CERTYX

AUTONOMOUS CERTIFICATE LIFECYCLE MANAGEMENT

Instalacja agenta

Agent sieciowy CLM — instalacja i wykrywanie

JĘZYK	WERSJA	DATA
Polski	8550060f	1 listopada 2026

Spis treści

1 Jak działa 3
2 Przygotowanie 3
3 Adres dla agentów i lokalny serwer testowy 3
4 Pięć kroków 3
5 Weryfikacja pakietu 4
6 Pobieranie pakietów 4
7 Linux / systemd 5
8 Windows / Harmonogram zadań 10
9 Uruchomienie ręczne lub w kontenerze 12
10 Zakres i postęp 13
11 Mapa sieci 14
12 Dla zespołu bezpieczeństwa klienta 15
13 Ustawienia operacyjne 15
14 Cofnięcie dostępu i odzyskiwanie 16
15 Budowanie pakietów 16
16 SSH 17
17 Sekrety kontrolowane przez klienta i subskrypcje KRL 18
18 Tożsamości użytkowników SSH dla procesów 18
19 Cele dla certyfikatów klienta TLS 18
20 Konektory wdrożenia i przekaźnik SSH 19
21 Przesyłanie zapisanego wyniku wykonania 22
22 Chroniony odczyt konfiguracji 22
23 Niezależna weryfikacja w prywatnej sieci 23
24 Urzędy publiczne: DNS-01, EAB i DigiCert 24
25 Podpisane aktualizacje 24
26 Kontrola wiersza poleceń 25
27 Wymagania hosta Windows 26

1 Jak działa

Agent działa wewnątrz sieci klienta i łączy się wychodząco z CLM przez HTTPS. Platforma nie wymaga dostępu przychodzącego do prywatnych podsieci. Agent zgłasza prywatne sieci IPv4, do których jego host jest podłączony lub do których ma trasy (mapa sieci), pobiera zatwierdzony profil wykrywania, znajduje aktywne hosty w zatwierdzonych zakresach, łączy się z ich portami TLS (połączenie TCP i bezpośredni handshake TLS) i przesyła wyłącznie publiczne certyfikaty. Nie przesyła kluczy prywatnych. Skanowanie jest jawne: uzgadnia się je wcześniej z zespołem bezpieczeństwa klienta (zob. „Dla zespołu bezpieczeństwa klienta") i nigdy nie próbuje omijać ani mylić systemów wykrywania włamań; tempo, limity na host, okna czasowe i zwalnianie chronią wyłącznie sieć i hosty przed obciążeniem. Zainstaluj po jednym agencie w każdej strefie, z której ma być dostęp do wskazanych sieci. Rejestracja agenta wykrywającego nie uprawnia go do wdrażania certyfikatów.

2 Przygotowanie

Wybierz host Linux z systemd albo Windows z Harmonogramem zadań, który ma dostęp do skanowanych hostów. Pobierz pakiet dla właściwego systemu i architektury; zawiera runtime .NET. Polecenia konsoli używają `clm-agent-linux-x64.tar.gz` i `clm-agent-win-x64.zip`; install-linux.sh przyjmuje też hosty aarch64 z pakietem `clm-agent-linux-arm64.tar.gz`, jeśli operator go publikuje (w poleceniach zastąp `x64` przez `arm64`). Host musi ufać nazwie i łańcuchowi CA adresu HTTPS platformy. Adres dla agentów musi używać HTTPS, być osiągalny z hosta agenta i mieć certyfikat, któremu host ufa i którego unieważnienie może sprawdzić online; konsola przez nieszyfrowany HTTP na 127.0.0.1 nie obsłuży agentów (zob. „Adres dla agentów i lokalny serwer testowy"). Zapewnij wychodzący HTTPS do CLM i połączenia TCP do zatwierdzonych sieci. Agent nie otwiera nowego portu przychodzącego.

3 Adres dla agentów i lokalny serwer testowy

Agenci łączą się z jednym adresem HTTPS — „Adresem dla agentów" pokazanym po „Utwórz kod rejestracji". Musi on spełniać cztery warunki, które instalator sprawdza, zanim zużyje kod (`Clm.Agent check`): host agenta rozwiązuje nazwę i osiąga port; certyfikat serwera zawiera dokładnie tę nazwę hosta lub adres IP i prowadzi do certyfikatu głównego w systemowym magazynie zaufania hosta; certyfikat wskazuje punkt dystrybucji CRL (lub responder OCSP), który host agenta może pobrać przez HTTP; zegary różnią się o mniej niż cztery minuty. Operator konfiguruje go przez `Clm:Ingress:AgentHosts` (pierwszy wpis), opcjonalnie `Clm:Ingress:AgentPort`, gdy agenci używają innego portu niż konsola, oraz `Clm:Ingress:AgentTrustRootFile`, gdy certyfikat pochodzi z prywatnego CA: konsola publikuje wtedy ten certyfikat główny pod `/downloads/clm-agent-trust-root.crt` i pokazuje polecenia sprawdzenia jego sumy SHA-256 i dodania go do zaufanych (zob. CONFIGURATION.md). Bez hosta agentów konsola podaje adres HTTPS, przez który ją otwarto; otwarta przez serwer proxy (np. brzegowy serwer wdrożenia kontenerowego), przez który agenci nie mogą się uwierzytelnić, podaje tę nazwę hosta z portem `Clm:Ingress:AgentPort` (8443 we wdrożeniu kontenerowym, który przekazuje TLS bez zmian do nasłuchu agentów), a bez portu agentów nie podaje żadnego; otwarta przez nieszyfrowany HTTP nie podaje żadnego. W obu przypadkach wyjaśnia dlaczego.

Lokalny serwer testowy. `./start.sh` udostępnia konsolę tylko na tym komputerze (https://127.0.0.1:7243 z certyfikatem deweloperskim, http://127.0.0.1:5180 przekierowuje na ten adres), którego agenci na innych komputerach nie mogą ani osiągnąć, ani mu zaufać. Uruchom go z adresem tego komputera, który osiąga host agenta (adres w sieci LAN, nazwa DNS albo 172.17.0.1 dla kontenerów Docker na tym samym komputerze):

```
ip -4 addr          # albo: hostname -I  - wybierz adres osiągalny z hosta agenta
CLM_AGENT_HOST=192.168.1.20 ./start.sh
```

start.sh tworzy wtedy w `.local-agent-endpoint/` lokalne CA i certyfikat HTTPS dla tego adresu (tylko dla właściciela, nigdy nie trafia do repozytorium; klucz CA zostaje tam), dodaje nasłuch HTTPS dla agentów na porcie 5443 (`CLM_AGENT_PORT`), publikuje listę odwołań przez HTTP na porcie 5181 (`CLM_AGENT_CRL_PORT`) i konfiguruje host, port i certyfikat główny agentów. Wypisuje adres dla agentów; konsola podaje go razem z krokiem zaufania. Adres jest zapamiętywany dla kolejnego `./start.sh`; `CLM_AGENT_HOST=off ./start.sh` go wyłącza. Zezwól w zaporze tego komputera na przychodzące TCP 5443 i 5181 (na przykład `sudo ufw allow 5443/tcp` i `sudo ufw allow 5181/tcp`). Sama konsola pozostaje dostępna tylko pod 127.0.0.1 i localhost. To rozwiązanie do testów; w produkcji używa się nazwy hosta z certyfikatem z własnej PKI lub publicznego CA.

4 Pięć kroków

1. W widoku Agenci i cele wdrożenia wpisz unikalny identyfikator agenta i wybierz „Utwórz kod rejestracji". Konsola pokaże adres dla agentów, identyfikator organizacji i ponumerowane kroki instalacji z gotowymi do skopiowania poleceniami dla Linux i Windows. Pobierz plik jednorazowego kodu ważnego dziesięć minut i przed wszystkim innym skopiuj go do pustego katalogu roboczego na serwerze: archiwum pakietu jest udostępniane tylko z ważnym kodem rejestracji, dlatego pobieranie pakietu wysyła kod z tego pliku. Sprawdzenie kodu przy pobieraniu go nie zużywa; zużywa go dopiero rejestracja. Chroń go jak hasło; nie wpisuj jego zawartości do argumentów polecenia.
2. W katalogu roboczym pobierz pakiet i sprawdź jego sumę kontrolną SHA-256 (krok 3 w sekcji Linux albo Windows). Jeśli operator publikuje klucz podpisywania wydań, sprawdź też podpisane wydanie: pobierz `clm-agent-<rid>.release.json`, `clm-agent-<rid>.release.json.sig` i `verify-release.sh` z tego samego adresu `/downloads/` (`<rid>` to `linux-x64` albo `win-x64`; osobny adres dla agentów udostępnia te pliki, oba

weryfikatory pakietu oraz `verify-release.sh`), pobierz release-signing.pub i jego odcisk SHA-256 z zaufanego kanału (nie tylko z tego samego serwera) i przed rozpakowaniem uruchom `bash verify-release.sh release-signing.pub clm-agent-linux-x64.release.json clm-agent-linux-x64.tar.gz`; nie instaluj, gdy sprawdzenie się nie powiedzie. Bez opublikowanego klucza nie ma pliku `.sig` i ta kontrola podpisu nie ma zastosowania; takich wydań nie można instalować jako podpisanych aktualizacji. Następnie sprawdź podpis wydawcy względem odcisku certyfikatu wydawcy uzyskanego od dostawcy CLM niezależnym kanałem (zob. „Weryfikacja pakietu"); nie instaluj odrzuconego pakietu. Pakiety Windows w podpisanym wydaniu mają dodatkowo podpisy Authenticode.

3. Zainstaluj pakiet poleceniami pokazanymi w konsoli (krok po kroku w sekcjach Linux i Windows poniżej). Agent lokalnie wygeneruje tożsamość RSA 3072, a CLM zapisze jej publiczny fingerprint w odpowiedniej organizacji. PFX i plik hasła pozostają na hoście.
4. W CLM wybierz zarejestrowanego agenta i zatwierdź jego profil wykrywania: zakresy IPv4 CIDR (pomaga mapa sieci zgłoszona przez agenta; zakresy publiczne nigdy nie są podpowiadane i administrator musi je wpisać oraz potwierdzić), wykluczenia, tryb (`standard`: sprawdzenie aktywności i typowe porty; `full`: dodatkowo każdy port 1–65535 na aktywnych hostach), porty sprawdzania aktywności, typowe porty, tempo, liczbę połączeń na host oraz okna skanowania ze strefą czasową. Włącz profil. Agent sam sprawdza każdy profil przed nawiązaniem połączeń: najwyżej 64 zakresy, każdy co najmniej /16 i najwyżej 65 536 adresów, najwyżej 262 144 adresy w profilu i 256 wykluczeń (albo ostrzejsze limity skonfigurowane na serwerze), tylko poprawne zakresy i wykluczenia IPv4 bez ustawionych bitów hosta, nigdy adresy pętli zwrotnej, link-local, multicast ani zarezerwowane; zakresy publiczne, zakresy krótsze niż /20 i tryb `full` tylko po potwierdzeniu przez administratora. Odrzucony profil jest zgłaszany jako nieprawidłowy i nic nie jest skanowane. Lokalna lista `AllowedCidrs` (początkowo bezpośrednio podłączone sieci IPv4) jest twardą granicą tylko wtedy, gdy właściciel hosta ustawi `RestrictToAllowedCidrs` na `true`.
5. Sprawdź wynik: instalator kończy się poleceniem `sudo clm-agent status` („Result: connected"), konsola pokazuje „pierwszy kontakt odebrany" dla nowego agenta, a sekcja Zarejestrowani agenci jego pierwszy i ostatni kontakt. Później sprawdzaj ostatni kontakt, logi agenta i wyniki w inwentarzu. Wykryte certyfikaty początkowo nie mają przypisanego właściciela. Błędy TLS są zachowane; samo wykrycie nie potwierdza zaufania certyfikatu.

5 Weryfikacja pakietu

Każdy pakiet wydania ma dwa niezależne podpisy:

- Podpis kodu wydawcy (sprawdzany przed instalacją): `<archiwum>.sha256` to skrót wydania, a `<archiwum>.sha256.p7s` jego odłączony podpis CMS certyfikatem podpisu kodu wydawcy (opcjonalny plik `.p7s.tsr` zawiera znacznik czasu RFC 3161). W pakiecie `SHA256SUMS` zawiera skrót SHA-256 każdego pliku, a `SHA256SUMS.p7s` go podpisuje; brakujące, dodane lub zmienione pliki i dowiązania są odrzucane. Pakiety Windows mają dodatkowo podpisy Authenticode na `Clm.Agent.exe`, bibliotekach `Clm.*.dll` i skryptach PowerShell.
- Offline'owy manifest wydania ECDSA P-384 (`.release.json` i `.sig`) używany przez `verify-release.sh` i podpisane aktualizacje Linux (zob. „Podpisane aktualizacje").

Kotwicą zaufania podpisu kodu jest odcisk SHA-256 certyfikatu wydawcy. Nie bierz go z serwera pobierania: otrzymaj go od dostawcy CLM niezależnym kanałem (dokumentacja umowy, portal wsparcia lub informacje o wydaniu) i przypnij. Opcjonalnie wymagaj także łańcucha do CA wydawcy przez `--ca-file` / `-CaFile`. Weryfikacja działa offline i wymaga tylko bash, OpenSSL 3 i coreutils (Linux) albo Windows PowerShell 5.1 / PowerShell 7 (Windows).

```
bash verify-agent-package.sh --publisher-sha256 ODCISK clm-agent-linux-x64.tar.gz
```

```
Get-AuthenticodeSignature .\verify-agent-package.ps1
.\verify-agent-package.ps1 -PublisherSha256 ODCISK -Archive .\clm-agent-win-x64.zip
```

`VERIFIED` z kodem 0 oznacza zgodność podpisu, przypiętego wydawcy, przeznaczenia do podpisu kodu, ważności certyfikatu i wszystkich skrótów. `REFUSED: <powód>` z kodem 1 oznacza: nie instaluj (brak podpisu, inny wydawca, niezgodny skrót lub wygasły certyfikat). Pobierz pakiet ponownie, a gdy odmowa się powtarza, skontaktuj się z dostawcą. Z `--tsa-ca-file` (tylko weryfikator Linux) znacznik czasu RFC 3161 jest wymagany, a ważność certyfikatu jest sprawdzana na moment podpisu. `--allow-unsigned-development` / `-AllowUnsignedDevelopment` służy wyłącznie do wersji deweloperskich: akceptuje brak podpisu po sprawdzeniu skrótów i wyświetla wyraźne ostrzeżenie. Nigdy nie używaj go na hostach produkcyjnych. Konsola buduje polecenia instalacji z katalogu pakietów: gdy serwer publikuje pakiet bez podpisu wydawcy (brak `.sha256.p7s`, niepodpisana kompilacja deweloperska), polecenia mówią o tym w wierszu komentarza, nie pobierają pliku `.p7s`, sprawdzają sumę kontrolną i używają tych obejść deweloperskich; podpisany pakiet zawsze dostaje polecenia z przypiętym wydawcą. Oba weryfikatory sprawdzają też rozpakowany pakiet z `--directory <katalog>` / `-Directory <katalog>` (podpisany `SHA256SUMS`, każdy skrót i dokładny zestaw plików); właśnie to uruchamiają instalatory. W tym trybie w Windows weryfikator PowerShell dodatkowo sprawdza podpisy Authenticode `Clm.Agent.exe`, zestawów `Clm.*.dll` i skryptów PowerShell względem tego samego przypiętego wydawcy (łańcuch musi być zaufany na hoście albo podaj `-CaFile`); w Linux i macOS wypisuje uwagę i opiera się na podpisanym manifeście. Sprawdzenie archiwum (`-Archive`) weryfikuje tylko podpisany skrót archiwum.

Instalatory powtarzają weryfikację i przerywają pracę przed zmianą hosta. Ponieważ uruchamiają kod z samego pakietu, chronią przed uszkodzonym lub częściowo zmienionym pakietem i egzekwują przypięty odcisk; nie zastępują wcześniejszej weryfikacji archiwum osobno pobranym weryfikatorem.

6 Pobieranie pakietów

Archiwa pakietów (`/downloads/clm-agent-*.tar.gz` i `.zip`) zawierają kod wykonywalny i są udostępniane tylko żądaniu, które przedstawi jedno z poniższych:

- zalogowaną sesję konsoli: karty pakietów w widokach Agenci i cele wdrożenia oraz Automatyzacja SSH działają w przeglądarce. Każda karta pokazuje nazwę pliku archiwum, wersję, rozmiar, SHA-256 (z przyciskiem Kopiuj), czy opublikowano podpis wydawcy (`.p7s`) i podpisany manifest wydania (`.sig`), albo że pakiet nie jest opublikowany na tym serwerze; jej przyciski pobierają pakiet, jego pliki `.sha256` , `.p7s` i `.tsr` , manifest wydania z podpisem i weryfikator, każdy zapisywany pod własną nazwą pliku (serwer wysyła każdy plik z `/downloads/` jako załącznik). Karta pokazuje, które pliki podpisów istnieją; sprawdza je weryfikator na hoście;
- ważny, niewykorzystany i niewygasły kod rejestracji Twojej organizacji, wysłany w nagłówkach `X-CLM-Enrollment-Token` i `X-CLM-Tenant` (identyfikator organizacji). Polecenia konsoli odczytują kod z pliku clm-enrollment.token w katalogu roboczym; w Linux jest przekazywany potokiem do curl i nigdy nie pojawia się w wierszu polecenia. Sprawdzenie kodu go nie zużywa; kod jest zużywany raz, przez rejestrację, a dziesięciominutowa ważność obejmuje łącznie pobranie i instalację;
- certyfikat klienta TLS zarejestrowanego agenta wraz z jego organizacją (podpisane aktualizacje, zob. „Podpisane aktualizacje”).

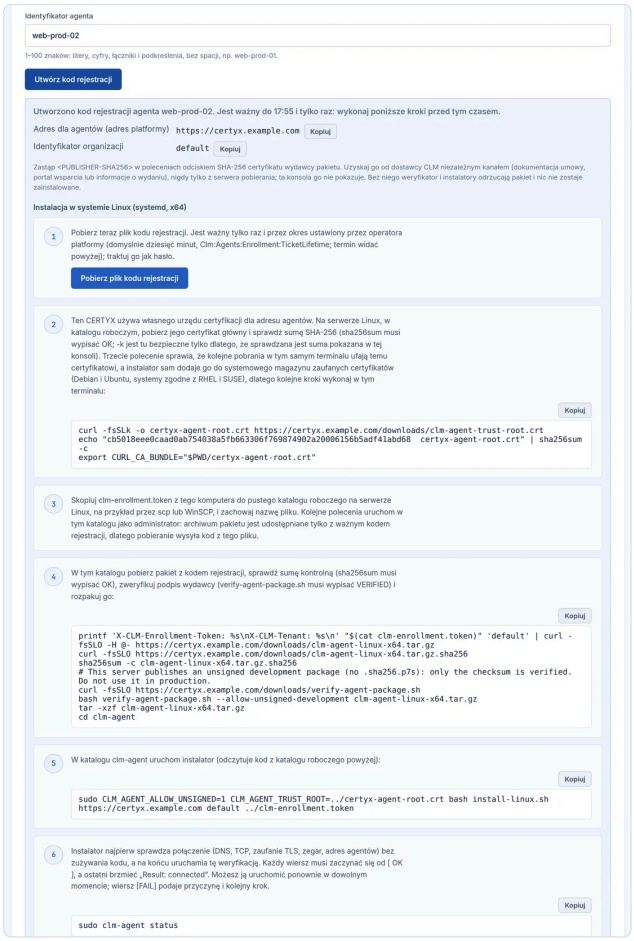
Każde inne żądanie otrzymuje HTTP 401, które curl zgłasza jako `The requested URL returned error: 401` , a PowerShell jako `(401) Unauthorized` . Oznacza to, że w bieżącym katalogu nie ma clm-enrollment.token, kod wygasł lub został już użyty albo identyfikator organizacji w poleceniu jest błędny: utwórz nowy kod dla tego samego identyfikatora agenta, skopiuj nowy plik do katalogu roboczego i uruchom polecenia ponownie. Sumy kontrolne (`.sha256`), podpisy (`.p7s` , `.tsr`), manifesty wydań (`.release.json` , `.sig`), skrypty weryfikujące i `clm-agent-trust-root.crt` nie zawierają kodu i pozostają publiczne, więc pakiet można zweryfikować bez poświadczeń.

7 Linux / systemd

Najszybciej i najbezpieczniej jest skorzystać z konsoli: po „Utwórz kod rejestracji” widok Agenci i cele wdrożenia pokazuje te kroki z już wpisanym prawdziwym adresem i identyfikatorem organizacji, gotowe do skopiowania. Poniższych poleceń używaj tylko wtedy, gdy nie możesz ich skopiować z konsoli; zastąp wtedy każdy symbol zastępczy w nawiasach ostrych, razem z nawiasami, prawdziwą wartością:

- `<CONTROL-PLANE-ORIGIN>` : adres HTTPS używany przez agentów, na przykład `https://agent.twoja-firma.pl` (bez ścieżki i bez końcowego ukośnika). Konsola pokazuje go po „Utwórz kod rejestracji” jako „Adres dla agentów”: jest to nazwa hosta agentów, jeśli operator ją skonfigurował (`Clm:Ingress:AgentHosts`), a w przeciwnym razie adres HTTPS, pod którym otwierasz konsolę (przez serwer proxy z portem agentów, np. `https://certyx.twoja-firma.pl:8443`). `https://clm.example.com` to tylko przykład; instalator go odrzuca.
- `<ORGANIZATION-ID>` : identyfikator Twojej organizacji, pokazany w tym samym miejscu jako „Identyfikator organizacji”.
- `<PUBLISHER-SHA256>` : odcisk SHA-256 certyfikatu wydawcy pakietów. Uzyskaj go od dostawcy CLM niezależnym kanałem (dokumentacja umowy, portal wsparcia lub informacje o wydaniu), nigdy tylko z serwera pobierania, i przypnij go (zob. „Weryfikacja pakietu”). Konsola go nie pokazuje.
- `<ADMIN-USER>` i `<SERVER>` : Twój login na serwerze Linux i nazwa hosta serwera; potrzebne tylko do skopiowania pliku kodu w kroku 2.

