is voldaan (bijv. een bepaalde datum, of het verwijderen van een gebruikersaccount).[11] Deze dreiging komt vaak van binnenuit (een ontevreden ontwikkelaar of medewerker) maar kan ook

door een externe aanvaller worden geplant.

- **Case Study: UBS (2006):** Roger Duronio, een systeembeheerder bij UBS, plantte een logic bomb die was ontworpen om duizenden servers te beschadigen. Zijn motief was financieel: hij had shortposities op het aandeel UBS en hoopte met de aanval de koers te laten kelderen. De bom werd geactiveerd, veroorzaakte aanzienlijke schade en Duronio werd veroordeeld tot een gevangenisstraf en een schadevergoeding van $3,1 miljoen.[11] Dit illustreert dat de motivatie voor een dergelijke aanval in een financiële context direct gekoppeld kan zijn aan marktmanipulatie.
- **Case Study: Fannie Mae (2008):** Een IT-contractant, Rajendrasinh Makwana, plantte na zijn ontslag een logic bomb die was ontworpen om alle 4.000 servers van de hypotheekgigant te wissen. De bom werd ontdekt en onschadelijk gemaakt voordat de triggerdatum (31 januari 2009) werd bereikt. Dit voorbeeld benadrukt het risico van insiders met verhoogde privileges, zelfs nadat hun dienstverband is beëindigd.[11]

Privileged Action Abuse

Dit risico omvat het misbruik van legitieme, toegekende permissies voor ongeautoriseerde doeleinden.[14] Dit kan opzettelijk gebeuren door een kwaadwillende ontwikkelaar of onopzettelijk wanneer een legitieme plugin wordt gecompromitteerd. In de context van S1mpleTrader kan dit zich manifesteren als:

- **Ongeautoriseerde Handel:** Een plugin die is ontworpen voor het analyseren van marktstructuren, begint plotseling met het plaatsen van grote, risicovolle orders.
- **Data-exfiltratie:** Een datavisualisatieplugin lekt gevoelige informatie, zoals de parameters van een propriëtaire handelsstrategie, accountgegevens of API-sleutels, naar een externe server.[15]
- **Sabotage:** Een plugin manipuleert opzettelijk de inkomende marktdata, corrumpeert de staat van andere plugins, of wist kritieke logbestanden om sporen uit te wissen.[15]

Integriteit, Onweerlegbaarheid en Compliance

In de financiële wereld is een onveranderlijk en verifieerbaar auditspoor van cruciaal belang. Het is niet voldoende om te weten dat een order is geplaatst; het moet onweerlegbaar zijn *wie* (welke gebruiker of welke plugin) de order heeft geplaatst, *wanneer* dit gebeurde, en op basis van *welke* data en code. Zonder deze garanties zijn forensische analyses na een incident, het oplossen van geschillen, en het voldoen aan regelgevende vereisten (compliance) onmogelijk.

Architecturale Doelstellingen en Principes

Op basis van het bovenstaande dreigingsmodel worden de volgende kernprincipes en doelstellingen voor de architectuur vastgesteld:

- **Minimale Privileges (Least Privilege):** Elke plugin moet standaard geen enkele permissie hebben. Permissies (netwerktoegang, bestandssysteemtoegang, orderuitvoering) moeten expliciet worden aangevraagd in een manifest en worden afgedwongen door het platform.
- **Defense-in-Depth:** Er wordt uitgegaan van de aanname dat elke afzonderlijke beveiligingslaag kan falen. Daarom worden meerdere, overlappende controles geïmplementeerd. Een kwaadaardige plugin moet achtereenvolgens de code-ondertekening, de statische analyse, de policy-engine, de runtime-sandbox en de gedragsmonitoring omzeilen om schade aan te richten.
- **Aantoonbare Integriteit (Verifiable Integrity):** Elk software-artefact—van de plugin-bundel zelf tot elke meegeleverde (vendored) afhankelijkheid—moet cryptografisch verifieerbaar zijn. De integriteit moet van de bron tot aan de uitvoering kunnen worden gevalideerd.
- **Forensische Herleidbaarheid (Forensic Traceability):** Alle significante gebeurtenissen in de levenscyclus van een plugin (installatie, update, uitvoering van een geprivilegieerde actie, runtime-fout) moeten worden gelogd in een fraudebestendig, centraal en auditeerbaar systeem.
- **Onafhankelijke Verificatie (Toekomst):** Het uiteindelijke doel is om de betrouwbaarheid van artefacten en gebeurtenissen te baseren op cryptografisch bewijs dat door elke partij onafhankelijk kan worden geverifieerd, zonder te hoeven vertrouwen op een centrale autoriteit.

D.1. Fundamentele Filosofieën: Zero Trust en Cryptografische Waarheid

De architectuur rust op twee complementaire filosofieën die samen een robuust en toekomstbestendig beveiligingsmodel vormen. De Zero Trust-aanpak biedt de pragmatische, direct implementeerbare basis, terwijl de visie op cryptografische waarheid een pad biedt naar een hoger niveau van aantoonbare veiligheid.

D.1.A Zero Trust Architectuur (ZTA) in de Praktijk

De kern van de onmiddellijke beveiligingsstrategie is de adoptie van een Zero Trust Architectuur (ZTA). Deze filosofie, formeel beschreven in de NIST Special Publication 800-207, breekt met het traditionele, op perimeters gebaseerde beveiligingsmodel. In plaats van aan te nemen dat alles binnen het netwerk "vertrouwd" is, stelt Zero Trust dat geen enkele entiteit, gebruiker of component impliciet vertrouwd mag worden, ongeacht zijn locatie of herkomst.[1] Elke toegangsverzoek moet expliciet en continu worden geverifieerd.

Voor het S1mpleTrader plugin-ecosysteem vertalen we de zeven fundamentele grondbeginselen van NIST SP 800-207 als volgt [1]:

1. **"Alle databronnen en computing services worden als resources beschouwd."**
    - **Toepassing:** Een plugin is niet één monolithische entiteit. Het is een verzameling van resources: de broncode, de gecompileerde bundel, de configuratie (plugin_manifest.yaml), de vendored dependencies, en de runtime-instantie. Elk van deze resources wordt afzonderlijk beheerd, beveiligd en geverifieerd.
2. **"Alle communicatie is beveiligd, ongeacht de netwerklocatie."**
    - **Toepassing:** Er is geen "vertrouwd intern netwerk". Alle communicatie tussen een plugin-proces en de S1mpleTrader-core (via Inter-Process Communication, IPC) moet worden gemedieerd en gecontroleerd. Elke uitgaande netwerkverbinding van een plugin naar een externe API (bijv. een exchange) moet expliciet worden toegestaan via een allowlist-policy en moet versleuteld zijn.
3. **"Toegang tot individuele resources wordt per sessie verleend."**
    - **Toepassing:** Een plugin die bij het opstarten toestemming krijgt om marktdata te lezen, behoudt deze toestemming niet voor onbepaalde tijd. De toegang wordt per sessie verleend en kan worden ingetrokken als de context verandert (bijv. als er verdacht gedrag wordt gedetecteerd). Authenticatie voor één resource (bijv. historische data) verleent niet automatisch toegang tot een andere resource (bijv. de order-API).
4. **"Toegang tot resources wordt bepaald door een dynamisch beleid."**
    - **Toepassing:** De beslissing om een plugin te laden of een actie toe te staan is geen statische ja/nee-beslissing. Het wordt dynamisch geëvalueerd door een policy engine (zie D.9.4). Dit beleid weegt meerdere factoren mee: de identiteit van de ondertekenaar van de plugin, de resultaten van de statische analyse, de aangevraagde permissies in het manifest, en de huidige risicostatus van het systeem.
5. **"De organisatie monitort en meet de integriteit en veiligheidspostuur van alle eigen en geassocieerde assets."**
    - **Toepassing:** Het platform zal continu de staat van alle geïnstalleerde plugins monitoren. Dit omvat het periodiek opnieuw scannen van de code op nieuw ontdekte kwetsbaarheden (CVE's) en het observeren van het runtime-gedrag om afwijkingen van de baseline te detecteren.

6. **"Alle resource-authenticatie en -autorisatie zijn dynamisch en strikt afgedwongen voordat toegang wordt verleend."**
    - **Toepassing:** Dit is het "always verify"-principe in de praktijk. Voordat een plugin-bestand wordt geladen, wordt de cryptografische handtekening geverifieerd (authenticatie). Voordat de plugin een order plaatst, wordt in real-time gecontroleerd of de plugin de exchange:place_order-permissie heeft en of de orderparameters binnen de door het beleid gestelde limieten vallen (autorisatie).
7. **"De organisatie verzamelt zo veel mogelijk informatie... om haar veiligheidspostuur te verbeteren."**
    - **Toepassing:** Alle beveiligingsgerelateerde gebeurtenissen—succesvolle en mislukte handtekeningverificaties, bevindingen van statische analyses, runtime policy violations, resourcegebruik—worden centraal gelogd. Deze data wordt gebruikt voor incident response, forensische analyse en, cruciaal, voor het continu verfijnen en aanscherpen van het beveiligingsbeleid.

D.1.B Blockchain & Cryptografie als Toekomstige Trust-Ankers

Waar Zero Trust zich richt op het *beheren* van risico in een inherent onbetrouwbare omgeving, richt het cryptografische spoor zich op het creëren van componenten met *aantoonbare* betrouwbaarheid, waardoor het risico zelf wordt verminderd. Deze aanpak verplaatst het vertrouwensanker van een centrale autoriteit (bijv. de beheerder van de S1mpleTrader Plugin Hub) naar onveranderlijke, wiskundig verifieerbare feiten op een gedistribueerd grootboek.

- **Decentralized Identifiers (DIDs):** De basis van dit spoor is de adoptie van de W3C-standaard voor Decentralized Identifiers.[17] Een DID is een wereldwijd unieke identifier (bijv.
  did:example:12345) die een entiteit (een ontwikkelaar, een organisatie, of zelfs een software-artefact) vertegenwoordigt. In tegenstelling tot een e-mailadres of een GitHub-gebruikersnaam, wordt een DID direct door de entiteit zelf gecreëerd en beheerd met behulp van cryptografische sleutels. Het is niet afhankelijk van een centrale provider en kan niet worden gecensureerd of ingetrokken door een derde partij.[17] Een DID verwijst naar een "DID Document", een JSON-bestand dat de publieke sleutels en service-eindpunten bevat die nodig zijn om met de DID-entiteit te interageren en diens identiteit te verifiëren.[19]
- **Verifiable Credentials (VCs):** Voortbouwend op DIDs, zijn Verifiable Credentials (eveneens een W3C-standaard) digitale, fraudebestendige claims.[20] Een VC is een verklaring (bijv. "Plugin X heeft een security audit van bedrijf Y doorstaan op datum Z") die cryptografisch is ondertekend door een
  issuer (uitgever, in dit geval bedrijf Y), wordt bewaard door een *holder* (houder, de ontwikkelaar van plugin X), en kan worden gepresenteerd aan een *verifier* (verifieerder,

het S1mpleTrader-platform).[21] De verifieerder kan de authenticiteit en integriteit van de VC controleren met behulp van de publieke sleutel van de issuer (die via diens DID kan worden gevonden), zonder ooit rechtstreeks contact op te nemen met de issuer. Dit maakt claims over softwarekwaliteit, audits of herkomst schaalbaar en onafhankelijk verifieerbaar.[22]

- **Smart Contracts voor "Programmable Enforcement":** Smart contracts zijn zelf-uitvoerende programma's die op een blockchain draaien. Ze kunnen worden gebruikt om regels en beleid op een transparante, onveranderlijke en geautomatiseerde manier af te dwingen.[24] In de context van S1mpleTrader kan een smart contract fungeren als een gedecentraliseerde plugin-registry. Een regel zou kunnen zijn: "Accepteer alleen een nieuwe versie van een plugin als de transactie een geldige, on-chain VC bevat die een succesvolle SLSA Level 2-build attesteert." De handhaving van dit beleid is niet langer afhankelijk van een centrale server, maar is ingebed in de code van het gedecentraliseerde netwerk zelf.[26]

Deze twee sporen zijn geen alternatieven; ze opereren op verschillende niveaus van de vertrouwenshiërarchie. Zero Trust is de operationele methodologie voor het veilig omgaan met componenten, uitgaande van het feit dat risico altijd aanwezig is. De cryptografische aanpak daarentegen verbetert de fundamentele betrouwbaarheid van die componenten door verifieerbare, positieve claims over hun eigenschappen te verschaffen. Een volwassen ecosysteem zal beide gebruiken: het zal Zero Trust-principes (zoals een strikte sandbox) toepassen, zelfs op een plugin van een zeer gereputeerde ontwikkelaar met sterke on-chain attestaties. De attestaties verminderen de *waarschijnlijkheid* dat de plugin kwaadaardig is, terwijl de Zero Trust-controles de *impact* verminderen als deze dat onverhoopt toch blijkt te zijn. Dit is een schoolvoorbeeld van een gelaagde verdediging (defense-in-depth).