7.1 Krok 1: utwórz kod rejestracji



Agenci i cele wdrożenia po „Utwórz kod rejestracji": adres dla agentów, identyfikator organizacji i ponumerowane kroki z gotowymi poleceniami (dane przykładowe)

W CERTYX → Agenci i cele wdrożenia wpisz identyfikator agenta (litery, cyfry, - i _; jeden identyfikator na agenta), wybierz „Utwórz kod rejestracji", a potem „Pobierz plik kodu rejestracji". Kody tworzą tylko administratorzy; identyfikator już zarejestrowanego agenta jest odrzucany, więc dla agenta zastępczego wybierz nowy identyfikator. Kolejny kod dla tego samego, jeszcze niezarejestrowanego identyfikatora jest dozwolony; każdy kod działa samodzielnie. Plik clm-enrollment.token jest ważny dziesięć minut i tylko raz: wykonaj kroki 2–4 (kopiowanie, pobranie i instalację) w ciągu dziesięciu minut albo utwórz nowy kod dla tego samego identyfikatora agenta. Chroń go jak hasło i nigdy nie wklejaj jego zawartości do polecenia.

7.2 Krok 2: skopiuj plik kodu do katalogu roboczego

Na serwerze Linux utwórz pusty katalog roboczy i skopiuj do niego clm-enrollment.token z komputera, na który go pobrano, na przykład poleceniem `scp clm-enrollment.token <ADMIN-USER>@<SERVER>:` (katalog domowy na serwerze) albo przez WinSCP. Zachowaj nazwę pliku. Polecenia kroków 3 i 4 uruchamiaj z tego katalogu.

7.3 Krok 3: pobierz, sprawdź i rozpakuj pakiet

Jeśli konsola pokazuje krok „Ten CERTYX używa własnego urzędu certyfikacji" (operator ustawił `Clm:Ingress:AgentTrustRootFile`, na przykład lokalny serwer testowy), wykonaj go najpierw, w katalogu roboczym i w tym samym terminalu co kolejne kroki, bo już pobranie pakietu wymaga tego zaufania. Suma kontrolna jest pokazana w konsoli obok polecenia; `-k` jest tu bezpieczne tylko dlatego, że pobrany plik jest z nią porównywany, a `sha256sum` musi wypisać `certyx-agent-root.crt: OK`. Trzecie polecenie sprawia, że kolejne pobrania `curl` w tym terminalu ufają certyfikatowi głównemu; instalator sam dodaje go potem do systemowego magazynu zaufanych certyfikatów (`CLM_AGENT_TRUST_ROOT` w poleceniu instalacji, zob. krok 4), więc żadnego polecenia magazynu certyfikatów nie wpisuje się ręcznie:

```
curl -fsSLk -o certyx-agent-root.crt <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-trust-root.crt
echo "<SHA256-FROM-CONSOLE>  certyx-agent-root.crt" | sha256sum -c
export CURL_CA_BUNDLE="$PWD/certyx-agent-root.crt"
```

Następnie na serwerze Linux (x64 z systemd, wychodzący HTTPS do CERTYX) jako administrator w katalogu roboczym z plikiem clm-enrollment.token. Pierwsze polecenie wysyła kod z tego pliku, bo archiwum jest udostępniane tylko z ważnym kodem rejestracji (zob. „Pobieranie pakietów"):

```
printf 'X-CLM-Enrollment-Token: %s\nX-CLM-Tenant: %s\n' "$(cat clm-enrollment.token)" '<ORGANIZATION-ID>' | curl -fsSLO -H @- <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-linux-x64.tar.gz
curl -fsSLO <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-linux-x64.tar.gz.sha256
```

```
sha256sum -c clm-agent-linux-x64.tar.gz.sha256
curl -fsSLO <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-linux-x64.tar.gz.sha256.p7s
curl -fsSLO <CONTROL-PLANE-ORIGIN>/downloads/verify-agent-package.sh
bash verify-agent-package.sh --publisher-sha256 <PUBLISHER-SHA256> clm-agent-linux-x64.tar.gz
tar -xzf clm-agent-linux-x64.tar.gz
cd clm-agent
```

`sha256sum` musi wypisać `clm-agent-linux-x64.tar.gz: OK`; w przeciwnym razie nie instaluj. Suma kontrolna wykrywa uszkodzone pobranie. `verify-agent-package.sh` musi wypisać `VERIFIED`; `REFUSED: <powód>` oznacza: nie instaluj (zob. „Weryfikacja pakietu"). Jeśli operator publikuje klucz podpisywania wydań, sprawdź też podpisany manifest, jak opisano w kroku 2 sekcji „Pięć kroków". Jeśli `curl` zgłasza problem z certyfikatem, najpierw przeczytaj „Prywatne CA" poniżej; `The requested URL returned error: 401` przy pierwszym poleceniu oznacza, że brakuje kodu, wygasł lub został użyty albo identyfikator organizacji jest błędny (zob. „Pobieranie pakietów").

7.4 Krok 4: uruchom instalator

W katalogu clm-agent utworzonym przez `tar` (plik kodu pozostaje w katalogu roboczym nad nim, stąd `../clm-enrollment.token`):

```
sudo CLM_AGENT_PUBLISHER_SHA256=<PUBLISHER-SHA256> bash install-linux.sh <CONTROL-PLANE-ORIGIN> <ORGANIZATION-ID> ../clm-enrollment.token
```

Odcisk po `CLM_AGENT_PUBLISHER_SHA256=` jest wymagany: jeśli polecenie skopiowane z konsoli go nie zawiera, dodaj go jak pokazano. Przed zmianą hosta instalator kopiuje pakiet do prywatnego katalogu tymczasowego należącego do roota w /var/tmp, weryfikuje tę kopię (`SHA256SUMS`, jego podpis i przypiętego wydawcę) i instaluje dokładnie zweryfikowaną kopię. `CLM_AGENT_PUBLISHER_CA_FILE` dodatkowo wymaga łańcucha CA; `CLM_AGENT_VERIFIER=/sciezka/verify-agent-package.sh` używa osobno pobranego weryfikatora. `bash install-linux.sh --verify-only` sprawdza rozpakowany pakiet bez instalacji. Bez przypiętego odcisku albo z niepodpisanym lub zmienionym pakietem nic nie zostaje zainstalowane, a kod rejestracji nie zostaje zużyty. `CLM_AGENT_ALLOW_UNSIGNED=1` to wyłącznie deweloperskie obejście, które wypisuje ostrzeżenie. `CLM_AGENT_ENDPOINT_VERIFIER=<plik>` dodatkowo czyni host niezależnym weryfikatorem endpointu celu wdrożenia (zob. „Niezależna weryfikacja w prywatnej sieci").

Przy prywatnym certyfikacie głównym (krok zaufania w konsoli) polecenie pokazane przez konsolę przekazuje dodatkowo `CLM_AGENT_TRUST_ROOT=../certyx-agent-root.crt`: instalator sprawdza, czy plik jest certyfikatem CA ważnym jeszcze co najmniej dobę, i przed sprawdzeniem połączenia dodaje go pod nazwą `certyx-agent-root.crt` do systemowego magazynu zaufanych certyfikatów dystrybucji (Debian i Ubuntu: /usr/local/share/ca-certificates z `update-ca-certificates`; zgodne z RHEL: /etc/pki/ca-trust/source/anchors z `update-ca-trust`; SUSE: /etc/pki/trust/anchors). Identyczny wpis zostaje bez zmian; gdy sprawdzenie połączenia się nie powiedzie, wpis dodany w tym uruchomieniu jest ponownie usuwany, więc magazyn zaufanych certyfikatów hosta pozostaje niezmieniony.

Instalator sprawdza każdy argument, zanim cokolwiek zmieni, i wskazuje błędny argument razem z kolejnym krokiem. Potem uruchamia `Clm.Agent check` dla adresu (DNS, TCP, zaufanie TLS ze sprawdzeniem unieważnienia online, zegar, ścieżki agentów); nieudany krok jest wypisywany z przyczyną, nic nie jest zmieniane, a kod rejestracji nie zostaje zużyty, więc możesz usunąć problem i uruchomić to samo polecenie ponownie. Następnie tworzy nieuprzywilejowane konto clm-agent, instaluje wydanie w /opt/clm-agent/releases/<wersja> z dowiązaniem /opt/clm-agent/current, rejestruje agenta (agent generuje klucz RSA 3072 na tym serwerze; klucz nigdy go nie opuszcza), usuwa własną kopię kodu, przechowuje prywatną konfigurację, tożsamość i stan w /etc/clm-agent (`agent.json`, `identity.pfx`, `identity.password`, `state/`; dostęp tylko dla konta clm-agent) oraz zapisuje i uruchamia jednostkę systemd `/etc/systemd/system/clm-discovery.service`, która wykonuje `/opt/clm-agent/current/Clm.Agent scan /etc/clm-agent/agent.json` jako clm-agent. Na końcu instaluje pomocnika `/usr/local/bin/clm-agent` (`uninstall-linux.sh` z pakietu zostaje w /opt/clm-agent/current do usuwania), czeka, aż CERTYX zapisze pierwszy kontakt działającej usługi, wypisuje `sudo clm-agent status` i kończy komunikatem „registered and running". Następnie usuń clm-enrollment.token z katalogu roboczego; kod został zużyty. Agent nie potrzebuje surowych gniazd ani demona roota; root jest potrzebny tylko do utworzenia konta i usługi.

7.5 Krok 5 (opcjonalny): podpisane aktualizacje

Tylko gdy operator CERTYX publikuje klucz podpisywania wydań. Pobierz od operatora release-signing.pub i jego odcisk SHA-256 zaufanym kanałem (nie tylko z tego samego serwera), skopiuj klucz obok instalatora i dodaj go jako czwarty argument:

```
sudo CLM_AGENT_PUBLISHER_SHA256=<PUBLISHER-SHA256> bash install-linux.sh <CONTROL-PLANE-ORIGIN> <ORGANIZATION-ID> ../clm-enrollment.token ./release-signing.pub
```

Instalator wypisuje odcisk klucza; porównaj go z opublikowanym. Usługa aktualizacji pobiera nowe archiwa z certyfikatem klienta tego agenta, co instalator konfiguruje (`AgentConfigurationFile` w update.json). Bez czwartego argumentu agent działa, ale nie aktualizuje się automatycznie; aby włączyć podpisane aktualizacje później, uruchom `sudo bash upgrade-linux.sh ./release-signing.pub` z nowszego rozpakowanego pakietu (zob. „Aktualizacja w miejscu").

7.6 Krok 6: sprawdź, czy agent działa

```
sudo clm-agent status
```

Instalator wypisał to już raz; możesz to uruchomić w dowolnym momencie. Każdy wiersz musi zaczynać się od `[ OK ]`, a ostatni brzmieć `Result: connected`, na przykład:

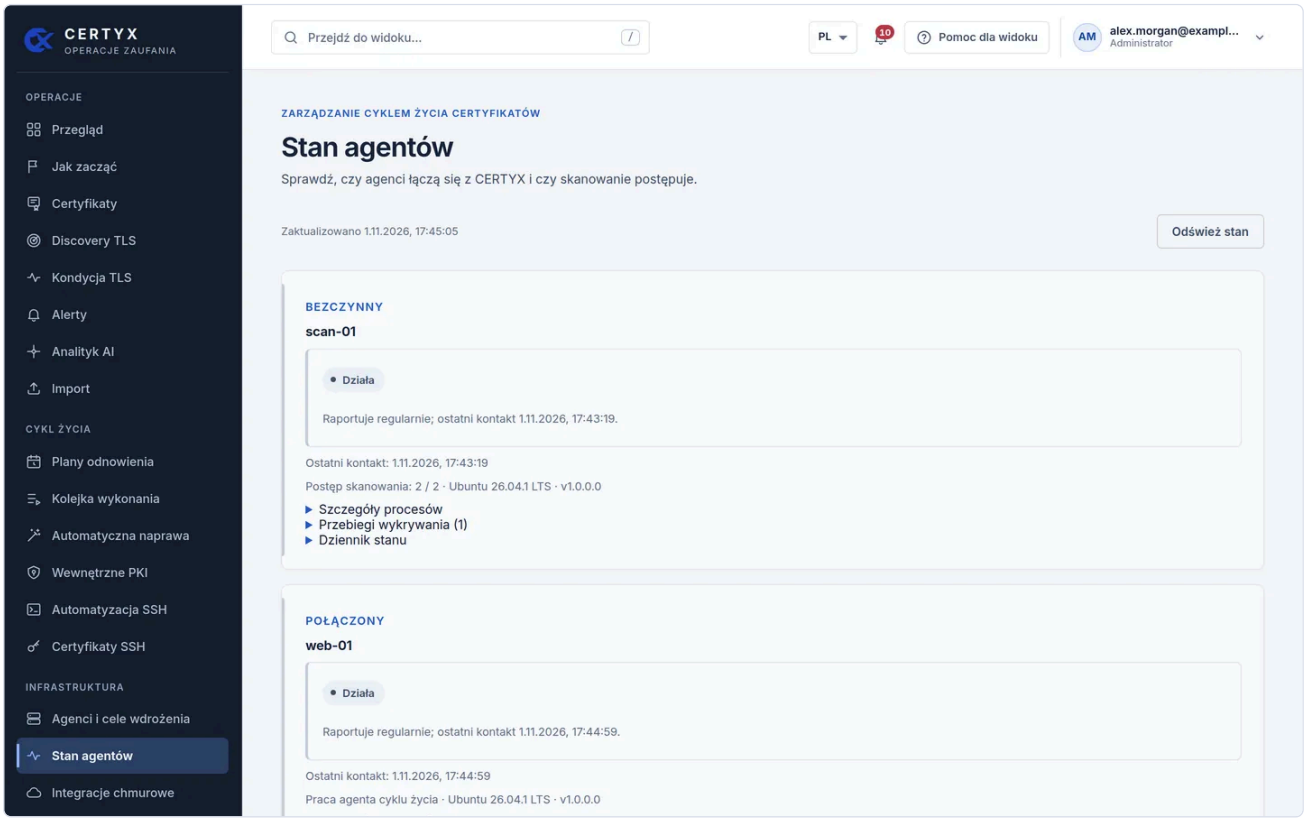
```
[ OK ] Service: clm-discovery.service is active (running).
CERTYX agent 1.0.0: status of /etc/clm-agent/agent.json
[ OK ] Configuration: agent address https://agent.your-company.com, organization acme.
[ OK ] Identity: SHA-256 3BCC36CD…43A67FC0, valid until 2027-04-08; the private key stays on this host.
[ OK ] DNS: agent.your-company.com resolves to 10.20.0.15.
[ OK ] TCP: connected to 10.20.0.15:443 in 5 ms.
[ OK ] TLS: certificate for agent.your-company.com issued by Company Issuing CA is trusted, matches agent.your-company.com, is not revoked and is valid until 2027-11-11.
[ OK ] Clock: this host's clock matches the server (difference 0 s).
[ OK ] Agent address: agent.your-company.com serves agents.
[ OK ] Registration: CERTYX accepts this host as agent "berlin-dc1" in organization "acme"; first contact 2026-10-10 18:30:33 UTC, last contact 2026-10-10 18:31:03 UTC.
Result: connected. CERTYX accepts this agent; its contact times are shown above and in CERTYX -> Agents and destinations.
```

Polecenie działa jako konto agenta i sprawdza usługę, konfigurację, lokalną tożsamość (odcisk i ważność certyfikatu), rozwiązywanie nazwy, połączenie TCP, zaufanie TLS z unieważnieniem, różnicę zegarów i to, czy CERTYX akceptuje tożsamość, razem z zapisanym przez CERTYX pierwszym i ostatnim kontaktem. Kod wyjścia 0 oznacza działanie, 1 problem (pierwszy wiersz `[FAIL]` go nazywa, zob. „Rozwiązywanie problemów"). `sudo clm-agent check` uruchamia tylko kroki usługi i połączenia; `sudo clm-agent log [wiersze]` pokazuje ostatnie wiersze logu usługi (domyślnie 50); `clm-agent version` wypisuje zainstalowane wydanie. W CERTYX konsola zaraz po „Utwórz kod rejestracji" pokazuje „Agent … działa: pierwszy kontakt odebrany …", gdy usługa się zgłosi, a sekcja Zarejestrowani agenci pokazuje pierwszy i ostatni kontakt („brak kontaktu", dopóki usługa nie połączyła się z CERTYX). Agent czeka potem na profil sieciowy („No network profile is assigned"), dopóki nie wykonasz kroku 4 sekcji „Pięć kroków". Skanowanie zatrzymasz przez `sudo systemctl stop clm-discovery.service` albo wyłączenie profilu w CERTYX; `sudo systemctl status clm-discovery.service` i `sudo journalctl -u clm-discovery.service` pokazują samą usługę.

7.7 Prywatne CA

Jeśli certyfikat HTTPS CERTYX pochodzi z własnego CA organizacji, serwer musi ufać temu certyfikatowi głównemu przed sprawdzeniem połączenia w kroku 4. Gdy operator publikuje certyfikat główny przez `Clm:Ingress:AgentTrustRootFile`, nic nie wpisuje się ręcznie: krok zaufania w konsoli pobiera go i sprawdza sumę SHA-256, a instalator dodaje go przez `CLM_AGENT_TRUST_ROOT` do systemowego magazynu zaufanych certyfikatów (Debian i Ubuntu, systemy zgodne z RHEL i SUSE) i usuwa go ponownie po nieudanym sprawdzeniu; `uninstall-linux.sh` usuwa go razem z agentem. Bez opublikowanego certyfikatu głównego pobierz go od zespołu PKI i przekaż plik w ten sam sposób: `sudo CLM_AGENT_PUBLISHER_SHA256=<PUBLISHER-SHA256> CLM_AGENT_TRUST_ROOT=<ROOT-CA-FILE> bash install-linux.sh …` (do pobrania pakietu użyj `curl --cacert <ROOT-CA-FILE>`). Instalator odrzuca plik, który nie jest certyfikatem CA albo wygasa w ciągu doby, i zatrzymuje się z komunikatem w dystrybucji bez `update-ca-certificates` ani `update-ca-trust`. Agent sprawdza też online status unieważnienia certyfikatu CERTYX, dlatego ten certyfikat musi wskazywać punkt dystrybucji CRL (lub responder OCSP) osiągalny z serwera. Publiczne CA zapewniają oba.

7.8 Diagnoza w konsoli



Stan agentów: każdy agent ze stanem, ostatnim kontaktem, zadaniami i przyczyną problemu (dane przykładowe)

Stan agentów pokazuje dla każdego agenta dokładną przyczynę, którą CERTYX wyprowadza z kontaktu, tożsamości i raportów — brak raportów od określonego czasu, brak pierwszego kontaktu, tożsamość wygasła lub wygasa, rozbieżność zegara (mierzona na podstawie znacznika czasu mapy sieci), błąd procesu ze stałą przyczyną zgłoszoną przez agenta, proces, który przekroczył okno raportowania, brak konfiguracji, zmniejszone tempo skanowania — z faktami i działaniami naprawczymi po kolei, każde z odnośnikiem do miejsca wykonania. Każda zmiana przyczyny pozostaje w dzienniku stanu agenta i w dzienniku bezpieczeństwa (`agent.health.changed`). Agent cyklu życia zgłasza też swoje miejsca wdrożenia (ID, nazwy DNS, konektor, tryb instalacji, host zdalny przekaźnika SSH, katalog wdrożenia, pliki, polecenia sprawdzenia i przeładowania, zasób chmurowy, endpoint; bez sekretów, tożsamości SSH i zawartości known_hosts), które gotowość odnowienia certyfikatu porównuje z celem wykonania: ID celu musi być równe ID miejsca wdrożenia o tych samych nazwach, przeznaczeniu, trybie instalacji i hoście zdalnym, w przeciwnym razie agent odrzuci zadanie.

7.9 Rozwiązywanie problemów w systemie Linux

Instalator, `Clm.Agent check <adres>` i `sudo clm-agent status` wypisują jeden wiersz na krok; usuń przyczynę pierwszego wiersza `[FAIL]` i uruchom polecenie ponownie. Komunikaty (wypisywane po angielsku):

- `[FAIL] DNS: <nazwa> cannot be resolved from this host (HostNotFound)`: nazwa jest błędna albo serwer DNS tego hosta jej nie zna. Sprawdź adres z konsoli i `/etc/resolv.conf`; do testu dopisz nazwę do `/etc/hosts`.
- `[FAIL] TCP: cannot connect to <host> port <port> (… ConnectionRefused)`: nic tam nie nasłuchuje. Sprawdź port w adresie dla agentów; na lokalnym serwerze testowym uruchom CERTYX z `CLM_AGENT_HOST` i zezwól na przychodzące TCP 5443.
- `[FAIL] TCP: … (… TimedOut)`: zapora odrzuca połączenie albo brak trasy. Zezwól na wychodzące TCP do portu (i przychodzące na serwerze).
- `[FAIL] TLS: no TLS handshake with …`: port nie obsługuje HTTPS, na przykład port konsoli HTTP 5180. Użyj adresu HTTPS dla agentów.
- `[FAIL] TLS: this host does not trust the issuer of the server certificate (…)`: dodaj prywatny lub lokalny certyfikat główny (krok zaufania w konsoli albo „Prywatne CA") i uruchom polecenie ponownie.
- `[FAIL] TLS: revocation of the server certificate cannot be checked: this host cannot download the revocation list http://…`: zezwól na wychodzący HTTP do podanego adresu (na lokalnym serwerze testowym: przychodzące TCP 5181 i działający `./start.sh`). `… the certificate names no CRL distribution point` oznacza, że operator musi użyć certyfikatu z CA publikującego CRL.
- `[FAIL] TLS: a certificate of the chain is outside its validity period (…; this host's time is …)`: sprawdź zegar tego hosta; jeśli jest poprawny, operator musi odnowić certyfikat serwera.
- `[FAIL] TLS: the server certificate does not name <host>`: użyj dokładnie adresu z konsoli; certyfikat musi zawierać tę nazwę lub adres IP.
- `[FAIL] Clock: this host's clock differs from the server by <n> minutes`: rejestracja i uwierzytelnianie nie działają przy różnicy większej niż kilka minut. Uruchom `sudo timedatectl set-ntp true` (albo skonfiguruj NTP) i spróbuj ponownie.
- `[FAIL] Agent address: <adres> does not serve agents (HTTP 404)`: użyto adresu konsoli, a instalacja ma osobny adres dla agentów; użyj „Adresu dla agentów".
- `[FAIL] Registration: CERTYX does not accept this agent identity (HTTP 401)`: agent został unieważniony, jego certyfikat wygasł albo należy do innej organizacji lub instalacji. Zastąp go jak przy „already has an enrolled CERTYX agent".

- `[FAIL] Service: clm-discovery.service is inactive`: uruchom usługę poleceniem `sudo systemctl start clm-discovery.service` i przeczytaj `sudo clm-agent log`.
- `[FAIL] Configuration: cannot read the agent configuration`: uruchom polecenie przez sudo; jeśli pliku nie ma, agent nie jest zainstalowany.

Komunikaty pobierania:

- `curl: (22) The requested URL returned error: 401` dla `clm-agent-linux-x64.tar.gz`: pobieranie archiwum nie zawierało ważnego kodu rejestracji. Uruchom polecenie w katalogu roboczym z plikiem clm-enrollment.token; jeśli plik tam jest, kod wygasł lub został użyty albo identyfikator organizacji jest błędny. Utwórz nowy kod, skopiuj nowy plik i uruchom polecenia ponownie (zob. „Pobieranie pakietów").

Komunikaty instalatora:

- `Package verification failed; nothing was installed.`: pakiet nie jest podpisany, został zmieniony po podpisaniu albo podpisał go inny wydawca niż wskazany w `CLM_AGENT_PUBLISHER_SHA256`. Sprawdź odcisk (uzyskany od dostawcy CLM niezależnie od pobrania), rozpakuj pakiet ponownie ze zweryfikowanego archiwum i nie dodawaj ani nie zmieniaj w nim plików.
- `Package verifier missing; nothing was installed.`: w rozpakowanym pakiecie brakuje `verify-agent-package.sh` albo `CLM_AGENT_VERIFIER` nie wskazuje zwykłego pliku. Rozpakuj ponownie zweryfikowane archiwum albo wskaż w `CLM_AGENT_VERIFIER` osobno pobrany weryfikator.
- `Enrollment token file not found: <ścieżka>`: trzeci argument nie wskazuje pliku kodu na tym serwerze. Pobierz go (krok 1), skopiuj do katalogu roboczego (krok 2), uruchom instalator w katalogu clm-agent pod nim i podaj `../clm-enrollment.token`. Komunikat pokazuje też bieżący katalog.
- `Argument 1 is still a placeholder`, `the example address from the guide` albo `not a valid control-plane origin`: użyj „Adresu dla agentów" z konsoli: `https://`, nazwa hosta i opcjonalnie port, bez ścieżki.
- `The enrollment code was rejected` (HTTP 400, 401, 403 lub 409): kod wygasł (jest ważny przez okres ustawiony w `Clm:Agents:Enrollment:TicketLifetime`, domyślnie dziesięć minut; konsola pokazuje termin), został już użyty albo identyfikator organizacji się nie zgadza. Utwórz nowy kod dla tego samego identyfikatora agenta, skopiuj nowy plik i uruchom to samo polecenie ponownie; lokalny klucz z nieudanej próby zostanie użyty ponownie.
- `does not serve agents` (HTTP 404): użyto adresu konsoli, a instalacja ma osobny adres dla agentów. Użyj „Adresu dla agentów" pokazanego przez konsolę.
- `cannot be reached` albo `cannot be resolved`: sprawdź adres, port, DNS i regułę zapory dla wychodzącego HTTPS. `curl -s -o /dev/null -w '%{http_code}\n' <CONTROL-PLANE-ORIGIN>/health/live` musi wypisać 200.
- `does not trust the HTTPS certificate`: zob. „Prywatne CA".
- `already has an enrolled CERTYX agent`: ten serwer jest już zarejestrowany i nic nie zostało zmienione; sprawdź go poleceniem `sudo clm-agent status`. Aby go zastąpić, usuń go poleceniem `sudo bash /opt/clm-agent/current/uninstall-linux.sh` (zob. „Usuwanie w systemie Linux"), unieważnij agenta w CERTYX, a potem utwórz kod dla nowego identyfikatora agenta (identyfikator można zarejestrować tylko raz) i zainstaluj ponownie.
- `clm-discovery.service is not running`: instalator wypisuje ostatnie wiersze logu. Więcej odczytasz poleceniem `sudo journalctl -u clm-discovery.service -n 50 --no-pager`; błędy są logowane według kategorii (typ wyjątku, status HTTP), nigdy z kodami ani kluczami.
- `This host does not run systemd`: użyj „Uruchomienia ręcznego lub w kontenerze".

7.10 Usuwanie w systemie Linux

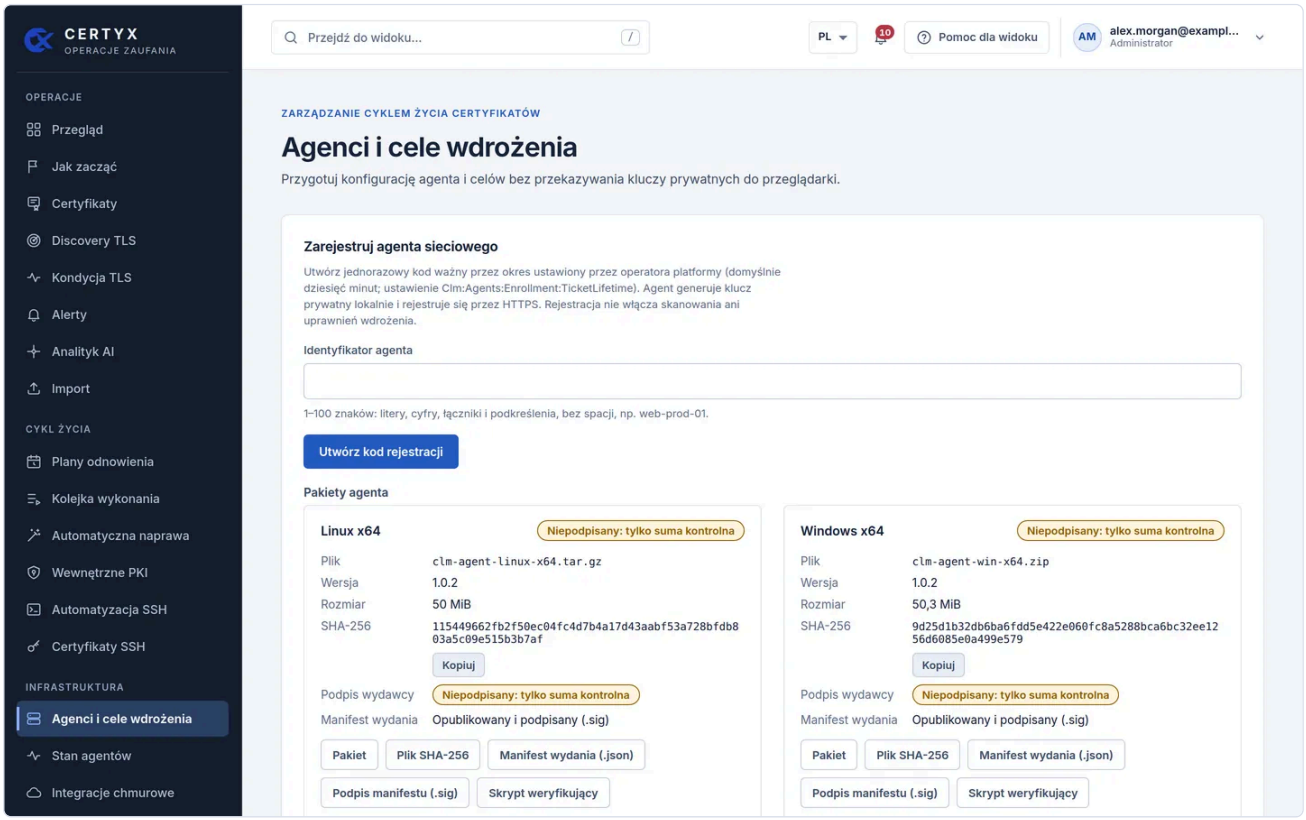
```
sudo bash /opt/clm-agent/current/uninstall-linux.sh
```

Skrypt kontynuuje z kopii tymczasowej (usuwa własny katalog), zatrzymuje, wyłącza i usuwa każdą jednostkę zainstalowaną przez skrypty CERTYX (`clm-discovery.service`, `clm-ssh.service` z jego plikiem drop-in wdrożenia na hoście, `clm-ssh-renewal.service` i `.timer`, `clm-endpoint-verifier.service`, `clm-agent-update.service` i `.timer`), usuwa blok CERTYX dodany przez `trust-user-ca` do /etc/ssh/sshd_config (sprawdzany przez `sshd -t`; kopia zapasowa jest zachowywana i przywracana, gdy sshd odrzuci wynik; `--keep-ssh-trust` pozostawia sshd bez zmian), regułę sudoers wdrożenia SSH na hoście i dodany przez instalator wpis zaufania `certyx-agent-root.crt` (`--keep-trust-root` go zachowuje) oraz usuwa /opt/clm-agent, /etc/clm-agent, /var/lib/clm-agent, /etc/clm-agent-update, /etc/clm-ssh, /etc/clm-verifier, /usr/local/bin/clm-agent i konto clm-agent. Klucze prywatne są usuwane razem z tymi katalogami. Ponowne uruchomienie na czystym hoście niczego nie zmienia. Następnie unieważnij agenta w CERTYX → Agenci i miejsca docelowe (Zarejestrowani agenci → Unieważnij agenta): dopiero wtedy CERTYX przestaje akceptować jego tożsamość.

8 Windows / Harmonogram zadań

Tak jak w Linux konsola pokazuje po „Utwórz kod rejestracji" te polecenia z prawdziwymi wartościami. Symbole `<CONTROL-PLANE-ORIGIN>`, `<ORGANIZATION-ID>` i `<PUBLISHER-SHA256>` znaczą to samo co w sekcji Linux; zastąp je razem z nawiasami ostrymi.

8.1 Krok 1: utwórz kod rejestracji



Agenci i cele wdrożenia: rejestracja, zarejestrowani agenci, profile sieci i szkice miejsc docelowych (dane przykładowe)

Tak jak w kroku 1 dla Linux: utwórz kod w CERTYX i pobierz clm-enrollment.token. Jest ważny tylko raz i przez skonfigurowany okres (`Clm:Agents:Enrollment:TicketLifetime`, domyślnie dziesięć minut; konsola pokazuje termin); wykonaj kroki 2–4 w tym czasie.

8.2 Krok 2: skopiuj plik kodu do folderu roboczego

Utwórz pusty folder roboczy na serwerze Windows i skopiuj do niego clm-enrollment.token. Zachowaj nazwę pliku. Polecenia kroków 3 i 4 uruchamiaj tam w PowerShell jako administrator.

8.3 Krok 3: pobierz, sprawdź i rozpakuj pakiet

Jeśli konsola pokazuje krok zaufania dla własnego urzędu certyfikacji, wykonaj go najpierw w PowerShell jako administrator; drugie polecenie musi wypisać `True`:

```
curl.exe -fsSLk -o certyx-agent-root.crt <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-trust-root.crt
(Get-FileHash certyx-agent-root.crt -Algorithm SHA256).Hash -eq '<SHA256-FROM-CONSOLE>'
Import-Certificate -FilePath .\certyx-agent-root.crt -CertStoreLocation Cert:\LocalMachine\Root
```

Następnie w PowerShell uruchomionym jako administrator na serwerze Windows (x64), w folderze roboczym z plikiem clm-enrollment.token. Pierwsze polecenie odczytuje kod z tego pliku do nagłówków `$clmDownload`, które wysyła pobieranie archiwum (zob. „Pobieranie pakietów"):

```
$clmDownload = @{ 'X-CLM-Enrollment-Token' = (Get-Content .\clm-enrollment.token -Raw).Trim(); 'X-CLM-Tenant' = '<ORGANIZATION-ID>' }
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-win-x64.zip -Headers $clmDownload -OutFile clm-agent-win-x64.zip -UseBasicParsing
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-win-x64.zip.sha256 -OutFile clm-agent-win-x64.zip.sha256 -UseBasicParsing
(Get-FileHash clm-agent-win-x64.zip -Algorithm SHA256).Hash -eq (Get-Content clm-agent-win-x64.zip.sha256).Split(' ')[0]
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-win-x64.zip.sha256.p7s -OutFile clm-agent-win-x64.zip.sha256.p7s -UseBasicParsing
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/verify-agent-package.ps1 -OutFile verify-agent-package.ps1 -UseBasicParsing
Get-AuthenticodeSignature .\verify-agent-package.ps1
.\verify-agent-package.ps1 -PublisherSha256 <PUBLISHER-SHA256> -Archive .\clm-agent-win-x64.zip
Expand-Archive clm-agent-win-x64.zip -DestinationPath clm-agent
cd clm-agent
```

Porównanie z `Get-FileHash` musi wypisać `True`, a `verify-agent-package.ps1` musi wypisać `VERIFIED`; w przeciwnym razie nie instaluj (zob. „Weryfikacja pakietu"). `(401) Unauthorized` przy pobieraniu archiwum oznacza, że brakuje kodu, wygasł lub został użyty albo identyfikator organizacji jest błędny (zob. „Pobieranie pakietów"). Podpisany manifest możesz dodatkowo sprawdzić w PowerShell 7 lub OpenSSL, jak opisano w „Podpisanych aktualizacjach".

8.4 Krok 4: uruchom instalator i sprawdź zadanie

W folderze clm-agent utworzonym przez `Expand-Archive` (plik kodu pozostaje w folderze roboczym nad nim, stąd `..\clm-enrollment.token`):

```
powershell -NoProfile -ExecutionPolicy Bypass -File .\install-windows.ps1 -ControlPlane <CONTROL-PLANE-ORIGIN> -Tenant <ORGANIZATION-ID> -TokenFile
..\clm-enrollment.token -PublisherSha256 <PUBLISHER-SHA256>
Get-ScheduledTask -TaskName 'CLM Discovery Agent' | Get-ScheduledTaskInfo
```

`-ExecutionPolicy Bypass` dotyczy tylko tego jednego procesu instalatora. Instalator najpierw sprawdza argumenty i wskazuje błędny argument razem z kolejnym krokiem (te same komunikaty co w Linux), potem uruchamia `Clm.Agent.exe check` dla adresu bez zużywania kodu. Na końcu czeka na pierwszy kontakt i wypisuje sprawdzenie stanu. Uruchamiaj je w dowolnym momencie w PowerShell jako administrator; każdy wiersz musi zaczynać się od `[ OK ]`, a ostatni brzmieć `Result: connected`:

```
& "$env:ProgramData\CLM-Agent\Clm.Agent.exe" status "$env:ProgramData\CLM-Agent\agent.json"
```

Instalator ogranicza ACL ProgramData\CLM-Agent, ustawia grupę Administratorzy jako właściciela, kopiuje kod do środka na czas rejestracji i potem usuwa tę kopię, a następnie rejestruje zadanie harmonogramu `CLM Discovery Agent`, które startuje przy uruchomieniu systemu jako LOCAL SERVICE, wykonuje `C:\ProgramData\CLM-Agent\Clm.Agent.exe scan C:\ProgramData\CLM-Agent\agent.json` bez limitu czasu i po błędzie jest uruchamiane ponownie do trzech razy co minutę. Agent i konfiguracja (`agent.json`) znajdują się w `C:\ProgramData\CLM-Agent`, tożsamość (`identity.pfx`, `identity.password`) w `C:\ProgramData\CLM-Agent\identity`, a stan w `state\`; LOCAL SERVICE może zapisywać tylko te dwa podkatalogi: rotacja tożsamości zapisuje następną tożsamość (`identity.pfx.pending`) w katalogu tożsamości i przenosi ją na miejsce bieżącej, a poza nimi konto zadania nie może niczego zmienić w katalogu programu. Instalator odrzuca niepusty katalog ProgramData\CLM-Agent, kopiuje pakiet do chronionego katalogu i weryfikuje tę chronioną kopię (podpisany manifest, przypiętego wydawcę i podpisy Authenticode) przed rejestracją; odrzucony pakiet jest usuwany i nic nie zostaje zarejestrowane. `-PublisherSha256` jest wymagany; `-PublisherCaFile` dodatkowo wymaga łańcucha CA, `-VerifierPath` używa osobno pobranego weryfikatora, a `-VerifyOnly` tylko sprawdza rozpakowany pakiet. `-AllowUnsignedDevelopment` to wyłącznie deweloperskie obejście, które wypisuje ostrzeżenie. Nie jest to usługa Windows. Następnie usuń clm-enrollment.token z folderu roboczego. Nowsze wersje instaluje się w miejscu skryptem `upgrade-windows.ps1` (zob. „Aktualizacja w miejscu").

Automatyczne podpisane aktualizacje (opcjonalnie): gdy operator publikuje klucz podpisywania wydań, dodaj do polecenia instalacji `-ReleaseKeyFile .\release-signing.pub` (otrzymany wraz z odciskiem zaufanym kanałem). Po rejestracji instalator uruchamia zainstalowany `upgrade-windows.ps1 -EnableUpdates`, który przypina klucz i odcisk wydawcy w `C:\ProgramData\CLM-Agent-Update` (tylko Administratorzy i LocalSystem) i rejestruje zaplanowane zadanie `CERTYX Agent Update`: codziennie o 03:00 z losowym opóźnieniem do sześciu godzin, jako LocalSystem, uruchamia `upgrade-windows.ps1 -Scheduled`, czyli tę samą podpisaną aktualizację `-Download` co w „Aktualizacji w miejscu" (z wycofaniem); wydanie, które nie jest nowsze, kończy się kodem 0, a wynik jest dopisywany do `C:\ProgramData\CLM-Agent-Update\update.log`. Sprawdzenie: `Get-ScheduledTask -TaskName 'CERTYX Agent Update' | Get-ScheduledTaskInfo`. Instalacja bez klucza otrzymuje zadanie przy pierwszej podpisanej aktualizacji (`-ReleaseKeyFile`) albo poleceniem `upgrade-windows.ps1 -EnableUpdates -PublisherSha256 <PUBLISHER-SHA256> -ReleaseKeyFile .\release-signing.pub`.

Zadanie nie ma okna konsoli: agent zapisuje swoje komunikaty w `C:\ProgramData\CLM-Agent\state\discovery.log` (wiersze ze znacznikiem czasu UTC, najwyżej 5 MB; poprzednia część zostaje jako `discovery.log.1`). Odczytasz go w PowerShell administratora poleceniem `Get-Content -Tail 50 C:\ProgramData\CLM-Agent\state\discovery.log` — to odpowiednik `sudo clm-agent log` w Windows. Zadanie obciążenia SSH zapisuje `C:\ProgramData\CLM-SSH\state\ssh-workload.log`, którego ostatnie wiersze pokazuje `configure-ssh-windows.ps1 status`. Log zawiera wyłącznie bezpieczne podsumowania agenta, nigdy tokeny, hasła ani klucze.

Jeśli CERTYX używa prywatnego CA bez opublikowanego certyfikatu głównego, najpierw zaimportuj certyfikat główny od zespołu PKI: `Import-Certificate -FilePath <ROOT-CA-FILE> -CertStoreLocation Cert:\LocalMachine\Root`, gdzie `<ROOT-CA-FILE>` to ścieżka certyfikatu głównego przekazanego przez zespół PKI. Gdy konsola pokazuje krok zaufania, użyj jego poleceń. Wymaganie punktu dystrybucji CRL z „Prywatne CA" obowiązuje także tutaj.

Rozwiązywanie problemów: komunikaty i sposoby postępowania są takie jak dla Linux. Aby zastąpić instalację, usuń ją skryptem `uninstall-windows.ps1` (zob. „Usuwanie"), unieważnij agenta w CERTYX i zainstaluj ponownie z kodem dla nowego identyfikatora agenta.