D.2. Dependency Management: Het Indammen van de Supply Chain

De software supply chain is de meest waarschijnlijke en gevaarlijke aanvalsvector voor het S1mpleTrader-ecosysteem. De strategie voor dependency management is daarom niet slechts een "best practice", maar een kritieke, niet-onderhandelbare beveiligingscontrole. De aanpak is defensief en restrictief, gericht op het minimaliseren van de aanvalsoppervlakte door de introductie van externe code drastisch te beperken en volledig te controleren.

D.2.A Zero Trust Benadering (Vandaag)

De onmiddellijke strategie is gebaseerd op drie pijlers die samen de controle over de dependency-keten maximaliseren.

D.2.A.1. De "Geen requirements.txt" Regel

Het dynamisch installeren van dependencies door plugins via pip install -r requirements.txt is categorisch verboden. Deze aanpak introduceert onaanvaardbare risico's:

- **Niet-reproduceerbare builds:** Zonder strikte versie-pinning (bijv. requests==2.27.1) kan een pip install vandaag een andere versie van een library installeren dan morgen. Dit kan leiden tot subtiele bugs of, erger nog, het ongemerkt binnenhalen van een nieuw gecompromitteerde versie.[28]
- **Ongecontroleerde Transitive Dependencies:** Zelfs met pinning van directe dependencies, specificeert een requirements.txt doorgaans niet de exacte versies van de *onderliggende* (transitieve) dependencies. Een kwaadaardige update in een diepgenestelde dependency kan zo onopgemerkt blijven.[29]
- **Gevoeligheid voor Typosquatting:** Ontwikkelaars die handmatig een requirements.txt onderhouden, kunnen typefouten maken, waardoor ze per ongeluk een kwaadaardig, typosquatted pakket specificeren.[30]

Implementatie: De PluginEnrollmentService zal elk ingediend plugin-pakket (.stplugin) inspecteren. Als een requirements.txt-bestand (of vergelijkbare bestanden zoals Pipfile of pyproject.toml met dynamische dependencies) wordt aangetroffen, wordt het pakket onmiddellijk en automatisch geweigerd met een duidelijke foutmelding.

D.2.A.2. S1mpleTrader "Standard Library"

Om de functionaliteit van plugins niet onnodig te beperken, zal het S1mpleTrader-platform een gecureerde, versie-gepinde en vooraf gescreende set van veelgebruikte, hoogwaardige bibliotheken aanbieden. Deze "Standard Library" omvat essentiële pakketten voor data-analyse, numerieke berekeningen en technische analyse, zoals pandas, numpy, scipy, statsmodels en ta-lib.

De voordelen van deze aanpak zijn significant en maken van de Standard Library een kernonderdeel van de beveiligingsstrategie:

- **Gecentraliseerde Veiligheidsscreening:** Het S1mpleTrader-kernteam is verantwoordelijk voor het grondig doorlichten van elke bibliotheek en elke update voordat deze wordt opgenomen. Dit omvat statische analyse, CVE-scanning en licentie-audits. Dit centraliseert de beveiligingsinspanning en -expertise, wat efficiënter en effectiever is dan te vertrouwen op elke individuele plugin-ontwikkelaar.
- **Stabiliteit en Consistentie:** Alle plugins draaien tegen dezelfde, bekende versies van de bibliotheken. Dit elimineert het risico op versieconflicten en "dependency hell" tussen verschillende plugins.
- **Prestatie-optimalisatie:** Veelgebruikte bibliotheken kunnen door het platform worden voorgeladen in het geheugen, waardoor de opstarttijd van individuele plugins aanzienlijk wordt verkort.
- **Verkleinde Aanvalsoppervlakte:** Door de set van toegestane externe code te beperken tot een bekende, gecontroleerde lijst, wordt de totale aanvalsoppervlakte van het ecosysteem drastisch verkleind.[31]

Beheer en Governance: De Standard Library wordt beheerd via een strikt governanceproces. Het toevoegen van een nieuwe bibliotheek of het updaten van een bestaande vereist een formeel reviewproces. Alle wijzigingen worden gedocumenteerd in een openbaar changelog, en er wordt een duidelijk beleid voor backward compatibility en deprecation gehanteerd om de stabiliteit voor plugin-ontwikkelaars te garanderen.

D.2.A.3. Vendoring (“Bring Your Own Code”)

Voor niche-bibliotheken die niet in de Standard Library zijn opgenomen, moeten ontwikkelaars de broncode van de dependency direct meeleveren in hun plugin-pakket, doorgaans in een vendor/-directory. Dit proces wordt "vendoring" genoemd.

- **Trade-offs en Mitigatie:** Vendoring heeft nadelen, zoals een grotere omvang van het plugin-pakket en de verantwoordelijkheid voor het patchen van kwetsbaarheden die bij de ontwikkelaar ligt.[33] Het voordeel is echter absolute transparantie en controle: de exacte code die wordt uitgevoerd, is aanwezig in het pakket en kan worden geïnspecteerd. Om de nadelen te mitigeren, kan S1mpleTrader een interne lijst van "goedgekeurde vendored libs" bijhouden, wat de reviewtijd voor veelvoorkomende niche-bibliotheken kan versnellen.
- **Geautomatiseerde Scans bij Enrollment:** De PluginEnrollmentService zal een cruciale rol spelen bij het beveiligen van vendored code. Bij elke indiening worden de volgende stappen automatisch uitgevoerd:
  1. Het pakket wordt uitgepakt in een tijdelijke, geïsoleerde omgeving.
  2. De vendor/-directory wordt gescand met pip-audit. Deze tool controleert de meegeleverde dependencies tegen bekende kwetsbaarheden uit databases zoals de

Python Package Advisory Database (via OSV) [citation from user doc].
3. De CI/CD-pijplijn faalt automatisch als er kwetsbaarheden met een 'High' of 'Critical' ernst worden gevonden.
4. Tegelijkertijd worden licentiescans uitgevoerd om compatibiliteit met het S1mpleTrader-licentiemodel te garanderen.

D.2.B Blockchain & Cryptografie Benadering (Toekomst)

Dit spoor transformeert het interne, gecentraliseerde auditproces in een open, verifieerbaar en gedecentraliseerd systeem van claims over de kwaliteit en veiligheid van dependencies.

- **On-chain Attestaties voor Dependency-audits:** In dit model kunnen onafhankelijke derde partijen (zoals security-auditfirma's) diepgaande analyses uitvoeren op softwarebibliotheken. De resultaten van deze audits worden niet als een PDF-rapport gepubliceerd, maar als een cryptografisch ondertekende **attestatie**. Deze attestaties volgen standaarden zoals **in-toto** en **SLSA**.[35]
  - **in-toto** is een framework dat de stappen van een software supply chain definieert (de "layout") en bewijs ("links") genereert voor elke uitgevoerde stap.[37] Een audit kan worden gezien als een stap in deze keten.
  - **SLSA (Supply-chain Levels for Software Artifacts)** is een raamwerk dat een maturiteitsniveau (bijv. SLSA Level 1, 2, 3) toekent aan de beveiliging van het bouwproces van een software-artefact.[39] SLSA beveelt het gebruik van in-toto attestaties aan om de "provenance" (herkomst) vast te leggen.[42]
- **Praktisch Voorbeeld:** Een gerespecteerde security-auditor voert een handmatige code review en dynamische analyse uit op pandas versie 2.0. Na afronding genereren zij een in-toto attestatie die stelt: "Wij, [Naam Auditor], attesteren dat de code met hash sha256:abc... (overeenkomend met pandas v2.0) is geanalyseerd en voldoet aan de criteria voor SLSA Build Level 2." Deze attestatie, ondertekend met de private sleutel van de auditor, wordt vervolgens gepubliceerd op een publieke blockchain.
- **Policy-verificatie bij Installatie:** De PluginEnrollmentService kan worden geconfigureerd met een beleid dat on-chain verificatie vereist. Bijvoorbeeld: "Sta de installatie van een plugin alleen toe als alle vendored dependencies een on-chain attestatie hebben van een auditor uit de 'Trusted Auditors List' die minimaal SLSA Level 2 bevestigt." De service zou dan de blockchain-query uitvoeren en de cryptografische handtekening van de attestatie verifiëren. Dit verplaatst het vertrouwen van een interne, herhaalde scan naar een eenmalige, verifieerbare, on-chain claim.

D.3. Authenticiteit en Integriteit van Code-Artefacten

Het is essentieel om cryptografisch te kunnen garanderen dat de code van een plugin die op het S1mpleTrader-platform wordt geïnstalleerd, exact dezelfde code is als die de ontwikkelaar heeft geschreven en dat deze niet is gemanipuleerd tijdens de distributie.
Code-ondertekening is het mechanisme om deze authenticiteit en integriteit te waarborgen. De traditionele complexiteit van sleutelbeheer heeft de adoptie hiervan echter historisch belemmerd.

D.3.A Zero Trust: Keyless Code Signing met Sigstore

De Zero Trust-aanpak maakt gebruik van het moderne **Sigstore**-ecosysteem, dat is ontworpen om code-ondertekening drastisch te vereenvoudigen en toegankelijk te maken voor elke ontwikkelaar via een "keyless" model.[44] Dit model elimineert de noodzaak voor ontwikkelaars om langlevende private sleutels te beheren, wat een significant beveiligingsrisico en een operationele last is.

- **Kerntechnologieën:**
  - **sigstore-python:** De Python-implementatie van de Sigstore-client die de ondertekening en verificatie orkestreert.[45]
  - **Fulcio:** Een gratis, openbare Certificate Authority (CA) die kortlevende code-ondertekeningscertificaten uitgeeft. In plaats van identiteit te verifiëren via traditionele methoden, koppelt Fulcio een certificaat aan een OpenID Connect (OIDC) identiteit.[47]
  - **Rekor:** Een openbare, onveranderlijke transparantielog. Elke ondertekeningsgebeurtenis (de handtekening, het certificaat en de hash van het artefact) wordt opgeslagen in Rekor, waardoor een auditeerbaar en onweerlegbaar bewijs van de ondertekening ontstaat.[48]
- **Gedetailleerde Workflow voor Ondertekening (Export):**
  1. **Initiatie:** De PluginPackagingService (of een ontwikkelaar lokaal) roept sigstore sign artifact.stplugin aan.
  2. **OIDC-Authenticatie:** De ontwikkelaar wordt via de browser doorgestuurd naar een OIDC-provider (bijv. GitHub, Google, Microsoft). Na succesvolle login geeft de provider een kortlevend, cryptografisch ondertekend ID-token terug. Dit token bevat de geverifieerde identiteit van de ontwikkelaar (bijv. e-mailadres of GitHub-repository-URL).
  3. **Certificaatuitgifte:** sigstore-python genereert lokaal een efemeer (kortlevend) sleutelpaar (private/public key). Het stuurt de publieke sleutel samen met het OIDC ID-token naar Fulcio. Fulcio valideert het ID-token en geeft een kortlevend (meestal 10 minuten) X.509-certificaat uit dat de publieke sleutel bindt aan de OIDC-identiteit.