8.5 Usuwanie

W PowerShell administratora:

```
powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\uninstall-windows.ps1"
```

Skrypt zatrzymuje i usuwa zaplanowane zadania `CLM Discovery Agent`, `CERTYX SSH Workload`, `CERTYX SSH Renewal`, `CERTYX Endpoint Verifier` i `CERTYX Agent Update`, usuwa blok CERTYX dodany przez `trust-user-ca` do `C:\ProgramData\ssh\sshd_config` razem z `certyx_user_ca.pub` (sprawdzane przez `sshd -t`; kopia zapasowa zostaje i jest przywracana, gdy sshd nie przyjmie wyniku; `-KeepSshTrust` pozostawia sshd bez zmian) oraz usuwa `C:\ProgramData\CLM-Agent` (program, konfiguracja, tożsamość, weryfikator endpointów, stan i logi), `C:\ProgramData\CLM-SSH` (CA SSH, klucze i stan obciążenia) i `C:\ProgramData\CLM-Agent-Update` (przypięty klucz wydań, odcisk wydawcy i dziennik aktualizacji). Klucze prywatne są usuwane razem z tymi folderami. Następnie unieważnij agenta w CERTYX → Agenci i cele wdrożenia (Zarejestrowani agenci → Unieważnij agenta): dopiero wtedy CERTYX przestaje akceptować jego tożsamość. Identyfikator agenta można zarejestrować tylko raz; nowa instalacja wymaga kodu dla nowego identyfikatora.

9 Uruchomienie ręczne lub w kontenerze

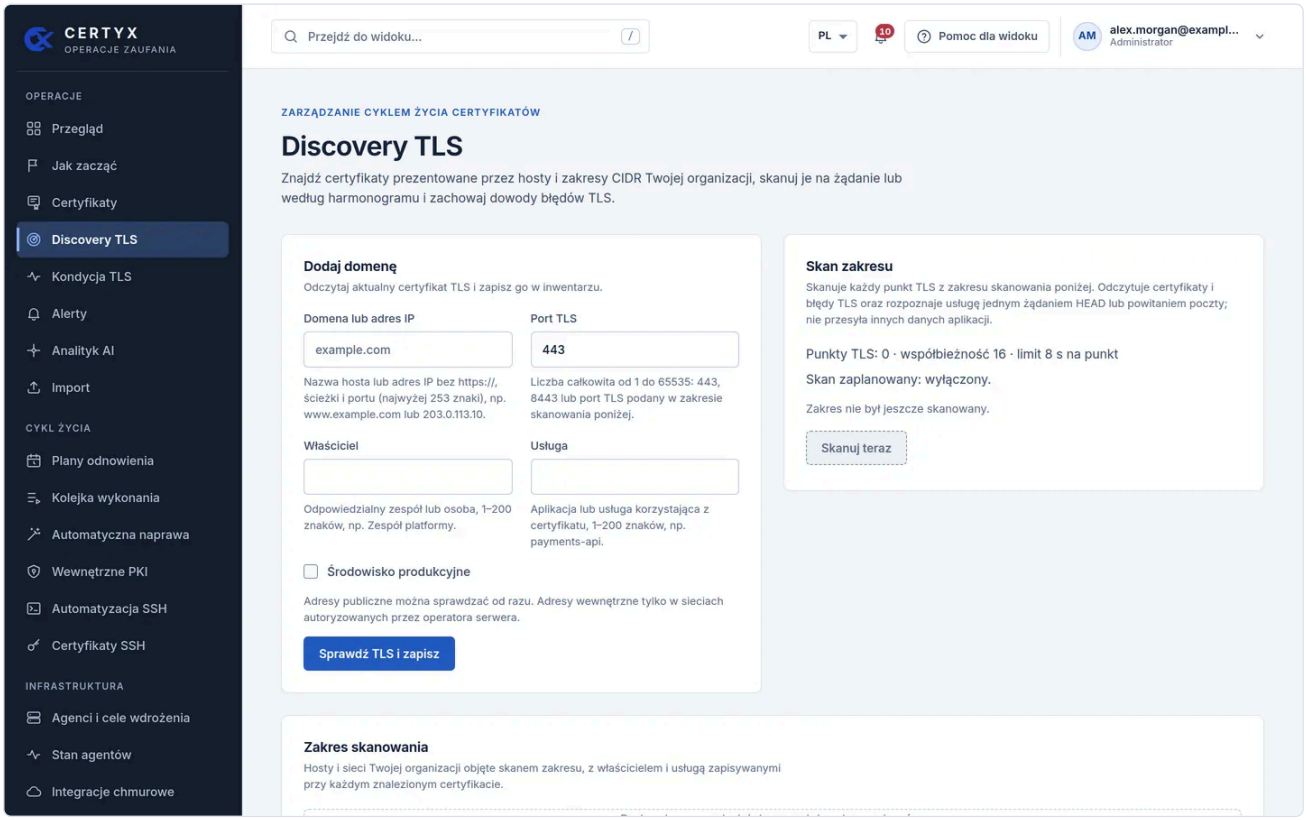
Ten sam plik wykonywalny działa bez instalatora systemowego:

```
Clm.Agent enroll <CONTROL-PLANE-ORIGIN> <ORGANIZATION-ID> <ENROLLMENT-TOKEN-FILE> <CONFIG-FILE>
Clm.Agent scan <CONFIG-FILE>
Clm.Agent scan <CONFIG-FILE> --once
```

`<CONTROL-PLANE-ORIGIN>` i `<ORGANIZATION-ID>` to wartości pokazywane przez konsolę po „Utwórz kod rejestracji"; `<ENROLLMENT-TOKEN-FILE>` to pobrany plik clm-enrollment.token (dostępny tylko dla właściciela, tryb 600); `<CONFIG-FILE>` to ścieżka, pod którą rejestracja zapisze konfigurację agenta, na przykład /var/lib/clm-agent/agent.json; w jej katalogu znajdą się też pliki tożsamości.

Chroń konfigurację i tożsamość, utrwal katalog stanu i zapewnij dostęp sieciowy do celów. Sieć mostkowa kontenera nie oznacza automatycznie dostępu do infrastruktury klienta: mapa sieci kontenera pokazuje jego sieć mostkową, dlatego jawnie zatwierdź osiągalne zakresy, a gdy host kontenera ma wymuszać własną granicę, ustaw `RestrictToAllowedCidrs` i `AllowedCidrs`. --once wykonuje jedną ograniczoną partię (najwyżej liczbę prób na cykl z profilu), nie całą sieć.

10 Zakres i postęp



Discovery TLS: zakres, postęp oraz punkty końcowe i certyfikaty znalezione w sieci prywatnej (dane przykładowe)

Każde przejście ma do trzech faz, zawsze w tej udokumentowanej kolejności:

1. **Aktywność (liveness):** każdy adres zatwierdzonych zakresów (bez wykluczeń oraz, do /30, bez adresu sieci i rozgłoszeniowego) w kolejności rosnącej. Agent próbuje kolejno portów sprawdzania aktywności; adres jest aktywny, gdy port przyjmie połączenie albo aktywnie je odrzuci (TCP reset). Adresy z tablicy sąsiadów ARP hosta uznaje się za aktywne bez nawiązywania połączenia.
2. **Typowe porty (common):** każdy aktywny host w kolejności rosnącej, jego typowe porty w kolejności z profilu, z połączeniem TCP i bezpośrednim handshake TLS, po którym następuje rozpoznanie usługi opisane w części „Dla zespołu bezpieczeństwa klienta".
3. **Pełna (full,** tylko tryb `full`): każdy pozostały port 1–65535 rosnąco, każdy port na wszystkich aktywnych hostach przed następnym portem, więc kolejne połączenia trafiają do różnych hostów.

Nie ma losowej kolejności ani losowych opóźnień (jitter). Agent nigdy nie otwiera do jednego hosta więcej połączeń niż limit z profilu (1–4) i rozpoczyna połączenia w stałym tempie najwyżej `ratePerSecond` (1–2000); lokalne `ProbeSpacingMilliseconds` może je tylko spowolnić. Poza oknami skanowania (okna tygodniowe w strefie czasowej profilu; koniec przed początkiem oznacza kontynuację następnego dnia, `24:00` kończy okno o północy, okna przy zmianie czasu letniego i zimowego trzymają się lokalnego zegara) agent nie nawiązuje połączeń i zgłasza stan `waiting-window` z początkiem następnego okna. Gdy w segmencie fazy typowych portów lub pełnej ponad `BackOffFailurePercent` (domyślnie 50%) prób połączenia kończy się przekroczeniem czasu, tempo następnego segmentu jest zmniejszane o połowę (nigdy poniżej `BackOffMinimumPercent`, domyślnie 10%,

zatwierdzonego tempa) i po każdym spokojniejszym segmencie podnoszone o `BackOffRestorePercent` (domyślnie 25%) zatwierdzonego tempa; telemetria pokazuje `backOff`. Telemetria zgłasza też fazę, pozycję w fazie, aktywne hosty i `baselineCompletedAt`, czyli czas zakończenia pierwszego pełnego przejścia bieżącego zakresu.

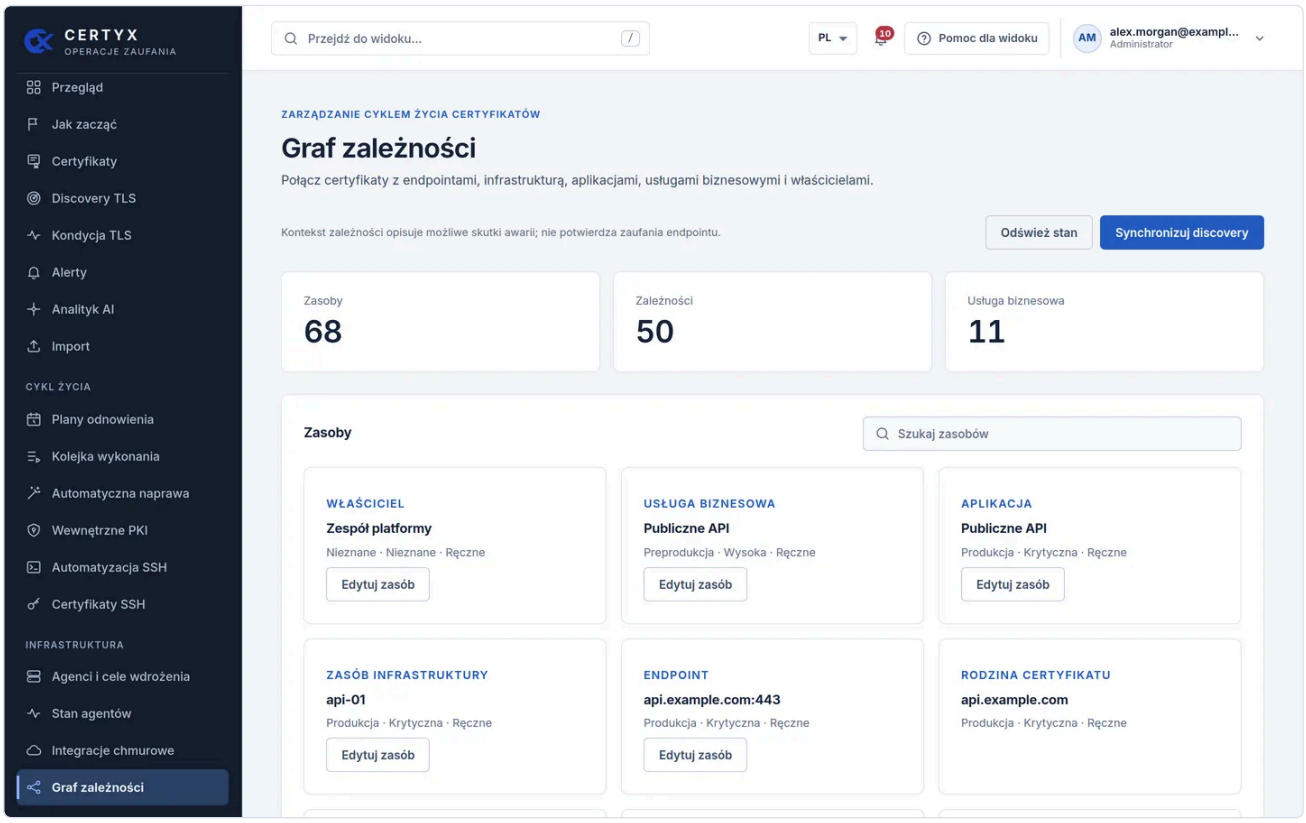
Postęp jest zapisywany po każdym segmencie (`SegmentSize`, domyślnie 200 kroków) w punkcie kontrolnym dostępnym tylko dla właściciela: faza, pozycja i aktywne hosty. Punkt kontrolny jest powiązany wyłącznie ze skrótem zakresu i portów, więc zmiana tempa, liczby połączeń na host, okien lub wartości partii kontynuuje przejście, a zmiana zakresu, trybu lub listy portów zaczyna je od nowa. Profil jest ponownie sprawdzany między segmentami; partia kończy się po liczbie prób na cykl z profilu, a następna zaczyna się po interwale z profilu. Obserwacja odrzucona przez CLM kodem HTTP 400 (na przykład dlatego, że zakres został w międzyczasie zmniejszony) jest pomijana, zgłaszana w telemetrii, a skanowanie trwa dalej. Niepoprawny profil jest odrzucany przy walidacji i zgłaszany jako nieprawidłowy; agent działa dalej i ponawia próbę po `RetrySeconds`.

Realistyczny czas to liczba kroków podzielona przez efektywne tempo, czyli mniejszą z wartości: `ratePerSecond` oraz `Concurrency` podzielone przez limit czasu połączenia (gdy większość prób kończy się przekroczeniem czasu). Partie (`attemptsPerCycle`) jedynie ograniczają pracę między odświeżeniem profilu a zapisem punktu wznowienia i następują bezpośrednio po sobie; `intervalSeconds` to przerwa po ukończeniu całego przejścia:

- Aktywność dla /24 z 4 portami sprawdzania: najwyżej około 1000 prób, około 10 sekund przy 100 na sekundę. Słabo zajęta sieć /16: około 262 000 prób, około 45 minut przy 100 na sekundę albo 9 minut przy 500 na sekundę. Nieistniejące adresy kończą się przekroczeniem czasu, więc 32 równoległe połączenia przy limicie połączenia 500 ms pozwalają na około 64 próby na sekundę (około 70 minut).
- Typowe porty: aktywne hosty × typowe porty, na przykład 300 hostów × 20 portów = 6000 prób, około minuty przy 100 na sekundę.
- Tryb `full`: aktywne hosty × 65 535 portów, na przykład 50 hostów = około 3,3 miliona prób, około 4,6 godziny przy 200 na sekundę albo 55 minut przy 1000 na sekundę. Firewalle odrzucające pakiety zamieniają większość prób w przekroczenia czasu i zwalnianie, co może wydłużyć skan do doby lub dłużej. Przy 10 000 prób na cykl, 200 na sekundę i interwale 300 sekund średnia wynosi około 29 prób na sekundę, więc to samo przejście trwa około 32 godzin; skróć interwał, jeśli pozwalają na to okna i zespół bezpieczeństwa.

Brak wykrytego certyfikatu nie dowodzi jego braku. Ten skaner obsługuje bezpośredni TLS na IPv4 TCP. UDP/DTLS/QUIC, STARTTLS SMTP/LDAP, przemiatanie IPv6, enumeracja wirtualnych hostów SNI, magazyny plikowe certyfikatów i konektory chmurowe nie są jego częścią.

11 Mapa sieci



Graf zależności: hosty, usługi i certyfikaty wraz z zależnościami (dane przykładowe)

Przy starcie, a potem co `NetworkMapIntervalMinutes` (domyślnie 60) agent wysyła do CLM prywatne sieci IPv4 swojego hosta: bezpośrednio podłączone sieci aktywnych interfejsów i cele tablicy routingu (Linux `/proc/net/route`, Windows tablica routingu IP Helper), a także prywatne adresy z tablicy sąsiadów ARP (Linux `/proc/net/arp`, Windows tablica sąsiadów IP Helper) jako wskazówki aktywności. Zgłaszane są tylko zakresy RFC 1918 i 100.64.0.0/10; trasy domyślne, pętla zwrotna, link-local, multicast, cele publiczne i wszystko szersze niż /8 są pomijane, a wysyłanych jest najwyżej 1024 sieci i 4096 sąsiadów. Odczyt tych tablic nie wysyła żadnych pakietów. Mapa jest tylko podpowiedzią dla administratora: nic nie jest skanowane, dopóki zakres nie zostanie zatwierdzony w profilu. Gdy CLM odrzuci mapę (na przykład starszy serwer bez tego punktu końcowego), agent zapisuje w logu kod HTTP i ponawia próbę w następnym interwale; skanowanie działa dalej. Tablice sąsiadów IPv6 nie są odczytywane; w innych systemach operacyjnych zgłaszane są tylko sieci interfejsów.

12 Dla zespołu bezpieczeństwa klienta

Wykrywanie ma być widoczne i rozpoznawalne, a nie ukryte. Przed włączeniem profilu uzgodnij z zespołem bezpieczeństwa:

- **Adresy źródłowe:** adresy IP hostów, na których działa agent wykrywający. Każdy agent łączy się tylko ze swojego hosta; samo CLM nigdy nie łączy się do Twojej sieci.
- **Cele i porty:** tylko zatwierdzone zakresy bez wykluczeń; porty sprawdzania aktywności, potem typowe porty aktywnych hostów, a w trybie `full` każdy port TCP 1–65535 aktywnych hostów. Są to zwykłe połączenia TCP (pełny handshake, bez surowych pakietów i bez podszywania się pod inne adresy) z następującym handshake TLS; połączenie jest zamykane po odczytaniu certyfikatu i rozpoznaniu usługi.
- **Rozpoznawanie usługi:** handshake TLS oferuje protokoły ALPN `h2` i `http/1.1` (nie na portach poczty, LDAP, AMQP, MQTT i DNS przez TLS) i zapisuje wersję TLS, zestaw szyfrów, protokół ALPN oraz certyfikaty wystawców przedstawione przez punkt. Następnie do punktu, który wynegocjował protokół HTTP albo żaden na porcie HTTP (443, 4443, 8443, 9443, 10443, 9000), agent wysyła dokładnie jedno żądanie — `HEAD / HTTP/1.1` z `Host` równym adresowi, `User-Agent: CERTYX-Discovery`, `Accept: */*` i `Connection: close` — i odczytuje tylko wiersz statusu i nagłówki (najwyżej 16 KiB, 2 sekundy); nigdy nie podąża za przekierowaniem, nie czyta treści i nie wysyła `GET`, również po `405`. Punkt, który wynegocjował HTTP/2, dostaje to żądanie przez jeszcze jedno połączenie oferujące tylko HTTP/1.1. Na SMTPS 465, IMAPS 993 i POP3S 995 agent niczego nie wysyła i odczytuje pierwszy wiersz powitania. Pozostałe porty inne niż HTTP (25, 110, 143, 587, 636, 853, 2376, 3269, 5671, 5986, 6443, 8883) nie dostają żadnego żądania. Dzienniki serwerów WWW i IDS pokazują te żądania z agentem użytkownika `CERTYX-Discovery`. `"Settings": { "ServiceIdentification": { "IdentifyServices": false } }` w konfiguracji skanu wyłącza żądanie i odczyt powitania; operator serwera robi to samo dla skanów po stronie serwera ustawieniem `Clm:Discovery:ServiceIdentification:IdentifyServices`.
- **Kolejność i czas:** rosnące adresy w stałej kolejności, bez losowej kolejności i bez jittera; tylko w zatwierdzonych oknach w strefie czasowej profilu; najwyżej `ratePerSecond` nowych połączeń na sekundę na agenta i najwyżej 1–4 jednoczesne połączenia na host docelowy; automatyczne zwolnienie, gdy wiele połączeń kończy się przekroczeniem czasu.
- **Rejestracja agenta jako zatwierdzonego skanera:** dodaj adresy hostów agentów do wyjątków dla skanerów lub narzędzi oceny podatności w systemach IDS/IPS, firewallach, NAC i EDR, ograniczonych do zatwierdzonych zakresów i okien, aby alarmy były klasyfikowane jako oczekiwana aktywność, a nie wyciszane globalnie. Bez takiej rejestracji zwłaszcza tryb `full` wygląda jak skanowanie portów i może zostać zablokowany albo zgłoszony jako atak.
- **Zatrzymanie:** wyłączenie profilu w CLM zatrzymuje skanowanie w następnym segmencie; zatrzymanie usługi (`systemctl stop clm-discovery.service` albo zadania harmonogramu) zatrzymuje je natychmiast; `RestrictToAllowedCidrs` pozwala właścicielowi hosta ograniczyć agenta do lokalnie zatwierdzonych sieci niezależnie od profilu.
- **Ślady:** log agenta na hoście pokazuje każdy segment; CLM pokazuje telemetrię agenta (faza, postęp, aktywne hosty, `waiting-window`, zwalnianie).

13 Ustawienia operacyjne

Każdy plik konfiguracji agenta może zawierać opcjonalny obiekt `Settings` na najwyższym poziomie. Pominięte wartości zachowują poniższe wartości domyślne, czyli wartości stosowane przez agenta dotychczas. Wartości są sprawdzane przy starcie polecenia; wartość spoza zakresu zatrzymuje polecenie przed jakimkolwiek połączeniem sieciowym i jest zgłaszana wyłącznie typem wyjątku (`InvalidDataException`). `enroll` nie zapisuje obiektu `Settings`; administrator dodaje go w razie potrzeby do chronionej konfiguracji. Ustawienia nigdy nie zawierają poświadczeń; sekrety pozostają w chronionych plikach i referencjach opisanych w tej instrukcji.