4. **Ondertekening Artefact:** Het plugin-artefact (.stplugin) wordt lokaal ondertekend met de efemere private sleutel.
5. **Publicatie in Transparantielog:** De handtekening, het Fulcio-certificaat en de hash van het artefact worden als één item naar de Rekor-transparantielog gestuurd. Rekor voegt het item toe, ondertekent de nieuwe staat van de log en geeft een "Signed Entry Timestamp" terug als bewijs van opname.
6. **Bundeling:** Alle bewijsmaterialen (handtekening, certificaat, Rekor-entry) worden samengevoegd in een "Sigstore bundle" (een .sigstore JSON-bestand), die naast het artefact wordt gedistribueerd.

- **Gedetailleerde Workflow voor Verificatie (Import):**
  1. **Initiatie:** De PluginEnrollmentService ontvangt artifact.stplugin en de bijbehorende artifact.stplugin.sigstore bundel.
  2. **Verificatie-aanroep:** De service roept sigstore.verify(...) aan met het artefact, de bundel en een beleid, zoals --cert-identity developer@email.com --cert-oidc-issuer https://accounts.google.com.
  3. **Stappen van Verificatie:**
     - De handtekening in de bundel wordt geverifieerd tegen de publieke sleutel in het certificaat en de hash van het artefact.
     - Er wordt gecontroleerd of het certificaat is uitgegeven door een vertrouwde Fulcio-root en of de geldigheidsperiode van het certificaat de tijd van de Rekor-entry omvat.
     - De identiteit in het certificaat (Subject Alternative Name) wordt vergeleken met het opgegeven beleid (--cert-identity).
     - De OIDC-uitgever in het certificaat wordt vergeleken met het beleid (--cert-oidc-issuer).
     - De integriteit van de Rekor-entry zelf wordt geverifieerd.
  4. **Beslissing:** Als alle stappen slagen, is de integriteit en authenticiteit van het artefact bewezen. De installatie kan doorgaan.
- **Revocation en Tijdsgebonden Vertrouwen:** Omdat de certificaten extreem kortlevend zijn, is traditionele revocation (intrekking) minder relevant. Vertrouwen is inherent tijdsgebonden. In geval van een compromittering van een ontwikkelaarsaccount, is de impact beperkt tot de levensduur van de OIDC-tokens. De primaire respons is het ongeldig verklaren van de handtekeningen van alle bekende kwaadaardige artefacten in een denylist die door de PluginEnrollmentService wordt gehandhaafd (zie D.7).

D.3.B Blockchain & Cryptografie: Onveranderlijke Identiteit en Reputatie

Dit spoor versterkt het Sigstore-model door de centrale componenten (Fulcio, Rekor) te

vervangen door gedecentraliseerde alternatieven die robuuster zijn tegen censuur en aanvallen op één enkele entiteit.

- **Immutable Ledger als Transparantielog:** In plaats van Rekor wordt een publieke of permissioned blockchain gebruikt om de hashes van de plugin-artefacten en de handtekeningclaims op te slaan. Dit biedt een hogere mate van onveranderlijkheid en fraudebestendigheid, aangezien het manipuleren van de log een consensusaanval op het gehele blockchain-netwerk zou vereisen, in plaats van het compromitteren van één enkele log-operator.[51]
- **DIDs als Fundament voor Reputatie:** De identiteit van de ontwikkelaar wordt niet langer vertegenwoordigd door een tijdelijke OIDC-claim, maar door een persistente, door de ontwikkelaar zelf beheerde Decentralized Identifier (DID).[17] Elke nieuwe, ondertekende release wordt on-chain gekoppeld aan deze DID. Na verloop van tijd bouwt een ontwikkelaar een openbaar verifieerbaar trackrecord op van betrouwbare releases. Deze reputatie is overdraagbaar tussen platforms en is cryptografisch gebonden aan de DID van de ontwikkelaar, niet aan een account bij een derde partij zoals GitHub.[52]
- **Revocation via Smart Contract:** De status van een ontwikkelaars-DID of een specifieke publieke sleutel kan worden beheerd door een smart contract. Het intrekken van een gecompromitteerde sleutel wordt een transactie op de blockchain. Deze transactie is atomair en onmiddellijk zichtbaar voor alle deelnemers van het ecosysteem. De PluginEnrollmentService kan bij elke verificatie de status van de DID in het smart contract controleren, wat een veel sneller en betrouwbaarder revocatiemechanisme is dan traditionele Certificate Revocation Lists (CRLs).

D.4. Proactieve Inspectie: Statische en Beleidsanalyse

Voordat een plugin ooit wordt uitgevoerd, moet de code proactief worden geïnspecteerd op potentiële veiligheidsrisico's, kwaadaardige patronen en afwijkingen van het aangevraagde permissiemodel. Deze geautomatiseerde analyse vormt een cruciale poortwachterfunctie in de PluginEnrollmentService, die fungeert als een eerste, diepgaande verdedigingslinie.

D.4.A Zero Trust Workflow

De Zero Trust-aanpak implementeert een verplichte, geautomatiseerde Static Application Security Testing (SAST)-pijplijn die elke ingediende plugin doorloopt.

- **Geïntegreerde Tooling:** De pijplijn combineert meerdere toonaangevende tools om een

brede dekking van potentiële problemen te garanderen:

- **Bandit:** Een op Abstract Syntax Tree (AST) gebaseerde security linter, specifiek voor Python. Bandit is uitermate geschikt voor het detecteren van veelvoorkomende, bekende kwetsbaarheden.[54] Relevante checks voor de S1mpleTrader-context zijn onder meer:
  - B105: hardcoded_password_string: Voorkomt het lekken van API-sleutels, wachtwoorden of andere geheimen in de broncode.[46]
  - B307: eval: Detecteert het gebruik van de eval()-functie, die kan leiden tot willekeurige code-uitvoering als deze wordt blootgesteld aan onbetrouwbare input.
  - B322: input: Waarschuwt voor het gebruik van input(), wat in een server-side context onverwacht gedrag kan veroorzaken.
  - B404: import_subprocess: Signaleert het gebruik van de subprocess-module, wat een indicatie is dat de plugin processen wil uitvoeren.
  - B506: yaml_load: Waarschuwt voor het gebruik van yaml.load() zonder de Loader=yaml.SafeLoader parameter, wat kan leiden tot willekeurige code-uitvoering via onveilige deserialisatie.
- **Semgrep:** Een geavanceerde, op patronen gebaseerde SAST-tool die de semantische structuur van code begrijpt. Semgrep is krachtiger dan traditionele linters en stelt ons in staat om op maat gemaakte, contextspecifieke regels te definiëren die verder gaan dan de generieke checks van Bandit.[59]
  - **Voorbeeld van een Custom Semgrep Regel:** Een belangrijk risico is het laden van data van een externe, potentieel door de gebruiker beïnvloede URL. De pandas.read_csv()-functie kan URLs als input accepteren.[61] Een aanvaller zou dit kunnen misbruiken voor Server-Side Request Forgery (SSRF) om interne netwerkdiensten te scannen. Een custom Semgrep-regel kan dit detecteren:

    ```yaml
    rules:
      - id: pandas-read-csv-from-variable
        languages: [python]
        message: >-
          Het gebruik van pandas.read_csv() met een niet-statische URL kan een SSRF-risico vormen.
          Verifieer dat de bron van de URL ($URL) vertrouwd is en valideer de input.
        severity: WARNING
        patterns:
          - pattern: pandas.read_csv($URL,...)
          - pattern-not: pandas.read_csv("http://...",...)
          - pattern-not: pandas.read_csv("https://...",...)
    ```

    Deze regel vindt alle aanroepen naar read_csv waar het eerste argument een variabele is, en niet een hardgecodeerde string die begint met http.
- **pip-audit:** Zoals beschreven in D.2, wordt deze tool gebruikt om alle meegeleverde

(vendored) dependencies te scannen op bekende kwetsbaarheden (CVE's) op basis van de OSV-database.

- **Technische Workflow en Policy-as-Code Handhaving:**
  1. De PluginEnrollmentService pakt de .stplugin-bundel uit in een tijdelijke, geïsoleerde map.
  2. De service voert de scans uit en genereert machine-leesbare rapporten (JSON-formaat):
     ```
     Bash
     bandit -r /path/to/temp_plugin -f json -o bandit_report.json
     semgrep --config p/ci /path/to/temp_plugin --json -o semgrep_report.json
     pip-audit -r /path/to/temp_plugin/vendor/requirements.lock --format json -o pip_audit_report.json
     ```
  3. Een gespecialiseerde parser in de service analyseert deze rapporten.
  4. De kern van de handhaving is de **vergelijking van de bevindingen met de permissies** die zijn gedeclareerd in het plugin_manifest.yaml. Dit transformeert het manifest van louter metadata naar een afdwingbaar beveiligingscontract.
     - **Praktisch Voorbeeld:** Bandit rapporteert een B404: import_subprocess-bevinding. De parser controleert vervolgens de permissions.process_exec-sleutel in het manifest. Als deze permissie niet is aangevraagd (de lijst is leeg of de sleutel ontbreekt), wordt de plugin-installatie **onmiddellijk geweigerd**. De ontwikkelaar ontvangt een foutmelding die de exacte codelocatie, de gedetecteerde actie (subprocess import) en de vereiste, ontbrekende permissie (process_exec) specificeert.
- **Beleid en Uitzonderingen:**
  - **Obfuscatie en Binaries:** Code die is geobfusceerd (onleesbaar gemaakt) of die binaire bestanden bevat, wordt automatisch geweigerd. Inspectie van dergelijke code is onmogelijk en wordt als inherent onbetrouwbaar beschouwd.
  - **False Positives:** Voor gevallen waarin een tool een bevinding onterecht rapporteert, moet de ontwikkelaar een # nosec-commentaar toevoegen aan de betreffende regel code, inclusief een rechtvaardiging. Deze suppressies worden onderworpen aan een strenger handmatig code review-proces.
  - **Rechtvaardigingen:** Voor bepaalde risicovolle, maar legitieme, operaties kan een ontwikkelaar een justification blok in het manifest opnemen. Dit vereist een gedetailleerde beschrijving van waarom de permissie nodig is en welke mitigerende maatregelen zijn genomen.

D.4.B Blockchain & Cryptografie: Verifieerbare Security Claims

Dit spoor verheft de resultaten van de statische analyse van een intern controlemechanisme

tot een openbare, verifieerbare claim over de veiligheidspostuur van een plugin.

- **On-chain Attestaties van Scanresultaten:** Onafhankelijke en vertrouwde security-auditors kunnen dezelfde set van SAST-tools (Bandit, Semgrep, etc.) draaien op een plugin-artefact. In plaats van de resultaten in een rapport te publiceren, genereren ze een **in-toto attestatie** die de resultaten bevat. Deze attestatie, die cryptografisch is ondertekend met de sleutel van de auditor en de hash van de plugin bevat, wordt vervolgens opgeslagen op een blockchain. Dit creëert een onveranderlijk en publiek verifieerbaar bewijs dat een specifieke versie van een plugin is gescand door een specifieke auditor, met een specifiek resultaat.
- **Reputatie-badges als Verifiable Credentials (VCs):** Een ontwikkelaar die consistent plugins publiceert die alle geautomatiseerde scans doorstaan zonder kritieke bevindingen, kan een **Verifiable Credential** ontvangen.[21] Dit kan worden uitgegeven door het S1mpleTrader-platform zelf of door een geaccrediteerde derde partij. Deze VC, gekoppeld aan de DID van de ontwikkelaar, functioneert als een "badge of honor".[53] Het kan worden weergegeven in de Plugin Hub als een visueel teken van betrouwbaarheid en kwaliteit, waardoor gebruikers sneller een risico-inschatting kunnen maken. Omdat de VC cryptografisch verifieerbaar is, biedt het een veel sterkere garantie dan een simpele "top developer" badge op een traditioneel platform.