- Konfiguracja `scan`: `RequestTimeoutSeconds` 20 (1–300), `IdentityRenewalDays` 30 (1–179), `ConnectTimeoutMilliseconds` 500 (1–60000), `HandshakeTimeoutSeconds` 3 (1–300), `ProbeSpacingMilliseconds` 0 (0–10000, lokalny minimalny odstęp między rozpoczęciami połączeń; 0 zostawia tempo profilowi, wyższa wartość może agenta tylko spowolnić), `SegmentSize` 200 (1–10000, liczba kroków między sprawdzeniem profilu, przesłaniem wyników, telemetrią i zapisem postępu), `IdleIntervalSeconds` 30 (1–86400, przerwa bez przypisanego profilu i najdłuższa przerwa w oczekiwaniu na okno skanowania), `RetrySeconds` 30 (1–86400, przerwa po nieudanym cyklu), `NetworkMapIntervalMinutes` 60 (1–1440, interwał mapy sieci), `RestrictToAllowedCidrs` false (`true`: każdy zakres profilu musi leżeć w lokalnej liście `AllowedCidrs`), `BackOffFailurePercent` 50 (1–100), `BackOffRestorePercent` 25 (1–100), `BackOffMinimumPercent` 10 (1–100) oraz obiekt `ServiceIdentification` z `IdentifyServices` true (`false` wyłącza żądanie HEAD i odczyt powitania), `TimeoutMilliseconds` 2000 (100–60000), `MaximumResponseHeaderKilobytes` 16 (1–64), listami portów `HttpPorts`, `HttpRequestExcludedPorts`, `AlpnExcludedPorts` i `GreetingPorts` (każda najwyżej 256 portów; wartości domyślne w DISCOVERY-FULL.md, sekcja 8) oraz `UserAgent` `CERTYX-Discovery`. Tryb, porty, tempo, liczba połączeń na host, okna, liczba prób na cykl i odstęp między partiami pochodzą z zatwierdzonego profilu.
- Konfiguracja `ssh manage` i `ssh monitor-krl`: `RequestTimeoutSeconds` 20 i `IdentityRenewalDays` 30, na najwyższym poziomie pliku (nie wewnątrz `Identity`). Ich interwały odpytywania pozostają w `PollSeconds` i `Subscription.PollSeconds`.
- Konfiguracja agenta cyklu życia: `PollSeconds` 10 (1–86400), `RequestTimeoutSeconds` 15 (1–300), `OperationTimeoutSeconds` 600 (1–86400, górny limit jednego zadania; cel w magazynie certyfikatów chmury dodaje trzykrotność swojego okna zbieżności, dla AWS ACM 18 minut), `HeartbeatSeconds` 25 (1–119, odnawianie dzierżawy; musi być krótszy niż 120-sekundowa dzierżawa serwera i limit operacji), `TrustChainTimeoutSeconds` 3 (1–300).
- Limity konektorów, w każdym pliku konfiguracji, także dla lokalnych poleceń `ssh` (`renew`, `publish-krl`, `sync-krl`, `apply-krl` i polecenia urzędu, które czytają `Settings` z pierwszego pliku konfiguracji): `Settings.Connectors` może zawierać `Process`, `Nginx`, `Deployment`, `HostDeployment`, `Acme`, `Dns`, `Vault`, `Secrets` i `CloudVerification` (agent cyklu życia i `secret-check`; `HostDeployment` zawiera limity konektorów plikowych i przekaźnika SSH: `CommandTimeout` 00:01:00 na sprawdzenie, przeładowanie, generowanie klucza lub przebieg SSH, `VerificationTimeout` 00:00:30, `RetryDelay` 00:00:00.5, `RevocationTimeout` 00:00:03, `MaximumReadBytes` 1048576 (4096–16777216) i `ChownExecutable` `/usr/bin/chown`; `CloudVerification` dotyczy celów AWS ACM — jeśli go ustawisz, `ConvergenceWindow` musi wynosić co najmniej `00:06:00`, czyli domyślną wartość AWS przy pominiętym obiekcie) oraz `SshKeygen`, `SshHostService`, `SshHostDeployment` i `SshKrlSubscription` (polecenia SSH).

Czasy podaje się jako tekst `hh:mm:ss`; muszą być dodatnie i nie dłuższe niż godzina; liczby sekund w `SshHostService` mieszczą się w zakresie 1–300. CONFIGURATION.md wymienia wszystkie wartości i domyślne ustawienia w sekcji „Customer-side connector limits". Pominięty obiekt zachowuje wartości domyślne konektora.

Przykład: `"Settings": { "SegmentSize": 100, "RetrySeconds": 60 }` albo `"Settings": { "Connectors": { "Nginx": { "VerificationTimeout": "00:00:30" } } }`. Granice protokołu i bezpieczeństwa są stałe: 5 obserwacji w jednym przesłaniu, limity zakresu wykrywania opisane w kroku 4, limity rozmiaru odpowiedzi, 1 MiB na wpis dziennika i tożsamości RSA-3072 ważne 180 dni.

14 Cofnięcie dostępu i odzyskiwanie

Przycisk Unieważnij agenta blokuje kolejne uwierzytelnione wywołania. Wyłączenie profilu zatrzymuje kolejne partie; trwająca próba może się zakończyć. Tożsamość jest ważna 180 dni; agent wykrywający automatycznie rotuje lokalny klucz i tożsamość 30 dni przed wygaśnięciem (domyślna wartość `IdentityRenewalDays`). Aby wykonać rotację natychmiast, np. przy podejrzeniu ujawnienia klucza, uruchom jako administrator `Clm.Agent rotate-identity <config-file>` (Linux: `sudo /opt/clm-agent/current/Clm.Agent rotate-identity /etc/clm-agent/agent.json`; Windows: `& "$env:ProgramData\CLM-Agent\Clm.Agent.exe" rotate-identity "$env:ProgramData\CLM-Agent\agent.json"`); działający agent wykona rotację przy najbliższym sprawdzeniu tożsamości, a `status` pokaże wtedy nowy odcisk. Zachowaj oczekujące pliki podczas odzyskiwania. 24-godzinny okres przejściowy obsługuje utracone potwierdzenia; zobacz AGENT-IDENTITY-ROTATION.md. Dotychczasowy agent cyklu życia działa osobno i wymaga konfiguracji CA oraz wdrożenia. Rejestracja sieciowa nie aktywuje jego celu wdrożenia. Przy niepewnym potwierdzeniu rejestracji zachowaj pliki tożsamości — nie podmieniaj klucza w ciemno. Nieudana konfiguracja jest zgłaszana typem wyjątku, a przy błędach HTTP także kodem statusu lub kategorią transportu (na przykład `HttpRequestException (HTTP 401)` albo `HttpRequestException (SecureConnectionError)`), bez wypisywania kodów; odrzucona rejestracja zgłasza status samej rejestracji, a nie próby odzyskania.

15 Budowanie pakietów

Operator wykonuje `bash scripts/package-agent.sh linux-x64` lub wybiera linux-arm64, win-x64, win-arm64. Pakiety tworzy się osobno dla architektur. Pobranie runtime może wymagać dostępu do oficjalnego feedu Microsoft NuGet. Ustaw `CLM_AGENT_VERSION` (major.minor.patch) i `CLM_RELEASE_SIGNING_KEY` (ścieżka offline'owego klucza ECDSA P-384), aby powstał podpisany manifest wydania dla aktualizacji. Z produkcyjnymi mechanizmami `pkcs11` i `external` skrypt odmawia działania bez klucza ECDSA P-384 `CLM_RELEASE_SIGNING_KEY` i sprawdza zapisany podpis; tylko mechanizmy deweloperskie `pfx` i `unsigned-development` mogą utworzyć niepodpisany manifest, oznaczony „DEVELOPMENT PACKAGE", który agenci odrzucają jako aktualizację (a instalatory odrzucają brak podpisu wydawcy `unsigned-development`, dopóki nie ustawiono ich wyjątku deweloperskiego). Oba klucze podpisu trzymaj poza serwerami CLM. Podpis kodu wydawcy wymaga dodatkowo OpenSSL 3, a dla pakietów Windows `osslsigncode` (Authenticode w Linux). Skrypt nie działa bez wyboru mechanizmu podpisu w `CLM_SIGN_BACKEND`; podpis wykonuje `scripts/sign-agent-package.sh`:

- `pkcs11` (produkcja): klucz pozostaje w HSM, karcie lub chmurowym HSM z modułem PKCS#11. Ustaw `CLM_SIGN_PKCS11_MODULE` (moduł dostawcy), `CLM_SIGN_KEY_URI` (URI `pkcs11:` z `pin-source=file:` wskazującym plik PIN dostępny tylko dla właściciela; PIN w URI jest odrzucany) i `CLM_SIGN_CERT_CHAIN` (najpierw certyfikat wydawcy, potem pośrednie). Wymagany jest silnik PKCS#11 OpenSSL (libp11).
- `external` (produkcja): `CLM_SIGN_AUTHENTICODE_COMMAND` (wywołanie `polecenie <wej> <wyj>`) i `CLM_SIGN_CMS_COMMAND` (wywołanie `polecenie <wej> <wyj.p7s>`, odłączony CMS DER z SHA-256) obudowują narzędzia operatora, np. signtool na hoście Windows lub narzędzie powiązane z chmurową usługą kluczy. Oba muszą używać tego samego certyfikatu wydawcy.
- `pfx` (tylko deweloperskie): `CLM_SIGN_PFX_FILE` i `CLM_SIGN_PFX_PASSWORD_FILE` dostępne tylko dla właściciela i poza repozytorium oraz `CLM_SIGN_ALLOW_DEVELOPMENT_KEY=1`. Certyfikat wydawcy musi być certyfikatem końcowym do podpisywania kodu (`CA:FALSE`): Windows odrzuca podpisy Authenticode certyfikatu CA nawet wtedy, gdy jest zaufany, dlatego `sign-agent-package.sh` przerywa pracę dla pakietów Windows podpisywanych takim certyfikatem. `scripts/create-development-publisher.sh <new-directory>` tworzy deweloperskie CA wydawcy i certyfikat końcowy wraz z plikiem PFX i hasła; na deweloperskim hoście Windows zaufaj jego `publisher-ca.crt` (`Import-Certificate -FilePath publisher-ca.crt -CertStoreLocation Cert:\LocalMachine\Root`) albo przekaż go jako `-PublisherCaFile`, nigdy na hostach klientów.
- `unsigned-development`: manifesty skrótów bez podpisów; weryfikatory i instalatory odrzucają takie pakiety bez jawnego obejścia deweloperskiego.

Kompilacja dla Windows zapisuje też obok archiwów podpisany Authenticode plik `verify-agent-package.ps1`; opublikuj go razem z nimi, aby `Get-AuthenticodeSignature .\verify-agent-package.ps1` (polecenie z konsoli) pokazywało przypiętego wydawcę.

`CLM_SIGN_TIMESTAMP_URL` dodaje znaczniki czasu RFC 3161 do podpisów Authenticode i CMS, aby pakiety dało się weryfikować po wygaśnięciu certyfikatu wydawcy. `CLM_SIGN_PUBLISHER_SHA256` ponownie sprawdza wynik względem oczekiwanego odcisku przed zakończeniem budowania. Krok wydania wypisuje odcisk wydawcy; publikuj go niezależnym kanałem do klientów, nigdy tylko obok pobierania. Klucze, PIN-y i pliki PFX nigdy nie trafiają do pakietu ani wyjścia budowania i nie mogą być commitowane. Publikuj archiwum razem z plikami `.sha256`, `.sha256.p7s` (jeśli używany) `.p7s.tsr`, `.release.json` i `.release.json.sig`. Gdy obrazy kontenerów są budowane bez podpisanych archiwów (deploy/), skopiuj cały zestaw plików pakietu do katalogu ustawionego jako `Clm:AgentPackages:Directory` (wdrożenie montuje `$CERTYX_ROOT/agent-packages`); jest on udostępniany pod `/downloads/` przed wwwroot/downloads, na hoście konsoli i na hoście agentów. Weryfikatory są udostępniane osobno pod `/downloads/verify-agent-package.sh`, `/downloads/verify-agent-package.ps1` i `/downloads/verify-release.sh`.

15.1 Powtarzalne kompilacje, SBOM i pochodzenie

`scripts/build-agent.sh <rid> <katalog>` buduje niepodpisane pliki binarne agenta powtarzalnie: w obrazie .NET SDK przypiętym skrótem w `deploy/Dockerfile`, z `git archive` commita, z deterministycznym wynikiem kompilatora (`ContinuousIntegrationBuild`, mapowane ścieżki źródeł, commit jako rewizja źródeł) i `SOURCE_DATE_EPOCH` równym czasowi commita. Zapisuje `clm-agent-<rid>.build.tar.gz` (posortowane, właściciel 0, stałe czasy), `clm-agent-<rid>.cdx.json` (SBOM CycloneDX 1.5 wszystkich pakietów NuGet i pakietów środowiska uruchomieniowego z wyniku restore, bez

narzędzi sieciowych) i `clm-agent-<rid>.build.json` (wersja, commit, SDK, obraz i SHA-256 każdego pliku binarnego). Dwie kompilacje tego samego commita dają identyczne pliki. `scripts/package-agent.sh` używa tego samego skryptu (`--local`, z SDK maszyny budującej) i umieszcza SBOM oraz zapis kompilacji w każdym pakiecie, gdzie obejmuje je `SHA256SUMS`.

Workflow GitHub `agent-release.yml` (tag `agent-v<wersja>`) buduje tak wszystkie cztery środowiska, poświadcza pochodzenie kompilacji (SLSA, `actions/attest-build-provenance`, gdy `CERTYX_ATTESTATIONS=true`) i po zatwierdzeniu w środowisku `agent-release` dołącza kompilacje do szkicu wydania. Klucze podpisu nie trafiają do GitHub. Przed podpisaniem sprawdź:

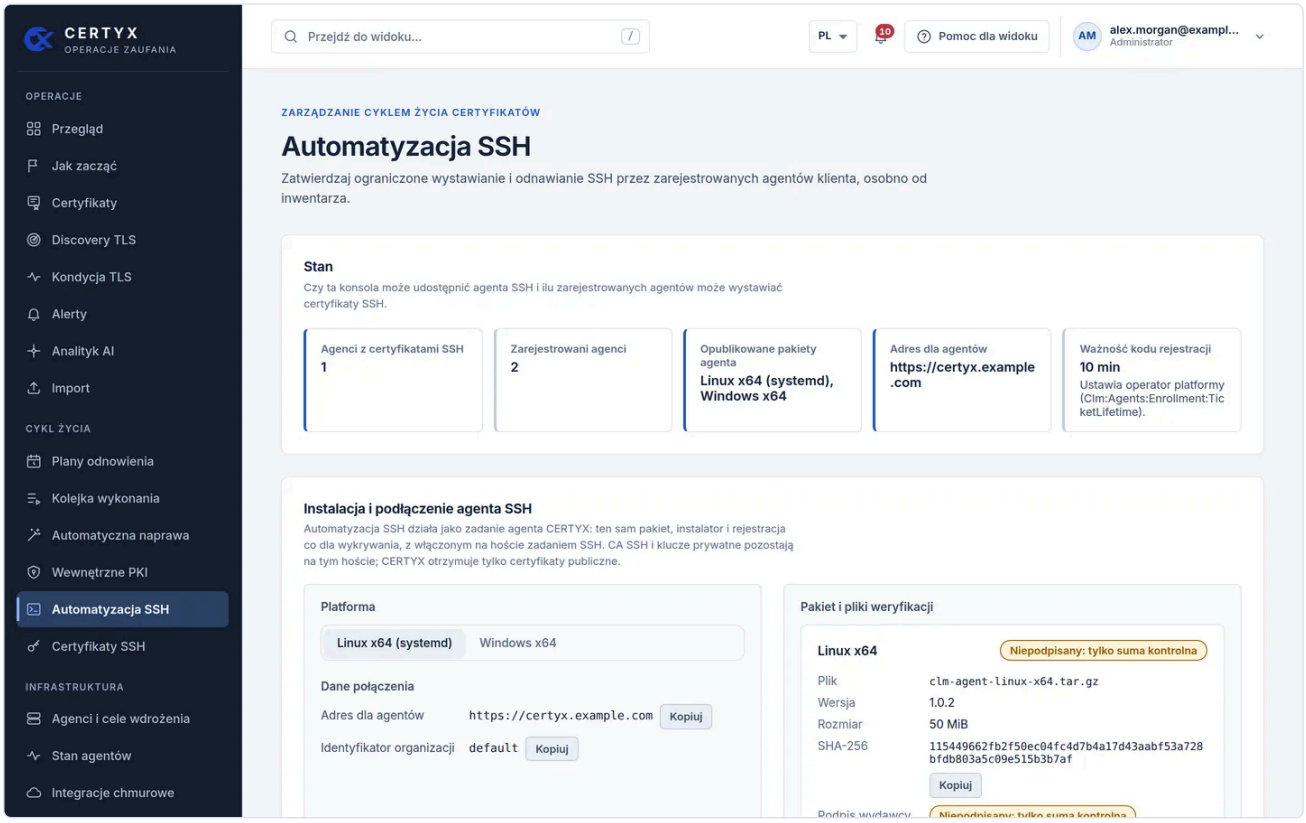
1. Pochodzenie: `gh attestation verify clm-agent-linux-x64.build.tar.gz --repo <owner>/<repository>` (także dla `.cdx.json` i `.build.json`).
2. Powtarzalność, niezależnie od GitHub: `git checkout agent-v<wersja>` oraz `CLM_AGENT_VERSION=<wersja> bash scripts/build-agent.sh linux-x64 /tmp/rebuild`; porównaj `cmp` plik `.build.tar.gz` z pobranym (albo skróty `files` obu `.build.json`).
3. Spakuj i podpisz offline dokładnie te pliki binarne: `CLM_AGENT_BUILD=<katalog z trzema plikami> CLM_AGENT_VERSION=<wersja> CLM_SIGN_BACKEND=pkcs11 CLM_RELEASE_SIGNING_KEY=<klucz> bash scripts/package-agent.sh linux-x64`; skrypt odrzuca pliki binarne różne od zapisu kompilacji.

Na hoście klienta `clm-agent-<rid>.build.json` w pakiecie podaje commit i SHA-256 pliku binarnego (`sha256sum Clm.Agent`); pliki binarne Windows mają podpis Authenticode dodany po kompilacji, więc ich skrót różni się od zapisu kompilacji; kompilacje Windows sprawdzaj przez `.build.tar.gz` wydania (krok 2).

Źródła: https://learn.microsoft.com/en-us/dotnet/core/deploying/ i https://learn.microsoft.com/en-us/powershell/module/scheduledtasks/new-scheduledtaskprincipal

16 SSH

16.1 Zadanie SSH: pobranie, instalacja i konfiguracja (Linux i Windows)



Automatyzacja SSH: instalacja i połączenie agenta SSH, zarządzane profile i zatwierdzenia (dane przykładowe)

Zadanie SSH jest częścią tego agenta; nie ma osobnego pakietu SSH. W CERTYX → **Automatyzacja SSH** → **Instalacja i podłączenie agenta SSH** wybierz Linux x64 lub Windows x64: panel pokazuje kartę pakietu tej platformy (wersja, rozmiar, SHA-256 z przyciskiem Kopiuj, stan podpisów i przyciski pobierania pakietu, jego `.sha256`, `.sha256.p7s` (gdy podpisany) i weryfikatora), tworzy kod rejestracji i pokazuje polecenia z tej instrukcji z wypełnionym adresem dla agentów i identyfikatorem organizacji (przycisk pakietu działa w zalogowanej sesji konsoli). Najpierw skopiuj clm-enrollment.token do pustego katalogu roboczego na serwerze, bo pobieranie pakietu wysyła tam kod (zob. „Pobieranie pakietów"). Po instalacji:

1. Nadaj agentowi w tym panelu **Certyfikaty SSH** (lub Agenci i cele wdrożenia → Włącz operacje SSH); **Instalację SSH** tylko wtedy, gdy agent ma instalować certyfikaty hosta lub użytkownika.
2. Linux: `sudo bash /opt/clm-agent/current/configure-ssh-linux.sh init` — Windows (PowerShell administratora): `powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\configure-ssh-windows.ps1" init`. Tworzy CA SSH hosta (`/etc/clm-ssh`, właściciel clm-agent, tryb 0700; `C:\ProgramData\CLM-SSH`, tylko SYSTEM i Administratorzy) i wypisuje `CA fingerprint: SHA256:…`.

3. Utwórz profil z tym odciskiem w Automatyzacji SSH, niech zatwierdzi go drugi administrator, i pobierz go (**Pobierz profil**, `ssh-profile-<ID>.json`).
4. `… configure-ssh-linux.sh add-profile ./ssh-profile-<ID>.json` lub `… configure-ssh-windows.ps1 add-profile .\ssh-profile-<ID>.json`: zapisuje `managed.json`, wykonuje jeden sprawdzony cykl `ssh manage --once` i uruchamia `clm-ssh.service` (Linux, użytkownik clm-agent) lub zadanie harmonogramu **CERTYX SSH Workload** (Windows, LocalSystem). Nieudany cykl podaje przyczynę, np. `HttpRequestException (HTTP 403)` bez uprawnienia SSH; nic nie jest uruchamiane.
5. Opcjonalnie `trust-user-ca`: lokalny sshd ufa certyfikatom użytkowników tego CA (`TrustedUserCAKeys`, sprawdzane przez `sshd -T`, kopia przywracana, gdy sshd się nie przeładuje). W Windows Server 2025 domyślny sshd_config dopuszcza tylko grupy `administrators` i `OpenSSH Users` (`AllowGroups`): dodaj konta logujące się certyfikatami użytkownika do `OpenSSH Users` (`Add-LocalGroupMember -Group 'OpenSSH Users' -Member <ACCOUNT>`) i uruchom serwer OpenSSH raz przed `trust-user-ca`, bo tworzy on `sshd_config` przy pierwszym starcie.
6. Sprawdzenie: `sudo clm-agent status` (Windows: `Clm.Agent.exe status agent.json`) wypisuje `[ OK ] Capabilities: discovery, ssh; …`; `configure-ssh-linux.sh status` / `configure-ssh-windows.ps1 status` pokazują usługę lub zadanie; tabela w Automatyzacji SSH pokazuje *Certyfikaty SSH: Nadane* i raport zadania SSH.

Wykrywanie i zadanie SSH dzielą zarejestrowaną tożsamość; rotację serializuje `identity-rotation.lock`. Do weryfikowanej instalacji certyfikatów hosta w Linuksie (nadana **Instalacja SSH**) `sudo bash /opt/clm-agent/current/configure-ssh-linux.sh enable-host-deploy` pozwala kontu clm-agent uruchamiać przez sudo dokładnie `sshd -t` i przeładowanie sshd (reguła sprawdzona przez visudo i plik drop-in usługi; `disable-host-deploy` usuwa oba). Szczegóły, reguły ACL Windows i cele instalacji na hoście: SSH-LIFECYCLE.md.

Lokalne operacje SSH są dostępne w Linuksie i Windows (OpenSSH dla Windows) z zainstalowanym OpenSSH ssh-keygen. Wystawiają certyfikaty Ed25519 użytkowników/hostów i generują KRL w ramach zatwierdzonej lokalnej polityki CA/principali. Zobacz SSH-LIFECYCLE.md i instrukcję widoku SSH. Zadania platformy i zweryfikowaną lokalną rotację hosta obsługuje osobno zatwierdzony ssh manage. Samo wystawienie nie instaluje certyfikatów użytkowników ani zasad zaufania.

Lokalne automatyczne odnawianie SSH bez raportowania do CERTYX (`Clm.Agent ssh renew`) instaluje polecenie `sudo bash /opt/clm-agent/current/configure-ssh-linux.sh enable-renewal ./ssh-profile-<PROFILE-ID>.json` (Windows: `configure-ssh-windows.ps1 enable-renewal .\ssh-profile-<PROFILE-ID>.json`). Skrypt zapisuje `renewal.json` z lokalnej polityki CA i zatwierdzonego profilu, wykonuje jeden cykl odnawiania i włącza `clm-ssh-renewal.timer` (co minutę usługa z jednym cyklem na koncie `clm-agent`, bez dostępu do sieci, identyfikator w dzienniku `clm-ssh-renewal`; Windows: zadanie harmonogramu `CERTYX SSH Renewal` jako LocalSystem, dziennik `state\ssh-renewal.log`). Cykl, w którym profil ma stan failed, blocked lub policy-conflict, kończy się kodem 3 i pozostawia `clm-ssh-renewal.service` w stanie failed (`systemctl --failed`); `renewal-status` pokazuje ostatni cykl, `disable-renewal` usuwa timer, usługę i `renewal.json` (certyfikaty i klucze pozostają). Profil odnawia albo `add-profile` (zadania zatwierdzone przez platformę, raportowane do CERTYX), albo `enable-renewal`, nigdy oba naraz. Profile, odzyskiwanie i stany opisano w `SSH-LIFECYCLE.md`.

17 Sekrety kontrolowane przez klienta i subskrypcje KRL

CUSTOMER-VAULTS.pl.md opisuje lokalne referencje Azure/AWS/GCP i HashiCorp Vault oraz test secret-check. SSH-LIFECYCLE.md opisuje publish-krl/sync-krl, podpisaną godzinną ważność, osobne uprawnienia relaya i odzyskiwanie. Podpisaną listę KRL publikuje publish-krl, a odbiorca sync-krl pobiera ją przez HTTPS klienta; brak aktualizacji zgłasza błąd lokalnie i zachowuje dotychczasowe unieważnienia. Szablony examples/ssh-krl-publisher.json i examples/ssh-krl-subscription.json wymagają podmiany wszystkich wartości zastępczych, w tym niezerowego UUID organizacji i niezależnie sprawdzonego odcisku CA. Samodzielny sync-krl zgłasza błędy tylko lokalnie; centralny stan odbiorcy daje `ssh monitor-krl` z osobno zarejestrowaną tożsamością, której jawnie nadano raportowanie KRL (widoczny w Stanie agentów).

18 Tożsamości użytkowników SSH dla procesów

Aktualny agent Unix obsługuje zatwierdzone profile Użytkownik z włączoną instalacją i zgodnymi lokalnymi UserTargets. Sprawdź SSH-LIFECYCLE.md oraz examples/ssh-user-target.json. Agent działa pod kontem procesu, wymaga działającej chronionej wersji bazowej, weryfikuje logowanie certyfikatem do osobno przypiętego serwera i wycofuje błędną aktywację. Proces używa current/identity i current/identity-cert.pub. Przed przypisaniem instalacji zaktualizuj agenta; zakładanie kont, Windows i zaufanie CA na serwerze pozostają osobnymi czynnościami.

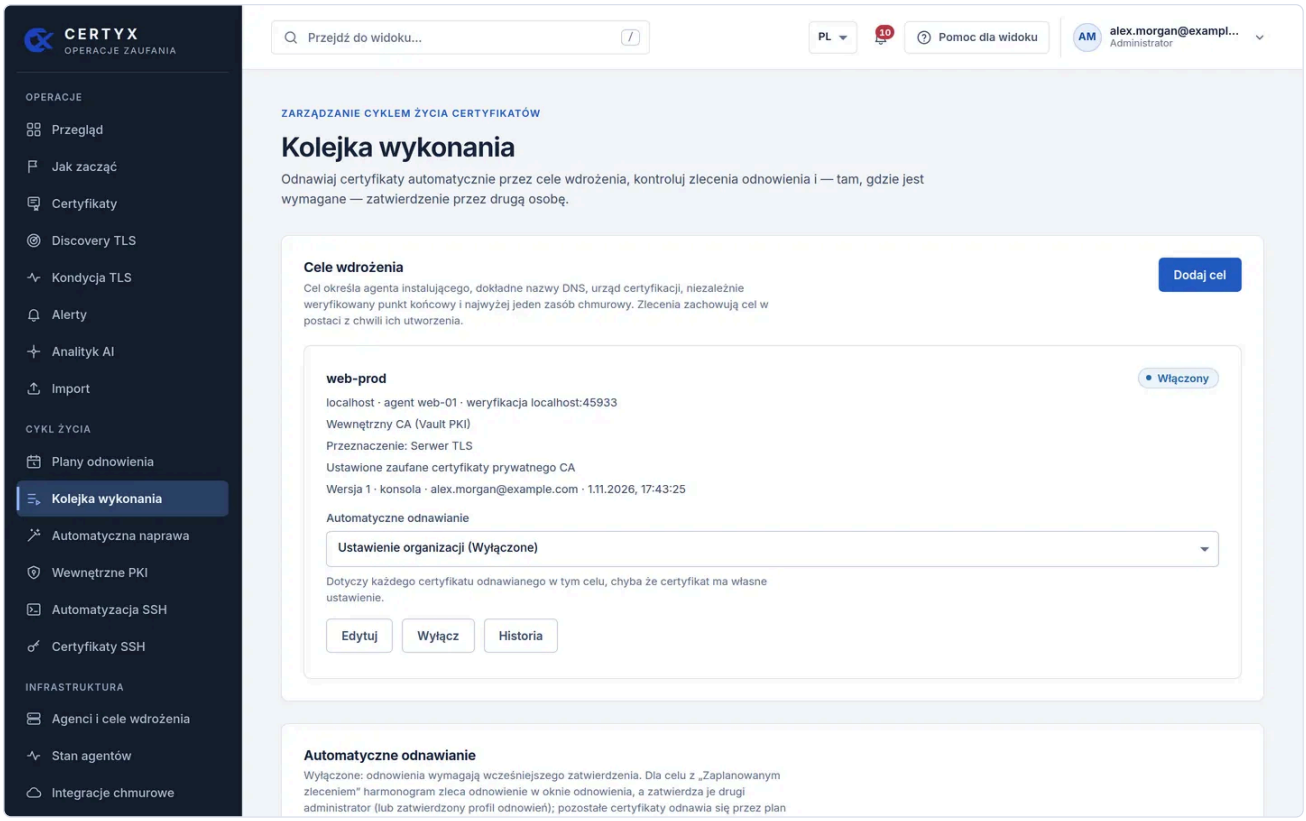
19 Cele dla certyfikatów klienta TLS

Cel cyklu życia może otrzymać certyfikat klienta TLS (użytkownicy, urządzenia, usługi w mutual TLS) zamiast certyfikatu serwera. Ustaw `Purpose` na `client` (lub `server-client` dla obu zastosowań; brak oznacza `server`) i podaj tożsamości w `Client`: `Emails` (rfc822Name, np. `jan.kowalski@example.com`), `UserPrincipalNames` (np. `jkowalski@corp.example.com`), `Uris` (np. `spiffe://example.org/ns/prod/sa/payments`) i opcjonalnie `CommonName` (1–64 znaki; wymagane, gdy podano tylko URI). Dla certyfikatu klienta `Domains` może być puste (`[]`). Wartości muszą dokładnie odpowiadać celowi w konsoli, w przeciwnym razie agent odrzuca zlecenie jako spoza lokalnego zakresu. Przeznaczenia klienckie wymagają `AuthorityKind: "vault"` i roli Vault z `client_flag=true` (`server_flag=false` dla samego klienta); cel kliencki z ACME lub DigiCert albo z zasobem chmurowym jest odrzucany przy starcie z tym powodem. Cel wyłącznie kliencki nie ma punktu końcowego: pozostaw puste `VerifyHost` i `Executable` albo ustaw `Executable` / `Arguments` na stałe bezwzględne polecenie (np. restart usługi) uruchamiane po aktywacji. Agent zapisuje `current/fullchain.pem` i `current/privkey.pem` (tylko właściciel), a gdy `Pkcs12PasswordReference` wskazuje chroniony sekret klienta, także

`current/bundle.p12` (PBES2 AES-256/SHA-256). Następnie lokalnie sprawdza, że zainstalowany klucz pasuje, certyfikat dopuszcza clientAuth, a łańcuch prowadzi do `TrustRootsFile`; błąd przywraca poprzednią generację. Tylko Unix; certyfikaty klienta nie są instalowane w magazynie certyfikatów Windows ani przez przekaźnik SSH.

```
{
    "Id": "vpn-jane",
    "Domains": [],
    "Purpose": "client",
    "Client": { "Emails": ["jane.doe@example.com"], "UserPrincipalNames": ["jdoe@corp.example.com"] },
    "DeploymentDirectory": "/etc/clm/client/vpn-jane",
    "KeyDirectory": "/var/lib/clm-agent/keys",
    "AuthorityKind": "vault",
    "VaultOrigin": "https://vault.example.com",
    "VaultMount": "pki_int",
    "VaultRole": "client-users",
    "VaultTokenFile": "/etc/clm/secrets/vault-token",
    "TrustRootsFile": "/etc/clm/trust/internal-roots.pem",
    "Executable": "",
    "Arguments": [],
    "VerifyHost": "",
    "VerifyPort": 0,
    "LifetimeHours": 720,
    "Pkcs12PasswordReference": { "Provider": "protected-file", "Location": "/etc/clm/secrets/vpn-jane-p12.password" }
}
```

20 Konektory wdrożenia i przekaźnik SSH



Kolejka wykonania: cele wdrożeń i zadania, które instalują odnowione certyfikaty i weryfikują każdą instalację (dane przykładowe)

Odnowiony certyfikat serwera TLS jest instalowany w jednym z dwóch trybów:

- **Agent na hoście** (`local-agent`): agent cyklu życia działa na maszynie, której usługa używa certyfikatu, generuje tam klucz i tam instaluje pliki.
- **Przekaźnik SSH** (`ssh-relay`): agent cyklu życia instaluje certyfikat przez SSH na innym hoście, który nie ma agenta. Klucz powstaje na tym hoście i nigdy go nie opuszcza; przekaźnik otrzymuje tylko CSR.

Bez agenta na hoście lub przekaźnika z dostępem SSH do niego CERTYX nie może zainstalować certyfikatu automatycznie; wykrywanie, monitorowanie, plany odnowienia i przypomnienia nadal działają, a certyfikat instaluje się wtedy ręcznie.

20.1 Konektory

Miejsce wdrożenia podaje swój konektor w `Connector`. Bez `Connector` (lub z `nginx` bez `Files` i bez `Remote`) obowiązuje bez zmian pierwotny układ Nginx: generacje w `DeploymentDirectory`, `current/fullchain.pem` i `current/privkey.pem` oraz `Executable` / `Arguments` uruchamiane z `-t` i `-s reload`. Każdy inny konektor publikuje pliki wymienione w `Files` w ścieżkach, których usługa już używa:

Konektor	Wymagane pliki	Domyślne sprawdzenie	Domyślne przeładowanie	Tryb pliku klucza	Uwagi
`pem-files`	dowolne z rodzajów plików	brak	brak	0600	Ogólny; ustaw `CheckCommand` i `ReloadCommand` dla usługi.
`nginx`	`FullChain` + `PrivateKey` lub `Combined`	`/usr/sbin/nginx -t`	`/usr/sbin/nginx -s reload`	0600	`ssl_certificate` / `ssl_certificate_key`.
`apache`	`FullChain` + `PrivateKey` lub `Certificate` + `Chain` + `PrivateKey`	`/usr/sbin/apachectl -t`	`/usr/sbin/apachectl graceful`	0600	`SSLCertificateFile` (pełny łańcuch od Apache 2.4.8), `SSLCertificateKeyFile`, `SSLCertificateChainFile` dla osobnego łańcucha.
`haproxy`	`Combined` (pełny łańcuch, po nim klucz)	`/usr/sbin/haproxy -c -f /etc/haproxy/haproxy.cfg`	`/usr/bin/systemctl reload haproxy`	0600	`bind :443 ssl crt <Combined>`.
`lighttpd`	`FullChain` + `PrivateKey` lub `Combined`	`/usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf`	`/usr/bin/systemctl reload lighttpd`	0600	`ssl.pemfile` i `ssl.privkey` (lighttpd 1.4.53 lub nowszy).
`caddy`	`FullChain` + `PrivateKey`; wymagane `Group`	`/usr/bin/caddy validate --config /etc/caddy/Caddyfile`	`/usr/bin/caddy reload --config /etc/caddy/Caddyfile --force`	0640	`tls <FullChain> <PrivateKey>` w bloku witryny; Caddy nie pobiera wtedy tego certyfikatu własnym ACME.
`traefik`	`FullChain` + `PrivateKey`	brak	brak	0600	Dostawca plików z `watch: true`; umieść pliki w katalogu obserwowanym przez dostawcę, aby zmiana nazwy wywołała przeładowanie.
`envoy`	`FullChain` + `PrivateKey`; wymagane `Group`	brak	brak	0640	SDS z plików (`path_config_source` z `watched_directory` na katalogu certyfikatów); zmiana nazwy wywołuje rotację. Dla SDS konfiguracja startowa wymaga `node` z `id` i `cluster`.
`cribl`	`FullChain` + `PrivateKey`; wymagane `Group`	brak	`/usr/bin/systemctl restart cribl`	0640	Wskaż pliki w ustawieniach TLS Cribl (serwer API lub Source: `certPath`, `privKeyPath`); Cribl Edge: `ReloadCommand` `["/usr/bin/systemctl","restart","cribl-edge"]`. Jedno miejsce wdrożenia na host Leader lub Worker. Bez systemd `["/opt/cribl/bin/cribl","restart"]`.
`tomcat`	`Pkcs12` lub `JavaKeystore`; wymagane `Group` i `Pkcs12PasswordReference`	brak	`/usr/bin/systemctl restart tomcat`	0640	`certificateKeystoreFile` z `certificateKeystoreType="PKCS12"` i tym samym hasłem; ustaw `ReloadCommand` na nazwę swojej jednostki (np. `tomcat10`).
`java-keystore`	`JavaKeystore`; wymagane `Pkcs12PasswordReference`	brak	brak	0600	keytool importuje pakiet PKCS#12 do `KeystoreType JKS` (domyślnie) lub `PKCS12` z `KeystoreAlias`; ustaw `ReloadCommand`, aby zrestartować aplikację.
`postfix`	`FullChain` + `PrivateKey`	`/usr/sbin/postfix check`	`/usr/sbin/postfix reload`	0600	`smtpd_tls_chain_files = <PrivateKey>, <FullChain>`. Weryfikacja wymaga niejawnego TLS: `VerifyPort` 465.
`dovecot`	`FullChain` + `PrivateKey`	`/usr/bin/doveconf -n`	`/usr/bin/doveadm reload`	0600	`ssl_cert = <<FullChain>` i `ssl_key = <<PrivateKey>` (Dovecot 2.3; 2.4 używa `ssl_server_cert_file` i `ssl_server_key_file`). Weryfikacja wymaga niejawnego TLS: `VerifyPort` 993 lub 995.
`kubernetes-secret`	brak; obiekt `Kubernetes`	brak	brak	—	`kubectl create secret tls … --dry-run=client` i `kubectl apply` na hoście wdrożenia (`tls.crt` = pełny łańcuch, `tls.key`); `VerifyHost` to Ingress lub Gateway, który go serwuje. Ustaw `ReloadCommand` (np. `kubectl rollout restart`), gdy obciążenie nie obserwuje sekretu.
`windows-store`	brak; obiekt `WindowsStore`	—	opcjonalnie `ReloadCommand`	—	Tylko agent Windows: magazyn LocalMachine, nieeksportowalny klucz maszynowy.
`iis`	brak; `WindowsStore` z powiązaniem	—	`netsh http update sslcert`	—	Tylko agent Windows; powiązanie HTTPS musi istnieć (utworzone raz w Menedżerze IIS).

Pola miejsca wdrożenia z konektorem (wszystkie z chronionej lokalnej konfiguracji; zadanie tylko wybiera miejsce):

- `Files`: `Certificate` (certyfikat końcowy), `Chain` (łańcuch wystawcy bez certyfikatu końcowego), `FullChain` (certyfikat końcowy i łańcuch), `PrivateKey`, `Combined` (pełny łańcuch, po nim klucz), `Pkcs12` (PKCS#12 AES-256) i `JavaKeystore`, każde jako bezwzględna ścieżka z liter, cyfr i `. _ / @ + = , : % -`, w istniejącym katalogu, do którego nie może pisać każdy, bez dowiązań symbolicznych w ścieżce, różne od siebie i poza magazynem generacji. `Owner` i `Group` ustawiają właściciela i grupę każdego publikowanego pliku (agent lub konto przekaźnika musi móc zmieniać własność, zwykle root); `Mode` (domyślnie `0644`, bez zapisu dla grupy i innych) dotyczy plików publicznych, `KeyMode` (domyślny dla konektora, `0600` lub `0640`, nigdy czytelny dla innych) plików zawierających klucz. `KeystoreType` (`JKS` lub `PKCS12`), `KeystoreAlias` (domyślnie `certyx`) i `KeytoolExecutable` (domyślnie `/usr/bin/keytool`, tylko agent na hoście) dotyczą `JavaKeystore`.
- `CheckCommand` i `ReloadCommand`: tablice argumentów zaczynające się od bezwzględnej ścieżki programu, np. `["/usr/sbin/nginx","-t"]`; argumenty używają tego samego zestawu znaków (bez powłoki, cudzysłowów, `$`, `;` czy `|`). Zastępują domyślne polecenia konektora; `[]` wyłącza polecenie domyślne. Przez SSH działają za `Remote.SudoPrefix`.
- `Pkcs12PasswordReference`: referencja sekretu klienta z hasłem PKCS#12 i magazynu kluczy (CUSTOMER-VAULTS.pl.md); przez SSH hasło trafia na host zdalny przez standardowe wejście szyfrowanej sesji, nigdy w wierszu poleceń.
- `Kubernetes`: `KubeconfigFile`, `Namespace`, `SecretName`, opcjonalnie `Context` oraz `KubectlExecutable` (domyślnie `/usr/local/bin/kubectl`, tylko agent na hoście).
- `WindowsStore`: `StoreName` (`My` lub `WebHosting`), `BindingHostName` (powiązanie SNI) lub `BindingAddress` (domyślnie `0.0.0.0`), `BindingPort` (443), `BindingAppId` (IIS: `{4dc3e181-e14b-4a21-b022-59fc669b0914}`) i `NetshExecutable`.
- `DeploymentDirectory`: na hoście agenta, prywatny (0700); zawiera dziennik i blokadę wdrożenia, a na własnym hoście agenta także magazyn generacji. `VerifyHost` i `VerifyPort` wskazują punkt końcowy TLS, który agent sprawdza po przeładowaniu (wymagane).

Co agent robi dla miejsca wdrożenia z konektorem: najpierw kończy przerwane wdrożenie; tworzy magazyn (0700) oraz na hoście wdrożenia klucz RSA-3072 z CSR (CSR musi mieć prawidłowy podpis i dokładnie zatwierdzone nazwy DNS); zleca podpis CA; sprawdza, że certyfikat zawiera dokładnie ten klucz publiczny i te nazwy, dopuszcza uwierzytelnianie serwera TLS i prowadzi do `TrustRootsFile`; zapisuje nową generację (`privkey.pem` 0600, `cert.pem`, `chain.pem`, `fullchain.pem`, pakiety); publikuje każdy plik atomowo (nowy plik obok docelowego, potem zmiana nazwy) z właścicielem, grupą i trybem; uruchamia sprawdzenie i przeładowanie; i weryfikuje przez TLS, że `VerifyHost:VerifyPort` prezentuje dokładnie nowy certyfikat. Następnie CERTYX niezależnie weryfikuje punkt końcowy, jak dla każdego celu serwerowego. Magazyn przechowuje bieżącą i poprzednią generację. Gdy sprawdzenie, przeładowanie lub weryfikacja się nie powiedzie, poprzednia generacja jest publikowana ponownie, sprawdzana, przeładowywana i weryfikowana, a zadanie zgłasza `rolled-back`. Przed pierwszym wdrożeniem agent kopiuje pliki opublikowane w tej chwili (bajt w bajt, z trybem i właścicielem) jako stan wyjściowy, który przywraca; gdy żaden z plików nie istnieje, nie ma czego przywracać i agent zatrzymuje się z komunikatem „Intervention required", zachowując `journal.json` w katalogu wdrożenia: sprawdź usługę, a potem usuń dziennik. Stan wyjściowy, który ma tylko plik `Combined`, PKCS#12 lub magazyn kluczy, jest przywracany bez sprawdzenia odcisku, bo jego odczyt skopiowałby klucz.