D.5. Gecontroleerde Uitvoering: Sandboxing en Runtime-Controles

Nadat een plugin is geverifieerd en geïnstalleerd, is de volgende kritieke fase de gecontroleerde uitvoering. Zelfs met rigoureuze proactieve inspectie kunnen er nog steeds onontdekte kwetsbaarheden of kwaadaardige logica in de code aanwezig zijn. Het doel van de runtime-controles is om de "blast radius" te minimaliseren: als een plugin kwaadaardig gedrag vertoont, moet de schade beperkt blijven tot de plugin zelf en mag deze het host-systeem, de S1mpleTrader-core of andere plugins niet kunnen beïnvloeden. Dit wordt bereikt door meerdere lagen van isolatie en handhaving van het "least privilege" principe op runtime.

D.5.A Zero Trust Benadering (Vandaag)

De Zero Trust-aanpak implementeert een gelaagd model van sandboxing, variërend van basis procesisolatie tot sterke, op virtualisatie gebaseerde technieken. De keuze van de technologie kan afhangen van het vertrouwensniveau van de plugin.

Baseline Isolatie: Proces-Sandboxing

Voor alle plugins wordt een basisniveau van isolatie afgedwongen met behulp van standaard OS-primitieven.

- **Procesisolatie met multiprocessing.Process:** Elke plugin wordt uitgevoerd in zijn eigen, aparte besturingssysteemproces, met behulp van Python's multiprocessing.Process.[63] Dit biedt fundamentele geheugenisolatie; het geheugen van het plugin-proces is gescheiden van het geheugen van de S1mpleTrader-core en andere plugins.
- **Gecontroleerde Communicatie via IPC:** Directe geheugentoegang tussen processen is onmogelijk. Alle communicatie verloopt via strikt gedefinieerde Inter-Process Communication (IPC) kanalen, zoals multiprocessing.Queue of multiprocessing.Pipe.[63] Een significant voordeel hiervan is dat alleen data die "pickleable" (serialiseerbaar) is, de grens tussen de processen kan passeren.[63] Dit voorkomt het lekken van complexe objecten, handles of pointers en dwingt een gestructureerde, op berichten gebaseerde communicatie af.

Versterkte Isolatie: Kernel Attack Surface Reduction

Voor plugins die een hoger risico vormen, wordt een extra laag van kernel-level hardening toegevoegd.

- **Syscall Filtering met seccomp-bpf:** De Linux-kernel biedt een krachtig mechanisme genaamd seccomp-bpf (secure computing mode met Berkeley Packet Filter) om de set van systeemoproepen (syscalls) die een proces kan aanroepen, drastisch te beperken.[65] De kernel is de ultieme autoriteit en een compromittering ervan is catastrofaal. Door de aanvalsoppervlakte van de kernel te verkleinen, wordt de kans op een succesvolle "container escape" of privilege-escalatie-aanval aanzienlijk verminderd.[66]
    - **Praktisch Voorbeeld:** Een plugin die alleen is bedoeld voor data-analyse en geen netwerk- of bestandstoegang nodig heeft, kan worden gestart met een seccomp-bpf profiel dat alleen syscalls zoals read, write (naar stdout/stderr), brk, mmap, en exit_group toestaat. Een poging om socket() of openat() aan te roepen, wordt direct door de kernel geblokkeerd en resulteert in het beëindigen van het proces.[67] Voor elk type plugin (bijv. 'signal', 'execution', 'planner') kan een specifiek, op maat gemaakt seccomp-bpf profiel worden gedefinieerd.

Maximale Isolatie (Optioneel, voor High-Risk Plugins)

Voor plugins die als zeer risicovol worden beschouwd (bijv. van onbekende ontwikkelaars of met toegang tot zeer gevoelige functies), kan worden gekozen voor nog sterkere, op virtualisatie gebaseerde isolatietechnologieën. Deze bieden een veel robuustere scheiding tussen de plugin en het host-besturingssysteem, ten koste van hogere prestatie-overhead.

- **User-space Kernel met gVisor:** gVisor is een open-source "application kernel", geschreven in Go, die een groot deel van de Linux-systeemoproep-API in user-space implementeert.[69] Wanneer een plugin in een gVisor-sandbox draait (via de OCI-runtime runsc), worden de meeste van zijn syscalls niet door de host-kernel afgehandeld, maar door de gVisor "Sentry"-proces.[71] De Sentry zelf communiceert met de host-kernel via een zeer beperkte set van ongeveer 20 syscalls, wat de aanvalsoppervlakte enorm verkleint.[65] gVisor biedt een uitstekende balans tussen sterke beveiliging en acceptabele prestaties voor I/O-gebonden workloads.[72]
- **MicroVMs met Firecracker:** Firecracker is een Virtual Machine Monitor (VMM) die is geoptimaliseerd voor het draaien van lichtgewicht virtuele machines (microVMs).[74] Elke plugin zou in zijn eigen, volledig geïsoleerde microVM kunnen draaien, compleet met zijn eigen uitgeklede Linux-kernel. Dit biedt de sterkst mogelijke isolatie, vergelijkbaar met traditionele VM's, maar met een opstarttijd van milliseconden en een minimale geheugen-footprint.[75] Firecracker is ideaal voor het veilig uitvoeren van volledig onbetrouwbare code, hoewel de prestatie-overhead voor I/O-intensieve taken hoger kan zijn dan bij gVisor.[76]

De volgende tabel biedt een vergelijkende analyse van deze sandboxing-technologieën om de keuze voor een specifieke plugin te ondersteunen.

Tabel 1: Vergelijkende Analyse van Sandboxing Technologieën

Technologie	Isolatiemodel	Beveiligingsgarantie	Prestatie-overhead	Opstarttijd	Primair Gebruiksscenario (S1mpleTrader)
multiprocessing.Process	OS Proces	Geheugenisolatie. Geen kernel-bescherming.	Zeer laag	Laag	Basis-isolatie voor alle vertrouwde, intern ontwikkelde

					plugins.
seccomp-bpf	Kernel Syscall Filter	Verkleint kernel-aanvalsoppervlakte. Voorkomt ongeautoriseerde syscalls.	Verwaarloosbaar	N.v.t. (toegepast op proces)	Standaard hardening-laag bovenop multiprocessing voor alle plugins.
gVisor (runsc)	User-space Kernel	Sterke isolatie van de host-kernel. Beschermt tegen kernel-exploits.	Matig (vooral bij I/O)	Laag (ms)	Plugins van geverifieerde, maar externe ontwikkelaars. Plugins die netwerk- of bestandstoegang vereisen.
Firecracker	MicroVM (KVM)	Maximale isolatie (hardware-geassisteerd). Eigen kernel per plugin.	Hoog (vooral bij I/O)	Zeer laag (ms)	Volledig onbetrouwbare of experimentele plugins. High-stakes plugins die maximale beveiliging vereisen.

Resource- en Capability-Controles

Naast isolatie worden strikte limieten opgelegd aan de resources die een plugin kan

verbruiken.

- **Quotas:** Met behulp van Linux cgroups worden harde limieten ingesteld op CPU-gebruik, geheugenallocatie en I/O-bandbreedte per plugin-proces of -container. Dit voorkomt dat een defecte of kwaadaardige plugin het hele systeem vertraagt of laat crashen door overmatig resourcegebruik.
- **Network Policy:** Standaard heeft een plugin **geen** netwerktoegang (default deny). Toegang tot specifieke domeinen of IP-adressen op specifieke poorten moet expliciet worden aangevraagd in het manifest en wordt afgedwongen via een firewall of, in een Kubernetes-context, via NetworkPolicy-objecten.[78]
- **Filesystem Policy:** De root-filesystem van de plugin wordt read-only gemount. Schrijftoegang tot specifieke bestanden of mappen (bijv. voor het opslaan van state) wordt verleend via expliciete bind mounts naar vooraf gedefinieerde, beperkte paden.

Handhaving en Auditing

Alle bovengenoemde controles worden strikt gehandhaafd. Een poging om een niet-toegestane syscall aan te roepen, resulteert in een PermissionError of het beëindigen van het proces. Een poging om een netwerkverbinding te openen die niet op de allowlist staat, wordt geblokkeerd. Al deze "policy violations" worden centraal gelogd en naar een observability-systeem gestuurd voor real-time monitoring en alarmering.

D.5.B Blockchain & Cryptografie Benadering (Toekomst)

- **Runtime Commits van Audit-logs:** Om de integriteit van de audit-logs te garanderen, kan het platform periodiek een cryptografische hash (een Merkle root) van de recente log-entries committeren aan een blockchain. Dit creëert een fraudebestendig en "tamper-evident" spoor. Als een aanvaller de centrale log-database zou manipuleren, zou de on-chain hash niet meer overeenkomen, wat de manipulatie onmiddellijk detecteerbaar maakt.
- **Economische Prikkels (Staking & Slashing):** Om ontwikkelaars economisch te binden aan het goede gedrag van hun plugins, kan een "staking"-mechanisme worden geïntroduceerd. Een ontwikkelaar moet een bepaald bedrag aan cryptocurrency "staken" (vastzetten) in een smart contract om hun plugin in de Hub te mogen publiceren. Als onweerlegbaar (cryptografisch) wordt bewezen dat de plugin kwaadaardig gedrag heeft vertoond (bijv. door een on-chain audit-log), kan de community of een governance-orgaan stemmen om de stake van de ontwikkelaar te "slashen" (in beslag te nemen).[80] Dit creëert een sterke financiële ontmoediging voor het publiceren van

malware.[82]

- **Build- en Runtime-attestaties:** De CI/CD-pijplijn die de sandbox-omgeving (bijv. de container-image) bouwt, kan zelf een **in-toto attestatie** publiceren die de exacte stappen en gebruikte materialen vastlegt. Voordat een plugin wordt geactiveerd, kan het S1mpleTrader-platform verifiëren dat de runtime-omgeving is gebouwd volgens de goedgekeurde, geattesteerde specificaties. Dit voorkomt dat een plugin in een ongeautoriseerde of gecompromitteerde omgeving wordt uitgevoerd.

D.6. Distributie en Ecosysteem

De manier waarop plugins worden gedistribueerd en geïnstalleerd is een kritiek controlepunt. De architectuur voorziet in meerdere distributiemodellen die verschillende niveaus van vertrouwen en controle bieden, waardoor zowel individuele gebruikers als grote organisaties een passend beveiligingsniveau kunnen kiezen.

D.6.A Zero Trust Benadering (Vandaag)

De Zero Trust-aanpak biedt een spectrum van distributieopties, van een sterk gecureerde centrale hub tot meer flexibele, door de gebruiker gecontroleerde installaties.

- **Gecureerde S1mpleTrader Plugin Hub:** Dit is het primaire en meest vertrouwde distributiekanaal. Plugins die in de officiële Hub worden aangeboden, ondergaan een rigoureus reviewproces door het S1mpleTrader-team. Dit proces omvat:
    1. **Handmatige Code Review:** Een security-engineer inspecteert de code op logische fouten, subtiele kwetsbaarheden en naleving van de best practices.
    2. **Verificatie van Ontwikkelaarsidentiteit:** De identiteit van de ontwikkelaar wordt geverifieerd.
    3. **Uitgebreide Geautomatiseerde Scans:** De plugin doorloopt de volledige SAST- en dependency-scanpijplijn zoals beschreven in D.4.
    4. **Gedragsanalyse in Sandbox:** De plugin wordt uitgevoerd in een zwaarbeveiligde sandbox om het daadwerkelijke runtime-gedrag te observeren en te vergelijken met de aangevraagde permissies.
    - **Officiële Ondertekening:** Na goedkeuring wordt de plugin ondertekend met de officiële S1mpleTrader-signing key. Deze sleutel fungeert als de "root of trust" voor het ecosysteem. Gebruikers en organisaties kunnen hun S1mpleTrader-installatie configureren om uitsluitend plugins te vertrouwen die met deze sleutel zijn ondertekend. Dit model is vergelijkbaar met de app stores van Apple of Google.