Przykłady: `examples/agent.deploy-<konektor>.json` dla każdego konektora i `examples/agent.ssh-relay.json`.

20.2 Przekaźnik SSH

Przekaźnik to zarejestrowany agent cyklu życia z uprawnieniem **Włącz wykonywanie cyklu życia certyfikatów** (`deployment`). Każde zdalne miejsce wdrożenia musi też być wpisane w `RelayDestinations` jego konfiguracji, a cel wdrożenia musi podawać tryb instalacji **przekaźnik SSH** z tym samym hostem zdalnym (Kolejka wykonania → Cele wdrożenia → Instalacja); przekaźnik odrzuca każde zadanie, którego cel podaje inny tryb lub host. Wymagania na hoście zdalnym: serwer OpenSSH, powłoka logowania POSIX dla konta przekaźnika (sh, bash, dash lub zsh), zwykłe narzędzia systemowe (`mktemp`, `find`, `chmod`, `chown`, `mv`, `cp`, `sed`, `sort`), OpenSSL 1.1.1 lub nowszy w `/usr/bin` lub `/usr/local/bin`, keytool dla magazynów Java, kubectl dla Kubernetes oraz `ss` (iproute2) dla `relay-check`. Przekazywane są tylko certyfikaty serwera TLS; certyfikaty klienta i magazyn Windows wymagają agenta na hoście.

Kroki 2, 4, 5 i 6 na hoście zdalnym to jedno polecenie z pakietu agenta: skopiuj `configure-relay-host-linux.sh` (z `/opt/clm-agent/current` na przekaźniku), klucz publiczny przekaźnika i pomocnika (krok 4) na host zdalny i uruchom jako root:

```
sudo sh configure-relay-host-linux.sh enable --account certyx-deploy --public-key ./id_ed25519.pub \
  --helper ./certyx-relay-helper --from 10.0.0.5 --root /etc/haproxy/certs --root /var/lib/certyx-relay \
  --allow "/usr/sbin/haproxy -c -f /etc/haproxy/haproxy.cfg" --allow "/usr/bin/systemctl reload haproxy"
```

Skrypt tworzy konto (tylko logowanie kluczem), zapisuje `authorized_keys` z `restrict` i `from=`, instaluje pomocnika, zapisuje plik roots, tworzy brakujące katalogi, zapisuje regułę sudoers dla pomocnika i dokładnie poleceń `--allow` (sprawdzoną przez `visudo`) i wypisuje odciski kluczy hosta do przypięcia; `status --account certyx-deploy` pokazuje konfigurację, a `disable --account certyx-deploy` ją usuwa (opublikowane certyfikaty i katalogi magazynu pozostają). Działa z `useradd` (Debian, Ubuntu, RHEL, SUSE) i `adduser` z BusyBox (Alpine). Poniższe kroki opisują, co robi i jak wykonać to ręcznie.

Krok po kroku (przykład: HAProxy na `web-01.internal.example`, konto przekaźnika `certyx-deploy`):

1. **Klucz przekaźnika** (na hoście przekaźnika, jako root): `install -d -m 700 /etc/clm-agent/ssh` i `ssh-keygen -t ed25519 -N '' -C certyx-relay -f /etc/clm-agent/ssh/id_ed25519`. Klucz prywatny pozostaje dostępny tylko dla właściciela na przekaźniku; nigdy nie trafia do CERTYX.
2. **Konto zdalne** (na hoście zdalnym): `useradd --system --create-home --shell /bin/sh certyx-deploy` i klucz publiczny w `~certyx-deploy/.ssh/authorized_keys` (katalog 700, plik 600) z ograniczeniami, np. `restrict,from="10.0.0.5" ssh-ed25519 AAAA… certyx-relay` (adres przekaźnika).
3. **Przypięcie klucza hosta**: na hoście zdalnym odczytaj odcisk `ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub -E sha256`. Na przekaźniku uruchom `ssh-keyscan -t ed25519 -p 22 web-01.internal.example > /etc/clm-agent/ssh/known_hosts`, `chmod 600 /etc/clm-agent/ssh/known_hosts` i porównaj `ssh-keygen -lf /etc/clm-agent/ssh/known_hosts -E sha256` z odciskiem z hosta zdalnego. Wpisz tę wartość

`SHA256:…` do `HostKeyFingerprints`. Agent odrzuca plik known_hosts zawierający inny klucz dla hosta, wpisy z symbolami wieloznacznymi lub zaprzeczeniem, wiersze `@cert-authority` lub `@revoked`; OpenSSH egzekwuje go potem ze ścisłym sprawdzaniem klucza hosta (bez zaufania przy pierwszym połączeniu). Dla innego portu wpis ma postać `[host]:port`.

4. **Pomocnik przekaźnika** (na hoście zdalnym): wypisz go na przekaźniku z podpisanego agenta poleceniem `sudo /opt/clm-agent/current/Clm.Agent relay-helper > certyx-relay-helper`, skopiuj i zainstaluj jako root: `install -o root -g root -m 0755 certyx-relay-helper /usr/local/libexec/certyx-relay-helper`; porównaj `sha256sum` na obu hostach. Pomocnik wykonuje tylko stałe operacje (zapis, kopiowanie, odczyt plików publicznych, tworzenie i usuwanie katalogów magazynu, generowanie klucza i CSR, PKCS#12, magazyn kluczy, sekret Kubernetes, nasłuchujące procesy) i znajduje swoje narzędzia w stałej zmiennej PATH. Bez `HelperPath` przekaźnik wysyła ten sam skrypt przez `/bin/sh -c`; nie wymaga to instalacji, ale nie da się tego ograniczyć w sudoers.
5. **Dozwolone katalogi**: `/etc/certyx-relay/roots` (właściciel root, tryb 0644, jeden bezwzględny katalog na wiersz) wskazuje, gdzie pomocnik może zapisywać i czytać, np. `/etc/haproxy/certs` i `/var/lib/certyx-relay`. Plik jest wymagany, gdy pomocnik działa jako root, i obowiązuje także bez sudo; każda ścieżka musi leżeć poniżej jednego z tych katalogów. Wymień tylko katalogi certyfikatów i magazyn, nigdy katalog główny serwera WWW ani katalog, do którego mogą pisać inne konta. Pomocnik odczytuje z powrotem tylko publiczne pliki certyfikatów i swój wskaźnik generacji, nigdy klucz, plik połączony ani magazyn kluczy.
6. **sudo** (gdy konto nie jest rootem; edycja przez `visudo -f /etc/sudoers.d/certyx-relay`): ``` Defaults:certyx-deploy !requiretty certyx-deploy ALL=(root) NOPASSWD: /usr/local/libexec/certyx-relay-helper, /usr/sbin/haproxy -c -f /etc/haproxy/haproxy.cfg, /usr/bin/systemctl reload haproxy ``` Wymień dokładnie pomocnika oraz polecenia sprawdzenia i przeładowania miejsca wdrożenia z ich argumentami. Ustaw `"SudoPrefix": ["/usr/bin/sudo","-n"]`. Nie zezwalaj na `/bin/sh` ani na edytor: to daje pełne uprawnienia roota.
7. **Miejsce wdrożenia** w konfiguracji cyklu życia przekaźnika (`examples/agent.ssh-relay.json`): ```json "RelayDestinations": ["web-01-haproxy"], "Destinations": [{ "Id": "web-01-haproxy", "Domains": ["www.example.com"], "Connector": "haproxy", "DeploymentDirectory": "/var/lib/clm-agent/relay/web-01-haproxy", "Files": { "Combined": "/etc/haproxy/certs/www.pem" }, "Remote": { "Host": "web-01.internal.example", "Port": 22, "User": "certyx-deploy", "KnownHostsFile": "/etc/clm-agent/ssh/known_hosts", "HostKeyFingerprints": ["SHA256:..."], "IdentityFile": "/etc/clm-agent/ssh/id_ed25519", "StoreDirectory": "/var/lib/certyx-relay/web-01-haproxy", "SudoPrefix": ["/usr/bin/sudo", "-n"], "HelperPath": "/usr/local/libexec/certyx-relay-helper" }, "VerifyHost": "www.example.com", "VerifyPort": 443, "AuthorityKind": "vault", "...": "pola CA jak w każdym miejscu wdrożenia" }] ``` Pola `Remote`: `Host` (nazwa DNS lub adres IP), `Port` (22), `User`, `KnownHostsFile` i `IdentityFile` (bezwzględne ścieżki lokalne, tylko dla właściciela, 0600), `HostKeyFingerprints` (1–8 przypiętych wartości `SHA256:`; przy rotacji klucza podaj obie), `StoreDirectory` (na hoście zdalnym, 0700: klucze i generacje), opcjonalnie `JumpHost` (`Host`, `User`, `Port`, `HostKeyFingerprints`; osiągany tą samą tożsamością i plikiem known_hosts), `ConnectTimeoutSeconds` (10, 1–120), `SshExecutable` (`/usr/bin/ssh`), `SudoPrefix` i `HelperPath`. `DeploymentDirectory` pozostaje na przekaźniku i zawiera tylko dziennik i blokadę.
8. **Sprawdzenie bez zmiany certyfikatu**: `sudo /opt/clm-agent/current/Clm.Agent relay-check <konfiguracja-cyklu-życia.json> [id-miejsca]` sprawdza przypięcie, uwierzytelnianie, wersję pomocnika i katalog magazynu, podaje proces nasłuchujący na `VerifyPort` z proponowanym konektorem (widoczny w gotowości odnowienia certyfikatu jako „Usługa wykryta przez relay-check") i uruchamia sprawdzenie konfiguracji. Kod wyjścia 0 oznacza gotowość.
9. **Cel w CERTYX**: w Kolejka wykonania → Cele wdrożenia wybierz **Instalacja: Agent jako przekaźnik SSH do hosta bez agenta** oraz host zdalny (`Remote.Host`), z agentem przekaźnika jako agentem instalującym. Gotowość odnowienia certyfikatu pokazuje wtedy przekaźnik, host zdalny i publikowane pliki, a przy różnicy trybu lub hosta blokuje z jasnym powodem.

Przekaźnik łączy się systemowym klientem OpenSSH wyłącznie z opcjami wiersza poleceń (`-F /dev/null`): `BatchMode=yes`, `StrictHostKeyChecking=yes` względem `KnownHostsFile`, uwierzytelnianie kluczem publicznym tylko z `IdentityFile` (`IdentitiesOnly=yes`, bez ssh-agent), bez przekazywania agenta i X11, bez przekierowań portów i tuneli, bez polecenia lokalnego, bez współdzielenia połączeń (`ControlMaster=no`) i z `ConnectTimeout`. Każde słowo zdalnego polecenia jest w pojedynczych cudzysłowach, więc ścieżki i argumenty pozostają dosłowne; polecenia sprawdzenia i przeładowania działają ze standardowymi strumieniami na /dev/null, więc restart uruchamiający demona (np. `cribl restart`) wraca, a liczy się tylko kod wyjścia. Każde wdrożenie przez przekaźnik trafia do dziennika bezpieczeństwa (`execution.relay.deployment.started|succeeded|failed|rolled-back` z agentem, celem, hostem zdalnym i nazwami). Jedno zdalne miejsce wdrożenia powinno należeć do jednego przekaźnika; dwa przekaźniki wdrażające te same pliki nie są koordynowane.

21 Przesyłanie zapisanego wyniku wykonania

Zapisane wyniki wykonania są ponownie przesyłane po przejściowych błędach połączenia, przekroczeniu czasu żądania oraz HTTP 408/429/500/502/503/504, z odstępem PollSeconds i ograniczeniem OperationTimeoutSeconds. Ponowienie zachowuje zadanie, dzierżawę i wynik; nie powtarza wydania ani instalacji certyfikatu. Przy odzyskiwaniu po restarcie agent odnawia dzierżawę podczas przesyłania zapisanego wyniku. Błędy uwierzytelniania, uprawnień, wygasłej lub uzgodnionej dzierżawy i zaufania TLS zatrzymują wykonanie do uzgodnienia. Mechanizm nie przywraca utraconej dzierżawy; awaria dłuższa niż dzierżawa lub limit operacji nadal wymaga uzgodnienia przez administratora.

22 Chroniony odczyt konfiguracji

Wszystkie mechanizmy odczytu konfiguracji JSON agenta odrzucają dowiązania pliku lub katalogów nadrzędnych, uprawnienia grupy lub innych użytkowników w Unix, pliki większe niż 1 MiB oraz niepoprawny UTF-8 przed deserializacją. W Unix zabezpiecz konfigurację poleceniem chmod 600. Względne ścieżki argumentów są przekształcane do bezwzględnych i sprawdzane według tych samych reguł. Obsługiwany jest UTF-8 z BOM i bez BOM. Odczyt ustawień i odczyt asynchroniczny stosują te same zasady; komunikaty błędów nie zawierają wartości konfiguracji. W Windows te same

kontrole odczytują ACL: prywatny plik lub katalog musi należeć do LocalSystem, grupy Administratorzy lub tożsamości samego agenta, a żaden inny podmiot nie może go czytać ani zmieniać (wpisy odmowy są pomijane). Pliki, których właścicielem jest pojedyncze konto administratora, są odrzucane; instalator ustawia właściciela na grupę Administratorzy.

23 Niezależna weryfikacja w prywatnej sieci

Dla endpointów dostępnych wyłącznie w sieci klienta użyj osobno zarejestrowanego weryfikatora. W widoku Agenci i cele wdrożenia, w sekcji Zarejestrowani agenci, wybierz dla instalatora **Włącz wykonywanie cyklu życia certyfikatów**, a dla obserwatora **Włącz niezależną weryfikację endpointów** (uprawnienia `deployment` i `endpoint-verification`). Używaj różnych tożsamości i kluczy prywatnych. Rejestracja discovery sama nie nadaje tych uprawnień. Zarejestruj oba procesy istniejącym mechanizmem jednorazowego tokena przez HTTPS; wygenerowane pliki tożsamości i hasła pozostają na odpowiednich hostach klienta.

Administrator organizacji ustawia ID agenta weryfikującego u klienta w celu wdrożenia w widoku Wykonanie (Cele wdrożenia), razem z hostem, portem i ścieżką kontroli weryfikacji; pole API to `verificationAgentId`. Zmiana unieważnia zatwierdzony zakres automatycznego odnowienia. Brak identyfikatora zachowuje weryfikację z serwera. Tryb zdalny wymaga zarejestrowanego instalatora: w konfiguracji lifecycle ustaw jednocześnie `tenant` i `clientCertificatePasswordFile`. Plik hasła ma pierwszeństwo przed dotychczasową zmienną środowiskową. Tożsamości instalatora i weryfikatora obracają się automatycznie przed wygaśnięciem.

Instalacja weryfikatora (jedno polecenie). Gdy cel wdrożenia wskazuje agenta weryfikującego, jego karta w Wykonanie → Cele wdrożenia pokazuje **Konfiguracja niezależnego weryfikatora endpointu**: wybierz dla tego agenta **Włącz niezależną weryfikację endpointów**, pobierz przyciskiem **Pobierz zakres weryfikatora** plik `endpoint-verifier-<TARGET-ID>.json` (ID celu, host weryfikacji, port, ścieżka kontroli stanu i zaufane certyfikaty główne celu, wyłącznie dane publiczne), skopiuj go na host weryfikatora i uruchom w katalogu z plikiem:

```
sudo bash /opt/clm-agent/current/configure-verifier-linux.sh add-target ./endpoint-verifier-<TARGET-ID>.json
```

```
powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\configure-verifier-windows.ps1" add-target .\endpoint-verifier-<TARGET-ID>.json
```

Skrypt zapisuje konfigurację weryfikatora z własnej zarejestrowanej tożsamości agenta i zakresu (Linux: `/etc/clm-verifier/verifier.json` i `roots/<TARGET-ID>.pem`, właściciel `clm-agent`, tryby 600 i 700; Windows: `C:\ProgramData\CLM-Agent\verifier`, właściciel Administratorzy, odczyt dla LOCAL SERVICE), sprawdza ją regułami zakresu samego weryfikatora, wykonuje jeden sprawdzony cykl (`verify-endpoints --once`) i uruchamia usługę: `clm-endpoint-verifier.service` na koncie `clm-agent` (system tylko do odczytu, `NoNewPrivileges`, ponowne uruchomienie po błędzie i razem z `clm-discovery.service`, więc podpisana aktualizacja uruchamia też weryfikator z nowego wydania) albo zadanie harmonogramu `CERTYX Endpoint Verifier` (LOCAL SERVICE, dziennik `state\endpoint-verifier.log`). Dopóki CERTYX odpowiada HTTP 403, bo weryfikacja nie została jeszcze nadana, skrypt czeka na nadanie do dziesięciu minut (`CLM_VERIFIER_GRANT_WAIT` lub `-GrantWaitSeconds` w sekundach; 0 kończy od razu i wskazuje krok w konsoli). Ponowne uruchomienie z tym samym celem zastępuje ten cel; kolejny plik celu dodaje cel. `remove-target <TARGET-ID>` kończy weryfikację jednego celu (z ostatnim celem usługa jest usuwana), `disable` usuwa usługę, konfigurację i zaufane certyfikaty, a `status` pokazuje usługę, ostatnie wiersze dziennika i stan agenta; `uninstall-linux.sh` i `uninstall-windows.ps1` usuwają wszystko razem z agentem. Nowy agent otrzymuje tę rolę przy instalacji: dodaj `CLM_AGENT_ENDPOINT_VERIFIER=./endpoint-verifier-<TARGET-ID>.json` do linuksowego polecenia instalacji (Windows: `-EndpointVerifierTarget .\endpoint-verifier-<TARGET-ID>.json`); instalator wykonuje wtedy ten sam krok po rejestracji agenta, czeka na nadanie uprawnienia i kończy się kodem 3, gdy agent działa, ale weryfikatora nie udało się włączyć.

Weryfikator łączy się z CERTYX z użyciem magazynu zaufania swojego hosta (krok zaufania instalatora) i sprawdza endpoint względem zaufanych certyfikatów z zakresu; zakres bez zaufanych certyfikatów (cel, którego certyfikaty główne są plikiem po stronie serwera) wymaga CA wystawcy w magazynie zaufania hosta. Produkcja wymaga HTTPS, zaufania łańcucha, uwierzytelniania serwera, sprawdzania unieważnienia online i poprawnego zegara. Konfiguracja napisana ręcznie według `examples/endpoint-verifier.json` (`identity` to konfiguracja rejestracji obserwatora) podlega tym samym regułom; `testMode` wyłącza sprawdzanie unieważnienia online i jest akceptowany tylko wtedy, gdy płaszczyzna sterowania i wszystkie cele mają adresy loopback; nie włączaj go u klienta. Weryfikator nie otrzymuje poświadczeń wydawcy, ścieżek instalacji ani kluczy prywatnych usług.

Po instalacji dzierżawa pozostaje aktywna, a instalator ponawia wyłącznie zapisany wynik. Osobny obserwator pobiera żądanie ważne 60 sekund, związane z organizacją, rewizją zadania, dzierżawą, celem i oczekiwanym SHA-256. Sprawdza rzeczywisty łańcuch TLS, nazwę endpointu, EKU serwera, unieważnienie, dokładny certyfikat i skonfigurowane zdrowie HTTP. Zwraca wyłącznie publiczny dowód certyfikatu. CLM przyjmuje sukces dopiero po zgodnym dowodzie. Brak lub błąd obserwatora nie oznacza udanego wdrożenia. Żądanie serwera nie rozszerza lokalnie zatwierdzonego zakresu. Samoweryfikacja i wspólne bieżące lub oczekujące klucze tożsamości są odrzucane.

Odzyskiwanie wykonania używa nowej obserwacji oraz zachowuje niezależne zatwierdzenie i kontrolę rewizji. UI czeka na dowód przez ograniczony czas. Przekroczenie czasu pozostawia zadanie do uzgodnienia; nie powtarza potajemnie wydania ani instalacji. Stan agentów pokazuje osobno tę funkcję, bezpieczny błąd i brak raportu. Zdrowe discovery nie ukrywa błędu weryfikatora. W lokalnym dzienniku usługi sprawdź kategorię wyjątku, zakres, CA, łączność, adres zdrowia, zegar i uprawnienia. Telemetria nie zawiera kluczy prywatnych ani poświadczeń CA. Wycofanie tożsamości lub uprawnienia blokuje akceptację dalszych obserwacji.