- **Self-Signed Distributie (voor Gevorderde Gebruikers):** Om een open ecosysteem te bevorderen, wordt het voor ontwikkelaars ook mogelijk gemaakt om hun plugins buiten de officiële Hub om te distribueren. In dit geval is de plugin ondertekend met de eigen (Sigstore/OIDC) sleutel van de ontwikkelaar.
    - **Expliciete Gebruikersconsent:** Wanneer een gebruiker een dergelijke "self-signed" plugin probeert te installeren, zal de S1mpleTrader-installer een duidelijke en niet-mis te verstane waarschuwing tonen. Dit "risk banner" zal de volgende informatie bevatten:
        - De identiteit van de ondertekenende ontwikkelaar.
        - Een volledige lijst van de door de plugin aangevraagde permissies (netwerk, bestandssysteem, etc.).
        - Een expliciete verklaring dat de plugin niet is beoordeeld door het S1mpleTrader-team.
    - De gebruiker moet actief en expliciet akkoord gaan met de installatie en de bijbehorende risico's.
- **Policy-Based Installatie voor Organisaties:** Grote organisaties (zoals hedge funds of prop trading firms) die S1mpleTrader gebruiken, hebben behoefte aan gecentraliseerd beheer en strikte governance. Zij kunnen niet toestaan dat individuele traders willekeurige plugins installeren.
    - **Policy Engine (OPA/Rego):** De installatie- en activatieprocessen worden gecontroleerd door een policy engine zoals **Open Policy Agent (OPA)**. OPA is een open-source, algemeen inzetbare policy engine die beleid evalueert dat is geschreven in de declaratieve taal **Rego**.[83]
    - **Voorbeeld van een Rego-beleid:** Een organisatie kan een beleid definiëren dat stelt: "Sta de installatie van een plugin alleen toe als (1) de plugin is ondertekend met de officiële S1mpleTrader Hub-sleutel, OF (2) de plugin is ondertekend door een ontwikkelaar die op onze interne 'vertrouwde ontwikkelaars'-lijst staat."
      Codefragment
      ```
      package s1mpletrader.plugin.install

      default allow = false

      # Regel 1: Toestaan als ondertekend door de officiële Hub
      allow {
        input.signature.issuer == "S1mpleTrader Official Hub"
        input.signature.valid == true
      }

      # Regel 2: Toestaan als ondertekend door een vertrouwde interne ontwikkelaar
      allow {
        trusted_developers := {"developer1@myfund.com", "developer2@myfund.com"}
        input.signature.identity_email_in trusted_developers
        input.signature.valid == true
      ```

```
}
```

- De S1mpleTrader-client op enterprise-niveau zou bij elke installatiepoging de metadata van de plugin als input naar de lokale OPA-engine sturen en de installatie alleen doorzetten als het allow-resultaat true is.

D.6.B Blockchain & Cryptografie Benadering (Toekomst)

Dit spoor decentraliseert de Hub-functionaliteit en integreert licentiebeheer en transparantie direct in de blockchain.

- **Smart-Contract Gebaseerde Hub:** De Plugin Hub is geen gecentraliseerde webserver meer, maar een set van smart contracts op een blockchain. Het registreren van een nieuwe plugin of het publiceren van een update is een transactie die naar het smart contract wordt gestuurd. Het contract kan automatisch regels afdwingen, zoals de vereiste van on-chain attestaties (zie D.2.B en D.4.B) voordat een nieuwe versie wordt geaccepteerd. Versiebeheer en de status van plugins (actief, verouderd, ingetrokken) zijn publiek en onveranderlijk vastgelegd.
- **Licenties als Tokens (NFTs):** De toegang tot of het gebruik van commerciële plugins kan worden beheerd via non-fungible tokens (NFTs) of semi-fungible tokens.
  - **ERC-721 (NFT):** Elke licentie is een unieke NFT. Een gebruiker moet het eigendom van de NFT in zijn wallet kunnen bewijzen om de plugin te activeren. Dit maakt licenties overdraagbaar en verhandelbaar op secundaire markten.
  - **ERC-1155 (Multi-Token):** Deze standaard is efficiënter voor het uitgeven van meerdere licenties (bijv. een "team-pakket" van 10 licenties). Een enkele token-ID kan meerdere, fungibele kopieën vertegenwoordigen.
  - De plugin-activatiecode zou on-chain de wallet van de gebruiker controleren op het bezit van het vereiste token voordat de plugin wordt gestart.
- **Transparantie van Ecosysteem-gebeurtenissen:** Elke belangrijke gebeurtenis—een nieuwe plugin-publicatie, een update, een installatie door een gebruiker, een intrekking—kan worden vastgelegd als een on-chain event. Dit creëert een volledig transparant en auditeerbaar logboek van de activiteit binnen het ecosysteem, met cryptografisch bewijs van de betrokken actoren (via hun DIDs) en de betreffende artefacten (via hun hashes).

D.7. Incident Response, Forensica en Compliance

Ondanks alle preventieve maatregelen is het essentieel om voorbereid te zijn op het moment dat er toch een beveiligingsincident plaatsvindt. Een robuust incident response (IR)-plan is cruciaal om de schade snel te beperken, de oorzaak te achterhalen, en herhaling te voorkomen. De aanpak is gebaseerd op de levenscyclus zoals gedefinieerd in de **NIST Special Publication 800-61 (Computer Security Incident Handling Guide)**.[85]

D.7.A Zero Trust Benadering (Vandaag)

De Zero Trust-aanpak richt zich op snelle detectie, geautomatiseerde inperking en gedetailleerde forensische analyse op basis van centraal verzamelde data. De volgende tabel vertaalt de fasen van de NIST IR-levenscyclus naar concrete, plugin-specifieke acties binnen het S1mpleTrader-ecosysteem.

Tabel 2: NIST SP 800-61 Incident Response Levenscyclus Toegepast op S1mpleTrader Plugins

Fase (NIST SP 800-61)	Beschrijving	S1mpleTrader Plugin-Specifieke Acties
1. Preparation (Voorbereiding)	Het opzetten van de tools, processen en training die nodig zijn om effectief op incidenten te kunnen reageren.	- **Tooling:** Implementeer een centraal observability-platform (bijv. ELK Stack, Splunk) dat logs verzamelt van de sandbox-omgeving, de PluginEnrollmentService en de core applicatie. - **Playbooks:** Ontwikkel specifieke playbooks voor scenario's zoals "kwaadaardige plugin gedetecteerd", "data-exfiltratie door plugin", en "plugin veroorzaakt DoS". - **Contactlijsten:** Houd een up-to-date lijst bij van ontwikkelaars van plugins voor snelle communicatie. -

		Training: Train het security-team in het analyseren van sandbox-logs en Sigstore-artefacten.
2. Detection & Analysis (Detectie & Analyse)	Het identificeren van een incident, het bepalen van de scope en het analyseren van de aard van de aanval.	- **Detectiemechanismen:** - **Drempelwaarden:** Automatische alerts bij afwijkend gedrag (bijv. CPU-pieken, ongebruikelijke netwerk-bursts, hoog geheugengebruik). - **Policy Violations:** Real-time alerts wanneer een plugin een geblokkeerde syscall aanroept, een netwerkverbinding buiten de allowlist probeert te openen, of naar een verboden bestandspad probeert te schrijven. - **Signature-based:** Scannen van plugin-code en runtime-geheugen op bekende malware-signaturen. - **Analyse:** Correlatie van alerts van verschillende bronnen. Analyse van de ondertekende audit-logs en scanrapporten (zie D.8) om de tijdlijn van de aanval te reconstrueren.
3. Containment, Eradication & Recovery (Inperking, Uitroeiing & Herstel)	Het beperken van de schade, het verwijderen van de oorzaak van het incident en het herstellen van de systemen naar een	- **Containment (Inperking):** - **Automatische Deactivatie:** Bij een high-confidence alert

	normale, veilige staat.	wordt de betreffende plugin onmiddellijk en automatisch gedeactiveerd en in quarantaine geplaatst (het proces wordt beëindigd, de plugin-bestanden worden verplaatst naar een veilige locatie). - **Netwerkisolatie:** Het IP-adres van de host wordt tijdelijk geïsoleerd van het productienetwerk. - **Eradication (Uitroeiing):** - **Herroeping van Handtekening:** De Sigstore-handtekening van de kwaadaardige plugin wordt gemarkeerd als ongeldig. De hash van het artefact wordt toegevoegd aan een centrale denylist die door alle PluginEnrollmentService-instanties wordt gebruikt om toekomstige installaties te blokkeren. - **Verwijdering:** De plugin wordt van alle systemen verwijderd. - **Recovery (Herstel):** - Herstel van eventuele beschadigde data vanuit backups. - Publicatie van een security advisory naar alle gebruikers met instructies.
4. Post-Incident Activity (Post-Incident Activiteit)	Het analyseren van het incident en de respons om lessen te trekken en de beveiliging en het IR-proces te verbeteren.	- **Lessons Learned:** Organiseer een post-mortem meeting. Analyseer de root cause. Was de kwaadaardige code te detecteren met de

		bestaande SAST-regels? Zo niet, ontwikkel een nieuwe Semgrep-regel. - **Rapportage:** Genereer een gedetailleerd incidentrapport. De machine-leesbare JSON-rapporten van de scans en tests (zie D.8) zijn cruciaal voor snelle triage en analyse. - **Verbetering:** Update de playbooks, SAST-regels, sandbox-profielen en het monitoringsysteem op basis van de bevindingen.

D.7.B Blockchain & Cryptografie Benadering (Toekomst)

Het gebruik van een blockchain voegt een laag van onweerlegbaarheid en transparantie toe aan het incident response-proces, wat met name waardevol is voor compliance en communicatie met externe partijen zoals toezichthouders.

- **On-Chain Audit Trail:** Alle belangrijke stappen in het incident response-proces—de detectie van het incident, de genomen inperkingsmaatregelen, de intrekking van de plugin—worden als transacties vastgelegd op een blockchain. Dit creëert een *append-only*, tijdgestempeld en onveranderlijk logboek van het incident. Dit is extreem waardevol voor toezichthouders, omdat het cryptografisch bewijs levert van de genomen maatregelen, zonder dat men hoeft te vertrouwen op de integriteit van de centrale log-servers van S1mpleTrader.
- **Verifieerbare Compliance:** In plaats van een rapport te sturen naar een regulator, kan S1mpleTrader cryptografisch bewijs leveren van de incident-tijdlijn. De regulator kan zelfstandig de on-chain data verifiëren. Dit model van "verifieerbare compliance" vermindert de afhankelijkheid van vertrouwen in rapportages en verplaatst dit naar vertrouwen in cryptografische protocollen. Het biedt een onweerlegbaar antwoord op de vragen "wie deed wat en wanneer?".

D.8. Plugin IDE, Levenscyclus en Kwaliteitsborging

Effectieve beveiliging begint bij de ontwikkelaar ("shift left"). De ontwikkelomgeving (IDE) en de continue integratie/continue deployment (CI/CD)-pijplijn moeten worden ontworpen om ontwikkelaars te begeleiden bij het schrijven van veilige code en om beveiligingscontroles zo vroeg en zo automatisch mogelijk in de levenscyclus te integreren.

D.8.A Zero Trust Benadering (Vandaag)

De Zero Trust-aanpak richt zich op het bieden van tools en geautomatiseerde workflows die veilige ontwikkeling tot de standaard maken.

- **Plugin Creator Wizard:** Om ontwikkelaars een vliegende start te geven en te zorgen dat ze de juiste structuur volgen, wordt een "wizard" of template-generator in de IDE geïntegreerd. Deze wizard genereert de boilerplate-code voor een nieuwe plugin, inclusief:
  - schema.py: Voor het definiëren van input- en output-data.
  - worker.py: De hoofdinlogica van de plugin.
  - plugin_manifest.yaml: Een vooraf ingevuld manifest met de meest restrictieve permissies als standaard (bijv. geen netwerk- of bestandstoegang). Dit dwingt de ontwikkelaar om bewust na te denken over welke permissies echt nodig zijn en deze expliciet aan te vragen.
- **Geïntegreerde Test Runner:** De IDE wordt geleverd met een geïntegreerde test-runner die gebruikmaakt van pytest. De runner is geconfigureerd om standaard een machine-leesbaar rapport te genereren met pytest --json-report. Deze JSON-rapporten worden later in de CI-pijplijn gebruikt om de kwaliteit en testdekking van de plugin automatisch te valideren en de resultaten weer te geven in de UI.
- **Continuous Security Pipeline (CI/CD):** Bij elke git push naar de repository van een plugin wordt een geautomatiseerde pijplijn getriggerd die de volledige levenscyclus van validatie en packaging simuleert:
  1. **(Re)Pack:** De plugin-code wordt gebundeld in het .stplugin-formaat.
  2. **Sign:** Het artefact wordt automatisch ondertekend met een Sigstore OIDC-identiteit die is gekoppeld aan de CI/CD-omgeving (bijv. de GitHub Actions-workflow-identiteit).
  3. **Scan:** De volledige suite van statische analyses (Bandit, Semgrep, pip-audit) wordt uitgevoerd op de gebundelde code. De pijplijn faalt als er kritieke kwetsbaarheden worden gevonden.
  4. **Sandbox Smoke Test:** De plugin wordt opgestart in een productie-representatieve

sandbox-omgeving om te controleren op opstartfouten en basisfunctionaliteit.

5. **Publiceer Artefact + Rapporten:** Als alle stappen slagen, worden het ondertekende .stplugin-artefact, de Sigstore-bundel, en alle JSON-rapporten (test-, scan- en validatierapporten) gepubliceerd als build-artefacten, klaar voor indiening bij de PluginEnrollmentService.

- **Manifest & Permissies als Contract:** Het plugin_manifest.yaml is het centrale contract dat de intenties van de ontwikkelaar vastlegt. Het principe van *least privilege* is verplicht. De PluginEnrollmentService gebruikt dit manifest als de "source of truth" om de bevindingen van de statische analyse tegen te valideren, zoals beschreven in D.4.
  - **Voorbeeld plugin_manifest.yaml (Uitgebreid):**

```yaml
# Unieke naam en versie van de plugin
name: market_structure_detector
version: "1.2.0"

# Metadata voor de S1mpleTrader-engine
type: structural_context
entry_class: MarketStructureDetector
schema_path: schema.py
params_class: MarketStructureParams
stateful: true # Geeft aan of de plugin state tussen runs moet behouden

# --- SECURITY & PERMISSIONS SECTIE ---
permissions:
  # Netwerktoegang: standaard leeg (geen toegang).
  # Expliciete allowlist van bestemmingen en doelen.
  network_access:
    - "exchange:read_quotes" # Symbolische permissie, vertaald naar specifieke API-endpoints
    - "https://api.some-news-service.com/v1/headlines"

  # Bestandssysteemtoegang: standaard leeg.
  # Expliciete allowlist van paden relatief aan de plugin-root.
  filesystem_access:
    - path: "./state.json"
      access: "rw" # Read-Write
    - path: "./config.json"
      access: "r"  # Read-Only

  # Procesuitvoering: standaard leeg (geen toegang).
  # Nodig voor plugins die externe tools aanroepen.
  process_exec:
```

```
# Input/Output contract
dependencies:
  inputs: ["high", "low", "close"]
  outputs: ["market_structure_state"]

# Beveiligingsverklaringen
security:
  # Verklaring dat de code niet geobfusceerd is (verplicht 'false')
  obfuscation: false
  # Verklaring dat de integriteit van vendored dependencies is gecontroleerd
  vendor_integrity: true
```

D.8.B Blockchain & Cryptografie Benadering (Toekomst)

- **DID-Integratie in de IDE:** De IDE kan worden geïntegreerd met een DID-wallet (bijv. een browser-extensie of een hardware-wallet). In plaats van te ondertekenen met een tijdelijke OIDC-identiteit, kan de ontwikkelaar commits en releases direct ondertekenen met hun persistente, zelfbeheerde DID.[87] De CI/CD-pijplijn kan vervolgens worden geconfigureerd om deze DID-handtekeningen te gebruiken voor het publiceren van on-chain attestaties (voor scans, builds, etc.).
- **Reputatie-badges/NFTs Zichtbaar in de Hub:** De Plugin Hub-interface kan direct on-chain data opvragen die is geassocieerd met de DID van een ontwikkelaar. Verifieerbare credentials, zoals "Heeft 10+ plugins gepubliceerd zonder kritieke CVE's" of "Heeft een externe security-audit doorstaan", kunnen als visuele badges worden weergegeven.[53] Dit biedt gebruikers een snelle, cryptografisch onderbouwde manier om het risico en de betrouwbaarheid van een ontwikkelaar en hun plugins te beoordelen.

D.9. Implementatiedetails en Richtlijnen

Deze sectie biedt concrete, technische richtlijnen en codevoorbeelden voor de implementatie van de kerncomponenten van de Zero Trust-architectuur.

D.9.1. Ondertekening en Verificatie (Sigstore)

De sigstore-python bibliotheek biedt zowel een command-line interface (CLI) voor handmatige operaties als een programmatische API voor integratie in de CI/CD-pijplijn.

- **Ondertekenen via CLI (voor ontwikkelaars):**
  Bash
  ```
  # Installeer sigstore
  python -m pip install sigstore

  # Onderteken het plugin-artefact. Dit start de OIDC-browser-flow.
  # De identiteit wordt gekoppeld aan het GitHub-account.
  python -m sigstore sign my_plugin.stplugin

  # Dit genereert een 'my_plugin.stplugin.sigstore' bundelbestand.
  ```

- **Verifiëren via CLI (voor handmatige controle):**
  Bash
  ```
  # Verifieer het artefact tegen de identiteit van de ontwikkelaar en de OIDC-uitgever.
  python -m sigstore verify identity \
    --bundle my_plugin.stplugin.sigstore \
    --cert-identity "https://github.com/login/oauth" \
    --cert-oidc-issuer "https://github.com/<gebruikersnaam>/<repo>@<ref>" \
    my_plugin.stplugin
  ```

  Referentie: [47]
- **Programmatische Verificatie (voor PluginEnrollmentService):**
  Python
  ```
  from sigstore.verify import Verifier
  from sigstore.verify.policy import Identity

  # Input bestanden
  artifact_path = "my_plugin.stplugin"
  bundle_path = "my_plugin.stplugin.sigstore"

  # Definieer het verificatiebeleid
  expected_identity = Identity(
      identity="developer@example.com",
      issuer="https://accounts.google.com"
  )

  verifier = Verifier.production()

  with open(artifact_path, "rb") as artifact, open(bundle_path, "r") as bundle:
  ```

```
try:
    verifier.verify(
        input_=artifact,
        bundle_file=bundle,
        policy=expected_identity
    )
    print("Verificatie succesvol!")
except Exception as e:
    print(f"Verificatie mislukt: {e}")
```

Referentie: [46]

D.9.2. Statische Analyse Profielen

- **Bandit:** Gebruik een baseline-configuratie die alle checks inschakelt, met een drempel voor severity en confidence op MEDIUM of hoger. Specifieke, potentieel luidruchtige checks kunnen per project worden uitgeschakeld via een bandit.yaml-configuratiebestand.[89]
- **Semgrep:** Start met de p/ci ruleset, een door de community gecureerde set van regels gericht op hoge signaal-ruisverhouding, ideaal voor CI/CD.[60] Breid dit uit met een S1mpleTrader-specifieke ruleset die de custom regels bevat (zoals de pandas-read-csv-regel uit D.4).
- **pip-audit:** Configureer de CI-pijplijn om te falen (exit-code!= 0) bij elke gevonden kwetsbaarheid met een HIGH of CRITICAL classificatie.

D.9.3. Sandbox-profielen

- **Seccomp:** Definieer JSON-gebaseerde seccomp-bpf profielen per type plugin. Een 'signal'-plugin heeft een veel beperktere set van toegestane syscalls nodig dan een 'execution'-plugin die mogelijk I/O-operaties uitvoert. Deze profielen worden samen met de OCI-runtime (zoals Docker of runsc) geconfigureerd.
- **Container Runtimes:** Voor high-risk plugins, configureer Kubernetes of Docker om de runsc (gVisor) runtime te gebruiken in plaats van de standaard runc. Voor Firecracker is een meer gespecialiseerde orchestrator nodig, maar het principe van het specificeren van een andere runtime blijft hetzelfde.

D.9.4. Policy Engine (OPA/Rego)

De Open Policy Agent (OPA) kan als een sidecar-container of een centrale service worden gedeployeerd. De PluginEnrollmentService en de runtime-omgeving maken API-aanroepen naar OPA om beleidsbeslissingen te verkrijgen.

- **Voorbeeld Rego-beleid voor Installatie:**
  Codefragment
  ```
  package s1mpletrader.admission.install

  # Standaard weigeren
  default allow = false

  # Toestaan als de handtekening geldig is en van de officiële Hub komt
  allow {
    input.artifact.signature.issuer == "S1mpleTrader Hub"
    input.artifact.signature.valid == true
  }

  # Toestaan als de handtekening geldig is, van een bekende ontwikkelaar komt,
  # en er geen kritieke CVE's zijn gevonden in de dependencies.
  allow {
    input.artifact.signature.identity == "trusted-dev@example.com"
    input.artifact.signature.valid == true
    count([scan | scan := input.scans.pip_audit.vulnerabilities[_]; scan.severity ==
  "CRITICAL"]) == 0
  }
  ```

  Referentie: [83]

D.9.5. Test- en Rapportageformaat

Standaardiseer op het JSON-formaat dat wordt geproduceerd door pytest-json-report. Dit zorgt voor consistente, machine-leesbare output die gemakkelijk kan worden geparsed, opgeslagen en weergegeven in de S1mpleTrader UI. De standaard commando in de CI-pijplijn is:

pytest --json-report --json-report-file=.report.json

D.10. Roadmap: Gefaseerde Implementatie

De implementatie van deze uitgebreide beveiligingsarchitectuur is een aanzienlijke onderneming. Een gefaseerde aanpak is essentieel om op een beheersbare en incrementele manier waarde te leveren, waarbij de meest kritieke risico's eerst worden gemitigeerd. De volgende roadmap schetst een logische volgorde van implementatie, van een basis-beveiligde omgeving voor solo-ontwikkeling tot een volledig gedecentraliseerd en cryptografisch gegarandeerd ecosysteem.

Tabel 3: Gefaseerde Implementatie Roadmap

Fase	Context / Doel	Aanpak	Sleuteltechnologieën en Bouwstenen	Succescriteria (Meetbaar)
Fase 0	**Initiële Ontwikkeling (Solo Dev, PoC)** Focus op basisintegriteit en het voorkomen van de meest voor de hand liggende fouten.	Zero Trust (Basis)	- **Code Signing:** sigstore-python CLI voor handmatige ondertekening en verificatie. - **Static Analysis:** Bandit geïntegreerd in de pre-commit hooks van de ontwikkelaar. - **Sandboxing:** Basisprocesisolatie met multiprocessing.Process en Queue voor IPC. - **Dependency**	- Alle uitgebrachte interne PoC-plugins zijn cryptografisch ondertekend. - De CI-pijplijn faalt als een Bandit-scan met HIGH severity mislukt. - Plugins draaien in aparte OS-processen.