23.1 Diagnostyka instalatora cyklu życia

Zarejestrowany agent cyklu życia raportuje stałe, bezpieczne komunikaty rozpoczęcia, zakończenia i błędu przez uprawnienie wdrożeń. Stan agentów pokazuje Wdrożenia osobno od discovery. Podczas operacji zdrowie jest odświeżane przy każdym odnowieniu dzierżawy i zgłasza interwał heartbeat, a bezczynne cykle — interwał odpytywania. Zarejestrowany instalator wymaga interwału odpytywania co najmniej pięciu sekund. Brak raportu staje się błędem po oknie raportowania, również przy działającym discovery. Rozpoczęcie lub ukończenie cyklu oznacza zdrowie procesu, nie dowód instalacji

certyfikatu: sprawdź zadanie wykonania i niezależny dowód endpointu. Nieznane skutki wydania lub instalacji nadal zatrzymują pracę do uzgodnienia stanu. Błąd raportowania nie powtarza skutków; serwer wykrywa brak telemetrii. Instalatory ze statycznego rejestru nie są zarejestrowanymi agentami monitorowanej floty.

23.2 Zarządzane ścieżki wdrożenia

Lokalne katalogi kluczy i wdrożeń PEM wymagają bezwzględnych ścieżek bez dowiązań, także w katalogach nadrzędnych. CERTYX tworzy katalogi dostępne tylko właścicielowi i odrzuca dowiązane lub podmienione katalogi przed zapisem kluczy albo stanu. Klucze prywatne i dzienniki muszą zachować dostęp wyłącznie właściciela (Linux 600), a wygenerowane katalogi — 700. Współdzielone istniejące dzienniki lub pliki blokad wymagają poprawienia uprawnień przed restartem. Odzyskiwanie odrzuca dowiązane, współdzielone i zbyt duże dzienniki. Jedyny względny link `current` jest dozwolony dla zarządzanej generacji UUID; katalogi generacji i pliki certyfikatów muszą być chronione i bez dowiązań. Nie kieruj konfiguracji na katalog domowy użytkownika, współdzielony upload ani drzewo wdrożeń zapisywalne przez obce podmioty. Kontrola ścieżek nie zastępuje ochrony hosta.

23.3 AWS ACM

Cel importowanego certyfikatu AWS: patrz AWS-LIFECYCLE.md i examples/agent.aws-vault.json. Podmień wszystkie identyfikatory i domeny, zarejestruj instalatora i osobnego obserwatora, jawnie nadaj deployment i zatwierdź cel serwera o takim samym zakresie AwsAcm. Chroń istniejący klucz i referencję; przed uruchomieniem zadań przygotuj rollback komendą aws-adopt. Przykład używa Vault do wystawiania i nie jest szablonem publicznie zaufanej CA. Poświadczenia roli AWS rozwiązuje lokalny SDK.

24 Urzędy publiczne: DNS-01, EAB i DigiCert

Cel ACME może używać walidacji `dns-01` (`AcmeChallengeType`, wymagana dla certyfikatów wildcard) z dostawcą DNS w `AcmeDns` (Route 53, Azure DNS, Google Cloud DNS lub Cloudflare, ograniczonym do jednej strefy) oraz powiązania konta z umową komercyjnego urzędu (`AcmeEabKeyId` i `AcmeEabHmacReference`). Cel `digicert` zamawia lub ponownie wystawia certyfikaty w DigiCert CertCentral w granicach polityki zakupów, a `Clm.Agent digicert-collect` zapisuje migawkę zakupionych certyfikatów do importu w konsoli. Klucze API i HMAC są sekretami klienta rozwiązywanymi lokalnie. Szczegóły: CA-CONNECTORS.md.

24.1 Odzyskiwanie tożsamości agenta wykrywania

Start wykrywania wykonuje wyłącznie lokalną walidację i blokadę. Rotacja i wywołania dostawców działają w cyklu z ponowieniami, więc przejściowa awaria przy starcie nie kończy stale działającego agenta. Nieudane potwierdzenie rotacji oznacza połączenie do odbudowania, nawet gdy następna rotacja nie zgłasza zmian; poprzedni klient jest zwalniany dopiero po poprawnym utworzeniu nowego. Nieudana telemetria nie maskuje błędu cyklu, który można naprawić. Przekroczenia czasu żądań oraz HTTP 408/429/500/502/503/504 są ponawiane po Settings.RetrySeconds; błędy uwierzytelniania i uprawnień, zaufania TLS oraz przerwanie przez użytkownika zatrzymują agenta. `--once` nadal kończy się kodem różnym od zera po nieudanym cyklu. Żadne kontrole zaufania ani zakresu nie są pomijane. Ścieżki stanu i blokady wykrywania odrzucają dowiązania i wymagają ochrony tylko dla właściciela; istniejący, szeroko dostępny stan jest odrzucany, a nie poprawiany przez chmod.

25 Podpisane aktualizacje

Wydania są podpisywane offline: `scripts/package-agent.sh` zapisuje `clm-agent-<rid>.release.json` (wersja, środowisko, nazwa archiwum, rozmiar, SHA-256), a przy `CLM_RELEASE_SIGNING_KEY` także odłączony podpis ECDSA P-384/SHA-384. Klucz utwórz raz na komputerze offline: `openssl ecparam -name secp384r1 -genkey -noout -out release-signing.key` oraz `openssl ec -in release-signing.key -pubout -out release-signing.pub`; opublikuj klucz publiczny i jego odcisk (`openssl pkey -pubin -in release-signing.pub -outform DER | sha256sum`). CLM jedynie udostępnia te pliki w `/downloads/` i nie może sfałszować aktualizacji.

Gdy install-linux.sh otrzyma klucz publiczny, instaluje go jako /etc/clm-agent-update/release-signing.pub (własność root), zapisuje /etc/clm-agent-update/update.json (w tym `"AgentConfigurationFile":"/etc/clm-agent/agent.json"`) i włącza codzienny `clm-agent-update.timer`. Usługa aktualizacji (root) uruchamia `Clm.Agent update /etc/clm-agent-update/update.json`, która:

- pobiera manifest i podpis przez HTTPS i sprawdza je przypiętym kluczem; inny klucz, brak podpisu lub inne środowisko powoduje odmowę;
- niczego nie instaluje, gdy opublikowana wersja jest taka sama lub starsza (brak obniżania wersji);
- pobiera archiwum z certyfikatem klienta i organizacją zarejestrowanego agenta (z `AgentConfigurationFile`), bo serwer udostępnia archiwa tylko uprawnionym żądaniom (zob. „Pobieranie pakietów"), sprawdza rozmiar i SHA-256 względem podpisanego manifestu i rozpakowuje je bez dowiązań, ścieżek bezwzględnych i wpisów `..`;
- uruchamia nowy plik z `--version`, który musi zgłosić podpisaną wersję, atomowo przełącza /opt/clm-agent/current, restartuje clm-discovery.service i czeka na `systemctl is-active`;
- gdy restart lub sprawdzenie zdrowia się nie powiedzie, przywraca `current` na poprzednie wydanie i je restartuje (kod wyjścia 3); poprzednie wydanie jest zawsze zachowywane.

Pola update.json: `ReleaseOrigin` (HTTPS), `Rid`, `ReleaseKeyFile`, `InstallRoot`, `RestartCommand` i opcjonalny `HealthCommand` (bezwzględne pliki wykonywalne ze stałymi argumentami), opcjonalny `TrustRootsFile` dla prywatnego CA, `AgentConfigurationFile` (plik `agent.json` zarejestrowanego agenta, którego certyfikat klienta i organizacja uprawniają do pobrania archiwum), `HealthTimeoutSeconds` (5–900, domyślnie 60) i `KeepReleases` (2–10). Kody wyjścia: 0 zainstalowano lub wersja aktualna, 1 odmowa bez zmian, 3 wycofano. Plikowi update.json zapisanemu przez starszy instalator

brakuje `AgentConfigurationFile`; ten serwer odpowiada wtedy na pobranie archiwum kodem HTTP 401, a aktualizacja niczego nie instaluje. Uruchom `sudo bash upgrade-linux.sh` z nowszego rozpakowanego pakietu, który dodaje tę właściwość (zob. „Aktualizacja w miejscu"), albo samodzielnie dodaj `"AgentConfigurationFile":"/etc/clm-agent/agent.json"` do pliku /etc/clm-agent-update/update.json (właściciel root, bez prawa zapisu dla innych). Podgląd: `journalctl -u clm-agent-update.service`. Samoaktualizacja działa w Linux; w Windows ten sam podpisany manifest sprawdza `upgrade-windows.ps1` (zob. „Aktualizacja w miejscu"). Podpisany ZIP możesz też sprawdzić samodzielnie (PowerShell 7: `[System.Security.Cryptography.ECDsa]::Create()`, `ImportFromPem`, potem `VerifyData` z SHA-384 i `Rfc3279DerSequence`, albo OpenSSL). Katalog zaktualizowanego wydania nadal zgadza się z `SHA256SUMS` pakietu. Podpis kodu wydawcy opisują sekcje „Weryfikacja pakietu" i „Budowanie pakietów".

25.1 Aktualizacja w miejscu

Nowszą wersję instaluje się na zarejestrowanym agencie bez nowego kodu rejestracji; konfiguracja, tożsamość i stan pozostają bez zmian. Gdy krok poniżej wymaga nowszego pakietu na hoście, pobierz jego archiwum przez zalogowaną konsolę (link pobierania w widoku Agenci i cele wdrożenia albo Automatyzacja SSH) i skopiuj je na host, bo archiwa są udostępniane tylko uprawnionym żądaniom (zob. „Pobieranie pakietów"); sumy kontrolne, podpisy i manifesty wydań są publiczne.

- **Linux z włączonymi podpisanymi aktualizacjami** (instalacja z kluczem wydań): codzienny timer aktualizuje automatycznie; `sudo bash upgrade-linux.sh` (w nowszym rozpakowanym pakiecie) sprawdza od razu. Skrypt dodaje `AgentConfigurationFile` do istniejącego update.json, w którym jej brakuje, więc instalacje starszego instalatora znów mogą pobierać archiwa. Pozwala też usłudze wykrywania odczytać sieci interfejsów hosta (`AF_NETLINK`), jeśli usługę zainstalował instalator sprzed wersji 1.0.1.
- **Linux zainstalowany bez klucza wydań**: pobierz od operatora zaufanym kanałem release-signing.pub i jego odcisk, a następnie uruchom `sudo bash upgrade-linux.sh ./release-signing.pub` z nowszego rozpakowanego pakietu. Skrypt przypina klucz, zapisuje /etc/clm-agent-update dokładnie tak jak instalator (w tym `"AgentConfigurationFile":"/etc/clm-agent/agent.json"`, aby aktualizacja przedstawiała certyfikat klienta tego agenta), włącza `clm-agent-update.timer`, wypisuje odcisk klucza i od razu uruchamia `Clm.Agent update`. Klucz inny niż już przypięty jest odrzucany. Potrzebne jest podpisane wydanie (`.release.json.sig`); bez opublikowanego klucza go nie ma i aktualizacja kończy się z tym powodem.
- **Windows, pobieranie przez agenta** (zalecane przy podpisanych wydaniach): w PowerShell administratora uruchom `powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\upgrade-windows.ps1" -Download -PublisherSha256 <PUBLISHER-SHA256> -ReleaseKeyFile .\release-signing.pub` (kolejne aktualizacje mogą pominąć `-ReleaseKeyFile`, klucz jest przypinany przy pierwszym użyciu). Zainstalowany skrypt zleca `Clm.Agent.exe download-release` pobranie `clm-agent-win-x64.release.json`, jego `.sig` i archiwum z adresu agentów z użyciem certyfikatu klienta tego agenta (bez pobierania przez konsolę i kopiowania), sprawdza rozmiar i SHA-256 archiwum względem manifestu, a następnie wykonuje wszystkie kontrole i kroki z następnego punktu. Bez klucza wydań (`-ReleaseKeyFile` lub już przypiętego) `-Download` jest odrzucane. Udana aktualizacja z kluczem wydań rejestruje też codzienne zadanie `CERTYX Agent Update` (zob. krok 4 dla Windows), więc kolejne wydania docierają bez ręcznego uruchamiania.
- **Windows**: archiwum `clm-agent-win-x64.zip` jest udostępniane tylko uprawnionym żądaniom (zob. „Pobieranie pakietów"), a host jest już zarejestrowany, więc kod rejestracji nie ma tu zastosowania: pobierz ZIP przez zalogowaną konsolę (link pobierania w widoku Agenci i cele wdrożenia albo Automatyzacja SSH) i skopiuj go na host. `clm-agent-win-x64.release.json` i `clm-agent-win-x64.release.json.sig` są publiczne; pobierz je bezpośrednio z `/downloads/` adresu dla agentów. Rozpakuj ZIP i w PowerShell administratora w rozpakowanym katalogu uruchom `.\upgrade-windows.ps1 -Archive ..\clm-agent-win-x64.zip -PublisherSha256 <PUBLISHER-SHA256> -ReleaseKeyFile .\release-signing.pub` (klucz zostaje przypięty w `C:\ProgramData\CLM-Agent-Update` przy pierwszym użyciu; kolejne aktualizacje mogą pominąć `-ReleaseKeyFile`). Skrypt ponownie rozpakowuje archiwum do chronionego katalogu programu, sprawdza podpis wydawcy, podpisany `SHA256SUMS` i podpisy Authenticode (jak instalator), a potem `Clm.Agent.exe verify-release` sprawdza podpis manifestu przypiętym kluczem, rozmiar i SHA-256 archiwum oraz to, że wersja jest nowsza. Dopiero wtedy zatrzymuje zadania `CLM Discovery Agent` i `CERTYX SSH Workload`, przenosi zastępowane pliki programu do `C:\ProgramData\CLM-Agent\previous`, kopiuje nowe pliki, przenosi tożsamość starszych instalacji do `identity\` (`Clm.Agent.exe migrate-identity`), uruchamia zadania i wymaga, by `Clm.Agent.exe status` zakończył się powodzeniem w ciągu 30 sekund. W przeciwnym razie przywraca poprzednie pliki, uruchamia je i kończy się kodem 3; odrzucona weryfikacja kończy się kodem 1 bez zmian. Bez opublikowanego klucza wydań skrypt odmawia, chyba że podano `-AllowUnsignedDevelopment` (tylko środowisko deweloperskie: wyłącznie podpis wydawcy i kontrola wersji).

Rotacja tożsamości w Windows przed tą wersją nie mogła zapisać następnej tożsamości, ponieważ katalog programu jest dla LOCAL SERVICE tylko do odczytu: rotacja zatrzymuje się wtedy z komunikatem „cannot write next to its identity". Uruchom `upgrade-windows.ps1` albo jako administrator `Clm.Agent.exe migrate-identity C:\ProgramData\CLM-Agent\agent.json`, a następnie `icacls C:\ProgramData\CLM-Agent\identity /grant *S-1-5-19:(OI)(CI)M`, i rotacja będzie kontynuowana.

26 Kontrola wiersza poleceń

`Clm.Agent check <https-origin>` (przed rejestracją, bez tożsamości) i `Clm.Agent status <config-file>` (po rejestracji, jako konto agenta; w Linux `sudo clm-agent status`) wypisują jeden wiersz `[ OK ]`, `[WARN]` lub `[FAIL]` na krok i kończą się wierszem `Result:`; kod wyjścia 0 oznacza działanie, 1 nieudany krok, 2 błędne użycie. Wypisują nazwy hostów, adresy, podmioty certyfikatów, publiczny odcisk certyfikatu i czasy kontaktu, nigdy kodów, haseł ani kluczy.

Polecenia `Clm.Agent` (`Clm.Agent.exe` w Windows); pierwsze słowo wybiera polecenie:

- `--version`: wypisuje wersję wydania.
- `check <https-origin>` i `status <config-file>`: kontrole opisane powyżej.
- `enroll <https-origin> <identyfikator-organizacji> <plik-kodu> <config-file>`: jednorazowa rejestracja (zob. „Uruchomienie ręczne lub w kontenerze").
- `scan <config-file> [--once]`: wykrywanie sieci; to polecenie uruchamiają clm-discovery.service i zadanie Windows.

- `update <update.json>`: instaluje nowsze podpisane wydanie (Linux, zob. „Podpisane aktualizacje").
- `verify-release <release-signing.pub> <release.json> <release.json.sig> <archive> <rid> [<installed-version>]`: weryfikacja offline pobranego podpisanego wydania (`VERIFIED`, kod 0, albo `REFUSED: <powód>`, kod 1); używana przez `upgrade-windows.ps1`.
- `migrate-identity <config-file>`: przenosi tożsamość Windows z katalogu programu do `identity\` (zob. „Aktualizacja w miejscu").
- `rotate-identity <config-file>`: zleca natychmiastową rotację tożsamości; działająca usługa wykrywania lub obciążenie SSH wymienia klucz i certyfikat przy najbliższym sprawdzeniu tożsamości (zob. „Cofnięcie dostępu i odzyskiwanie").
- `download-release <config-file> <rid> <directory>`: pobiera opublikowany manifest wydania, jego podpis i archiwum z użyciem certyfikatu klienta agenta do istniejącego katalogu; niczemu nie ufa i niczego nie instaluje (używane przez `upgrade-windows.ps1 -Download`).
- `--log-file <path>` (z dowolnym poleceniem): zapisuje komunikaty do pliku logu ze znacznikami czasu, najwyżej 5 MB (plus `<path>.1`); używają go zadania Windows.
- `<config-file> [--once]`: agent cyklu życia certyfikatów (każda lista argumentów, która nie pasuje do innego polecenia).
- `verify-endpoints <config-file> [--once]`: niezależny weryfikator endpointów.
- `verifier add-target <agent-config> <verifier-directory> <target-file>` i `verifier remove-target <verifier-directory> <target-id>`: zapisują konfigurację weryfikatora (używane przez `configure-verifier-linux.sh` i `configure-verifier-windows.ps1`).
- `ssh add-renewal-profile <ssh-directory> <profile-file>`: zapisuje `renewal.json` samodzielnego lokalnego odnawiania (używane przez `enable-renewal`).
- `secret-check <config-file>`: test referencji sekretów klienta (CUSTOMER-VAULTS.pl.md).
- `relay-helper`: wypisuje pomocnika przekaźnika z tego podpisanego agenta do instalacji na hostach, na które wdraża przekaźnik SSH (zob. „Przekaźnik SSH").
- `relay-check <konfiguracja-cyklu-życia> [id-miejsca]`: sprawdza miejsca wdrożenia przekaźnika SSH bez zmiany certyfikatu: przypięcie klucza hosta, uwierzytelnianie, wersję pomocnika, katalog magazynu, usługę nasłuchującą na porcie TLS z proponowanym konektorem i sprawdzenie konfiguracji (kod 0 gotowe, 1 nieudany krok, 2 złe użycie).
- `aws-plan`, `aws-adopt`, `gcp-adopt`, `azure-adopt` i `digicert-collect`: przygotowanie chmury i DigiCert (AWS-LIFECYCLE.md, GCP-LIFECYCLE.md, AZURE-LIFECYCLE.md, CA-CONNECTORS.md).
- `ssh manage|renew|monitor-krl|publish-krl|sync-krl <config-file> [--once]`, `ssh apply-krl <konfiguracja-odbiorcy> <plik-koperty>` oraz lokalne polecenia urzędu `ssh issue|revoke|list|export-krl|deploy-host|deploy-user` (tylko Unix, SSH-LIFECYCLE.md).

Instalator Linux dodaje pomocnika `/usr/local/bin/clm-agent`: `sudo clm-agent status`, `sudo clm-agent check`, `sudo clm-agent log [wiersze]` i `clm-agent version`. W Windows wywołuj `Clm.Agent.exe` z `C:\ProgramData\CLM-Agent` bezpośrednio w PowerShell administratora.

Zła liczba argumentów lub nieznana opcja kończy się komunikatem `Usage:` i kodem 2, zanim agent odczyta jakąkolwiek konfigurację lub utworzy katalog: `secret-check`, `scan <konfiguracja> [--once]`, `enroll`, `Clm.Agent <konfiguracja> [--once]` oraz podpolecenia SSH z własną konfiguracją (`manage`, `renew`, `monitor-krl`, `publish-krl`, `sync-krl`, `apply-krl`). Rejestracja wymaga, aby plik tokenu był zwykłym plikiem dostępnym tylko dla właściciela, bez dowiązań symbolicznych w ścieżce; jest to sprawdzane przed odczytem tokenu i utworzeniem katalogu konfiguracji, a odpowiedzi rejestracji są ograniczone do 64 KiB. Konfiguracja cyklu życia bez wymaganego elementu (adres platformy, certyfikat klienta, katalog stanu, cele z identyfikatorem i domenami, źródło hasła certyfikatu) kończy się błędem `InvalidDataException`. Punkt kontrolny wykrywania jest zapisywany tylko dla właściciela i trwale. Polecenia uruchamiane przez agenta (nginx, przeładowanie usługi, ssh-keyscan, sonda ssh) nie czytają standardowego wejścia agenta; przekaźnik SSH zapisuje zawartość plików i hasła magazynów kluczy tylko na standardowe wejście własnej sesji ssh.

27 Wymagania hosta Windows

Przed instalacją uruchom `powershell -NoProfile -File .\check-windows.ps1` w PowerShell administratora, w rozpakowanym pakiecie. Powstanie `certyx-windows-preflight.json`: wyłącznie system/wersja/architektura, wersja PowerShell, status administratora i dostępność zadań/SSH. Skrypt nie zbiera haseł, tokenów, zawartości certyfikatów, nazw hostów ani użytkowników. Raport wymienia tylko wymagania; nie instaluje agenta ani nie weryfikuje pakietu. Przed uruchomieniem plików pakietu zweryfikuj zatwierdzony skrót i podpis. Instalator wymaga jawnego adresu HTTPS, odrzuca dowiązania tokenu/katalogu instalacji i usuwa lokalną kopię tokenu także przy błędzie rejestracji. Oryginalny plik tokenu pozostaje pod Twoją kontrolą; usuń go po użyciu.