			Control: Strikte "no requirements.txt" en "vendoring" policy, handmatig gehandhaafd.	
Fase 1	**Uitbreiding naar Eerste Gebruikers** Verharden van de beveiliging en automatiseren van controles.	Zero Trust (Hardening)	- **Geautomatiseerde CI/CD Pipeline:** Integratie van sigstore (met OIDC via GitHub Actions), Bandit, Semgrep (met p/ci ruleset), en pip-audit. - **Runtime Hardening:** Implementatie van seccomp-bpf profielen voor de belangrijkste plugin-types. - **Resource Management:** Implementatie van CPU- en geheugenquota's via cgroups. - **Observability:** Centrale logging van alle	- 100% van de ingecheckte code wordt automatisch ondertekend en gescand. - De pijplijn blokkeert code met kritieke CVE's in dependencies. - Er zijn gedefinieerde seccomp-profielen voor 'signal' en 'execution' plugins. - Een centraal dashboard toont alle policy violations.

			security-events (scans, verificaties, runtime-blokkades).	
Fase 2	**Openen voor Meerdere Ontwikkelaars / Organisatiegebruik** Focus op governance, beleidshandhaving en een gecureerd ecosysteem.	Policy & Governance	- **Plugin Enrollment Service:** Een geautomatiseerde service die de CI/CD-artefacten ontvangt en valideert. - **Policy Engine:** Implementatie van **Open Policy Agent (OPA)** om installatiebeleid af te dwingen. - **Gecureerde Plugin Hub:** Opzetten van de infrastructuur voor een officiële Hub met een handmatig reviewproces en S1mpleTrader-ondertekening.	- De installatie van een niet-ondertekende of afgekeurde plugin wordt programmatisch geblokkeerd door de OPA-integratie. - De eerste 5 plugins zijn succesvol gereviewd en gepubliceerd in de officiële Hub. - Organisaties kunnen een eigen OPA-beleid configureren om alleen Hub-plugins toe te staan.
Fase 3	**Integratie met Externe Partners en Auditors**	Attestations & Verifiable Claims	- **Attestation Generation:** De CI/CD-pijplijn	- Elke plugin in de Hub heeft een publiek beschikbare

	Verhogen van de transparantie en verifieerbaarheid van de supply chain.		wordt uitgebreid om **in-toto/SLSA** provenance-attestaties te genereren voor elke build. - **Attestation Verification:** De Plugin Hub en/of PluginEnrollmentService worden aangepast om deze attestaties te verifiëren en beleid af te dwingen (bijv. "vereist SLSA Level 2"). - **Publicatie van Attestaties:** Opzetten van een publiek eindpunt waar attestaties kunnen worden opgevraagd.	en verifieerbare SLSA Level 2-provenance attestatie. - Het is mogelijk om programmatisch te bewijzen welke broncode-commit heeft geleid tot een specifiek plugin-artefact. - Een externe partner kan een plugin indienen met een zelf-gegenereerde, geldige attestatie.
Fase 4	**High-Stakes Trading & Regelgevende Druk** Implementatie van een volledig gedecentraliseerd trust-model	Blockchain & Crypto-economics (Optioneel)	- **DID-Integratie:** Ondersteuning voor ontwikkelaars om hun plugins te ondertekenen met hun	- Een ontwikkelaar kan zijn reputatie, opgebouwd via zijn DID, meenemen naar andere platformen. - De registratie

	met economische garanties.		**Decentralized Identifier (DID)**. - **On-Chain Registry:** Migratie van de Hub-logica naar een **smart contract**. - **Tokenized Licensing:** Implementatie van **ERC-721/1155** voor licentiebeheer. - **Slashing-mechanismen:** Ontwerp en implementatie van een smart contract voor **staking en slashing**.	en intrekking van een plugin is een on-chain, onveranderlijke transactie. - Toegang tot een premium plugin kan worden gekocht en verkocht als een NFT. - Er is een gedocumenteerd, on-chain proces voor het bestraffen van bewezen kwaadwillende ontwikkelaars.

Conclusie

De voorgestelde tweeledige architectuur biedt een robuust en pragmatisch antwoord op de complexe beveiligingsuitdagingen van een plugin-ecosysteem in de high-stakes omgeving van financieel handelen.

Het **Zero Trust-spoor** levert onmiddellijke en concrete waarde. Door de principes van "never trust, always verify" rigoureus toe te passen op elk aspect van de plugin-levenscyclus—van het verbieden van dynamische dependencies en het afdwingen van cryptografische ondertekening met Sigstore, tot geautomatiseerde statische analyse en strikte runtime-sandboxing—wordt een gelaagde verdediging opgebouwd die de meest waarschijnlijke en gevaarlijke aanvalsvectoren effectief mitigeert. Deze aanpak is vandaag

implementeerbaar met volwassen, open-source technologie en legt een onmisbaar fundament voor elke verdere ontwikkeling.

Het **Blockchain & Moderne Cryptografie-spoor** is de strategische horizon. Het bouwt voort op het Zero Trust-fundament door de aard van vertrouwen zelf te transformeren. Waar Zero Trust risico beheert, streeft dit spoor ernaar risico te verminderen door vertrouwen te verankeren in wiskundige zekerheden in plaats van in centrale processen. Door concepten als Decentralized Identifiers, Verifiable Credentials, on-chain attestaties (SLSA/in-toto) en economische prikkels (slashing) te omarmen, creëert S1mpleTrader een pad naar een ecosysteem dat niet alleen veilig is, maar waarvan de veiligheid ook onafhankelijk, transparant en cryptografisch verifieerbaar is. Dit is niet alleen technologisch superieur, maar ook een krachtig signaal naar professionele gebruikers en toezichthouders die steeds hogere eisen stellen aan transparantie en onweerlegbaarheid.

Door beide sporen parallel te ontwerpen en de implementatie te faseren, kan S1mpleTrader V2 stapsgewijs naar een hoger niveau van beveiliging en betrouwbaarheid groeien. Het kan vandaag beginnen met het beschermen van zijn gebruikers en kapitaal, zonder de innovatiekracht van een open plugin-ecosysteem te smoren, en tegelijkertijd bouwen aan een toekomst waarin het platform de gouden standaard zet voor aantoonbare veiligheid in de financiële technologie.

Bronnen (Selectie)

- **Supply Chain & Threats:**
  - S1, S2, S3, S4, S5 (Algemene Supply Chain Attacks)
  - S6, S7, S8, S9, S10 (Python/PyPI Typosquatting)
  - S11, S12, S13, S14, S15 (Logic Bombs)
  - S16, S17, S18, S19, S20 (Privilege Abuse)
- **Zero Trust & Incident Response:**
  - S21, S22, S23, S24, S25 (NIST Zero Trust Architecture)
  - S190, S191, S192, S193, S194, S195 (NIST SP 800-61 Incident Response)
- **Cryptografie & Decentralisatie:**
  - S90, S91, S92, S93, S94 (Sigstore & sigstore-python)
  - S95, S96, S97, S98, S99 (Rekor Transparency Log)
  - S26, S27, S28, S29, S30 (Decentralized Identifiers - W3C, DIF)
  - S31, S32, S33, S34, S35, S36 (Decentralized Reputation)
  - S37, S38, S39, S40, S41, S42 (Smart Contract Policy Enforcement)
  - S169, S170, S171, S172 (Slashing Mechanisms)
- **Attestations & Supply Chain Integrity:**
  - S61, S62, S63, S64, S65, S66 (in-toto Attestations)

    - S67, S68, S69, S70, S71, S72, S73, S74 (SLSA Framework)
- **Tooling & Implementatie:**
    - S49, S50, S51, S52, S53, S54 (Python Dependency Management)
    - S100, S101, S102, S103, S104 (Bandit SAST)
    - S105, S106, S107 (Semgrep SAST)
    - S146, S147, S148, S149, S150, S151 (Python Multiprocessing)
    - S152, S153, S154, S155, S156 (Linux Seccomp-bpf)
    - S132, S133, S134, S135, S136, S137 (gVisor)
    - S138, S139, S140 (Firecracker)
    - S184, S185, S186, S187, S188, S189 (Open Policy Agent - OPA/Rego)
    - S157, S158, S159, S160, S161, S162 (Kubernetes Network Policies)
    - S163, S164, S165, S166, S167, S168 (Kubernetes Resource Quotas)
    - S196, S197, S198, S199, S200 (Plugin Manifests)

Geciteerd werk

1. What is the NIST SP 800-207 cybersecurity framework? - CyberArk, geopend op oktober 2, 2025, https://www.cyberark.com/what-is/nist-sp-800-207-cybersecurity-framework/
2. Supply chain attack - Wikipedia, geopend op oktober 2, 2025, https://en.wikipedia.org/wiki/Supply_chain_attack
3. Secure software supply chain management in the financial sector - Fluid Attacks, geopend op oktober 2, 2025, https://fluidattacks.com/blog/supply-chain-financial-sector
4. Software Supply Chain Attacks Risk on the Rise | Ivanti, geopend op oktober 2, 2025, https://www.ivanti.com/blog/software-supply-chain-attack-risk
5. Software Supply Chain Attacks - DNI.gov, geopend op oktober 2, 2025, https://www.dni.gov/files/NCSC/documents/supplychain/Software_Supply_Chain_Attacks.pdf
6. Top 15 software supply chain attacks: Case studies - Outshift | Cisco, geopend op oktober 2, 2025, https://outshift.cisco.com/blog/top-10-supply-chain-attacks
7. PyPI Inundated by Malicious Typosquatting Campaign - Check Point Blog, geopend op oktober 2, 2025, https://blog.checkpoint.com/securing-the-cloud/pypi-inundated-by-malicious-typosquatting-campaign/
8. A PyPI typosquatting campaign post-mortem - Phylum.io, geopend op oktober 2, 2025, https://blog.phylum.io/a-pypi-typosquatting-campaign-post-mortem/
9. Typosquatting Campaign Targets Python Developers with Hundreds of Malicious Libraries, geopend op oktober 2, 2025, https://rhisac.org/threat-intelligence/typosquatting-campaign-targets-python-developers-with-hundreds-of-malicious-libraries/
10. FYI: Threat actor targeting many PyPI packages with automated typosquatting campaign : r/cybersecurity - Reddit, geopend op oktober 2, 2025, https://www.reddit.com/r/cybersecurity/comments/10zrx55/fyi_threat_actor_targeting_many_pypi_packages/

11. Logic bomb - Wikipedia, geopend op oktober 2, 2025, https://en.wikipedia.org/wiki/Logic_bomb
12. Logic Bomb - 1Kosmos, geopend op oktober 2, 2025, https://www.1kosmos.com/security-glossary/logic-bomb/
13. What is a Logic Bomb? | Security Encyclopedia - HYPR, geopend op oktober 2, 2025, https://www.hypr.com/security-encyclopedia/logic-bomb
14. Privilege Misuse: Causes, Mitigation, and Remediation Strategies - Kiteworks, geopend op oktober 2, 2025, https://www.kiteworks.com/risk-compliance-glossary/privilege-misuse/
15. Privileged Identity Playbook - IDManagement.gov, geopend op oktober 2, 2025, https://www.idmanagement.gov/playbooks/pam/
16. Zero Trust Architecture - Glossary | CSRC - NIST Computer Security Resource Center, geopend op oktober 2, 2025, https://csrc.nist.gov/glossary/term/zero_trust_architecture
17. Decentralized Identifiers (DIDs) v1.0 - W3C, geopend op oktober 2, 2025, https://www.w3.org/TR/did-1.0/
18. Decentralized Identifiers (DIDs) v1.1 - W3C, geopend op oktober 2, 2025, https://www.w3.org/TR/did-1.1/
19. Decentralized Identifier Resolution (DID Resolution) v0.3 - W3C, geopend op oktober 2, 2025, https://www.w3.org/TR/did-resolution/
20. Verifiable Credentials: The Ultimate Guide 2025 - Dock Labs, geopend op oktober 2, 2025, https://www.dock.io/post/verifiable-credentials
21. Decentralized Identity: The Ultimate Guide 2025 - Dock Labs, geopend op oktober 2, 2025, https://www.dock.io/post/decentralized-identity
22. Trust Supply Chain | Secure & Transparent Solutions - EveryCRED, geopend op oktober 2, 2025, https://everycred.com/trust-supply-chain/
23. Enhancing Trust with Blockchain Verifiable Credentials - Protokol, geopend op oktober 2, 2025, https://www.protokol.com/insights/blockchain-verifiable-credentials/
24. How smart contracts can automate cybersecurity - Paubox, geopend op oktober 2, 2025, https://www.paubox.com/blog/how-smart-contracts-can-automate-cybersecurity
25. Automating Policy Enforcement With Smart Contracts - Chainlink Blog, geopend op oktober 2, 2025, https://blog.chain.link/policy-enforcement-via-smart-contracts/
26. Access Control in Solidity Smart Contracts: RBAC & Ownable Explained - Metana, geopend op oktober 2, 2025, https://metana.io/blog/access-control-in-solidity-smart-contracts/
27. Blockchain enabled policy-based access control mechanism to restrict unauthorized access to electronic health records - PMC, geopend op oktober 2, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11784709/
28. Do not use requirements.txt - quanttype, geopend op oktober 2, 2025, https://quanttype.net/posts/2023-10-31-do-not-use-requirements.txt.html
29. Hidden Dangers of Requirements.txt: How Dependency Pinning Can Save You -

Xygeni, geopend op oktober 2, 2025, https://xygeni.io/blog/hidden-dangers-of-requirements-txt-how-dependency-pinning-can-save-you/
30. 4 Reasons why Python libraries are not secure - Spectral, geopend op oktober 2, 2025, https://spectralops.io/blog/4-reasons-why-python-libraries-are-not-secure/
31. What Is Curated Content? Benefits, Tips, and Best Practices - Slate, geopend op oktober 2, 2025, https://slateteams.com/blog/what-is-curated-content
32. What Is Curated Content? Benefits, Tips & Best Practices To Know - Digital Guider, geopend op oktober 2, 2025, https://digitalguider.com/blog/what-is-curated-content/
33. Vendoring Policy - pip documentation v22.0.dev0, geopend op oktober 2, 2025, https://daobook.github.io/pip/zh-CN/development/vendoring-policy.html
34. Vendoring Policy - pip documentation v25.2, geopend op oktober 2, 2025, https://pip.pypa.io/en/stable/development/vendoring-policy/
35. in-toto attestations - Trustification, geopend op oktober 2, 2025, https://trustification.io/blog/2023/03/13/in-toto-attestations/
36. in-toto and SLSA, geopend op oktober 2, 2025, https://slsa.dev/blog/2023/05/in-toto-and-slsa
37. Layout Creation Example — in-toto 3.0.0 documentation, geopend op oktober 2, 2025, https://in-toto.readthedocs.io/en/latest/layout-creation-example.html
38. Getting started | in-toto, geopend op oktober 2, 2025, https://in-toto.io/docs/getting-started/
39. What is the SLSA Framework? - JFrog, geopend op oktober 2, 2025, https://jfrog.com/learn/grc/slsa-framework/
40. Introduction to SLSA - Chainguard Academy, geopend op oktober 2, 2025, https://edu.chainguard.dev/compliance/slsa/what-is-slsa/
41. SLSA • Security levels, geopend op oktober 2, 2025, https://slsa.dev/spec/v1.0/levels
42. Frequently asked questions - SLSA.dev, geopend op oktober 2, 2025, https://slsa.dev/spec/v1.0/faq
43. Frequently Asked Questions - SLSA.dev, geopend op oktober 2, 2025, https://slsa.dev/spec/v0.1/faq
44. Python - Sigstore, geopend op oktober 2, 2025, https://docs.sigstore.dev/language_clients/python/
45. sigstore - PyPI, geopend op oktober 2, 2025, https://pypi.org/project/sigstore/0.6.5/
46. A Sigstore client written in Python - GitHub, geopend op oktober 2, 2025, https://github.com/sigstore/sigstore-python
47. Sigstore Information | Python.org, geopend op oktober 2, 2025, https://www.python.org/downloads/metadata/sigstore/
48. Rekor - Sigstore, geopend op oktober 2, 2025, https://docs.sigstore.dev/logging/overview/
49. sigstore/rekor: Software Supply Chain Transparency Log - GitHub, geopend op oktober 2, 2025, https://github.com/sigstore/rekor

50. Setting up your own Rekor Transparency Log Server using Helm | by Andrew Block, geopend op oktober 2, 2025, https://medium.com/@sabre1041/setting-up-your-own-rekor-transparency-log-server-using-helm-fc7bbeafb59c
51. Why Blockchain Digital Credentials Are the Future | BCdiploma, geopend op oktober 2, 2025, https://www.bcdiploma.com/en/blog/digital-credentials-5-reasons-why-the-blockchain-is-a-great-success-with-students
52. The Dynamics of Blockchain-Based Reputation Systems - Cardano Spot, geopend op oktober 2, 2025, https://cardanospot.io/news/the-dynamics-of-blockchain-based-reputation-systems-g1lENIFMejOqlk0L
53. Decentralized Identity: Your Reputation Travels With You - a16z crypto, geopend op oktober 2, 2025, https://a16zcrypto.com/posts/article/decentralized-identity-on-chain-reputation/
54. Welcome to Bandit — Bandit documentation, geopend op oktober 2, 2025, https://bandit.readthedocs.io/
55. Bandit: Python Static Application Security Testing Guide - DEV Community, geopend op oktober 2, 2025, https://dev.to/angelvargasgutierrez/bandit-python-static-application-security-testing-guide-47l0
56. Bandit's Hardcoded Password Detection: Rules B105-B107 in Practice | McGinnis, Will, geopend op oktober 2, 2025, https://mcginniscommawill.com/posts/2025-05-27-hardcoded-password-detection-b105-b107/
57. hardcoded-password-string (S105) | Ruff - Astral Docs, geopend op oktober 2, 2025, https://docs.astral.sh/ruff/rules/hardcoded-password-string/
58. Bandit Security Rules: Complete Python Vulnerability Guide | McGinnis, Will, geopend op oktober 2, 2025, https://mcginniscommawill.com/posts/2025-05-26-bandit-security-rules-complete-guide/
59. Writing Semgrep rules, geopend op oktober 2, 2025, https://semgrep.dev/blog/2020/writing-semgrep-rules-a-methodology/
60. Run rules - Semgrep, geopend op oktober 2, 2025, https://semgrep.dev/docs/running-rules
61. pandas.read_csv — pandas 2.3.3 documentation - PyData |, geopend op oktober 2, 2025, https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
62. How to Read CSV Directly from a URL in Pandas and Requests - DataScientYst, geopend op oktober 2, 2025, https://datascientyst.com/how-to-read-csv-directly-from-a-url-in-pandas-and-requests/
63. multiprocessing — Process-based parallelism — Python 3.13.7 ..., geopend op oktober 2, 2025, https://docs.python.org/3/library/multiprocessing.html
64. Python multiprocessing.Queue vs multiprocessing.manager().Queue() - GeeksforGeeks, geopend op oktober 2, 2025,

https://www.geeksforgeeks.org/python/python-multiprocessing-queue-vs-multiprocessing-manager-queue/
65. Sandboxing and Workload Isolation - Fly.io, geopend op oktober 2, 2025, https://fly.io/blog/sandboxing-and-workload-isolation/
66. Optimizing seccomp usage in gVisor, geopend op oktober 2, 2025, https://gvisor.dev/blog/2024/02/01/seccomp/
67. Python bindings added for libseccomp 2.2.0 - LWN.net, geopend op oktober 2, 2025, https://lwn.net/Articles/634391/
68. Reference Guide — seccomp documentation - pkgbuild.com, geopend op oktober 2, 2025, https://pkgbuild.com/~jelle/seccomp/function_reference.html
69. gVisor: The Container Security Platform, geopend op oktober 2, 2025, https://gvisor.dev/
70. What is gVisor?, geopend op oktober 2, 2025, https://gvisor.dev/docs/
71. Enhancing Container Security with gVisor: A Deeper Look into Application Kernel Isolation | by Gitesh Wadhwa | Medium, geopend op oktober 2, 2025, https://medium.com/@GiteshWadhwa/enhancing-container-security-with-gvisor-a-deeper-look-into-application-kernel-isolation-585af4652781
72. Kata Containers vs Firecracker vs gvisor : r/docker - Reddit, geopend op oktober 2, 2025, https://www.reddit.com/r/docker/comments/1fmuv5b/kata_containers_vs_firecracker_vs_gvisor/
73. Security Model - gVisor, geopend op oktober 2, 2025, https://gvisor.dev/docs/architecture_guide/security/
74. firecracker-microvm/firecracker: Secure and fast microVMs for serverless computing. - GitHub, geopend op oktober 2, 2025, https://github.com/firecracker-microvm/firecracker
75. MicroVMs: Scaling Out Over Scaling Up in Modern Cloud Architectures - OpenMetal, geopend op oktober 2, 2025, https://openmetal.io/resources/blog/microvms-scaling-out-over-scaling-up/
76. Impact of Secure Container Runtimes on File I/O Performance in Edge Computing - MDPI, geopend op oktober 2, 2025, https://www.mdpi.com/2076-3417/13/24/13329
77. Blending Containers and Virtual Machines:A Study of Firecracker and gVisor - Computer Sciences User Pages, geopend op oktober 2, 2025, https://pages.cs.wisc.edu/~swift/papers/vee20-isolation.pdf
78. Guide to Kubernetes Egress Network Policies - Red Hat, geopend op oktober 2, 2025, https://www.redhat.com/zh-tw/blog/guide-to-kubernetes-egress-network-policies
79. Enable a default deny policy for Kubernetes pods - Calico Documentation - Tigera, geopend op oktober 2, 2025, https://docs.tigera.io/calico/latest/network-policy/get-started/kubernetes-default-deny
80. What Is Slashing in Crypto and How Does it Affect You? - Everstake, geopend op oktober 2, 2025,

https://everstake.one/blog/what-is-slashing-in-crypto-and-how-does-it-affect-you
81. Slashing | Binance Academy, geopend op oktober 2, 2025, https://academy.binance.com/en/glossary/slashing
82. What Is Slashing in Crypto? How It Works and Why It Matters - Changelly, geopend op oktober 2, 2025, https://changelly.com/blog/what-is-slashing-in-crypto/
83. Open Policy Agent, geopend op oktober 2, 2025, https://openpolicyagent.org/
84. Introduction | Open Policy Agent, geopend op oktober 2, 2025, https://openpolicyagent.org/docs
85. Computer Security Incident Handling Guide - NIST Technical Series Publications, geopend op oktober 2, 2025, https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf
86. SP 800-61 Rev. 2, Computer Security Incident Handling Guide | CSRC, geopend op oktober 2, 2025, https://csrc.nist.gov/pubs/sp/800/61/r2/final
87. Decentralized Identifiers (DIDs): The Ultimate Beginner's Guide 2025 - Dock Labs, geopend op oktober 2, 2025, https://www.dock.io/post/decentralized-identifiers
88. Walkthrough of Decentralized Identity (DID) Network - tsmatz - WordPress.com, geopend op oktober 2, 2025, https://tsmatz.wordpress.com/2019/12/24/decentralized-identifiers-did-tutorial/
89. Bandit Documentation, geopend op oktober 2, 2025, https://bandit.readthedocs.io/_/downloads/en/1.5.1/pdf/
90. Bandit Documentation, geopend op oktober 2, 2025, https://bandit.readthedocs.io/_/downloads/en/1.7.5/pdf/
91. Using Open Policy Agent (OPA) with Terraform: Tutorial and Examples | env zero, geopend op oktober 2, 2025, https://www.env0.com/blog/open-policy-agent
92. Getting Started with Open Policy Agent (OPA): Installation, Setup, and Real-World Example, geopend op oktober 2, 2025, https://palak-bhawsar.hashnode.dev/getting-started-with-open-policy-agent-opa-installation-setup-and-real-world-example?source=more_articles_bottom_blogs