Poświadczenia i bezpieczeństwo w Jenkinsie

Poziom: operacyjny **Dotyczy: Jenkins LTS** **Czas: ~30 min**

Jenkins z natury wykonuje kod z repozytoriów i ma dostęp do poświadczeń infrastruktury — jeśli te dwie rzeczy nie są od siebie odseparowane uprawnieniami, dowolny commit z Jenkinsfile potrafi wykraść wszystko, co kontroler trzyma w magazynie poświadczeń.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
CSRF	Cross-Site Request Forgery: atak podszywający się pod kliknięcie użytkownika na innej stronie
LTS	Long Term Support: wersja z długim wsparciem poprawkami
REST	sposób budowania API przez zwykłe żądania HTTP (GET, POST) zamiast osobnego protokołu

Fiszka

TYLKO ODCZYT

Użycie poświadczenia w kroku pipeline'u przez credentials()

```
environment {
    LOGIN_REJESTRU = credentials('id-poswiadczenia-rejestru')
}

steps {
    sh 'docker login git.zielona.example -u $LOGIN_REJESTRU_USR -p $LOGIN_REJESTRU_PSW'
}
```

Poświadczenie typu „Username with password" po podstawieniu przez `credentials()` rozbija się na dwie zmienne środowiskowe z sufiksami `_USR` i `_PSW` — sama nazwa zmiennej głównej (`LOGIN_REJESTRU`) też jest ustawiona, ale dla tego typu nie jest bezpośrednio użyteczna.

sh = shell: podstawowy interpreter poleceń Linuksa, tu uruchamia pobrany skrypt instalacyjny

TYLKO ODCZYT

Weryfikacja maskowania sekretu w logu builda

```
+ sh 'docker login git.zielona.example -u ****** -p ******'
Login Succeeded
```

Jenkins automatycznie podmienia w logu każde wystąpienie wartości pobranej z `credentials()` na gwiazdki — maskowanie działa na poziomie dosłownego dopasowania tekstu, więc zakodowanie sekretu (np. base64) w skrypcie przed jego wypisaniem omija tę ochronę.

TYLKO ODCZYT

Nadanie uprawnień przez macierz uprawnień

> **Pojęcie: macierz uprawnień (Matrix Authorization Strategy).** Tabela w **Manage Jenkins → Security**, w której dla każdego użytkownika lub grupy zaznaczamy osobno prawo do odczytu, budowania, konfigurowania czy usuwania — daje pełną kontrolę kosztem ręcznego zarządzania wieloma wierszami przy większej liczbie kont.

TYLKO ODCZYT

Nadanie uprawnień przez strategię opartą na rolach

> **Pojęcie: role-based strategy.** Rozszerzenie macierzy uprawnień o wtyczkę Role-based Authorization Strategy — zamiast przypisywać uprawnienia każdemu kontu z osobna, definiujemy role (np. „deweloper", „administrator zespołu") z konkretnym zestawem uprawnień i przypisujemy konta do ról. Skaluje się znacznie lepiej niż płaska macierz, gdy kont są dziesiątki.

Magazyn poświadczeń

> **Pojęcie: magazyn poświadczeń.** Zaszyfrowany magazyn wewnątrz `JENKINS_HOME`, w którym Jenkins trzyma hasła, klucze i tokeny potrzebne buildom, żeby nie musiały być wpisywane wprost w treści Jenkinsfile ani konfiguracji zadania. Klucz szyfrujący magazyn leży w `secrets/master.key` — to jeden z powodów, dla którego kopia `JENKINS_HOME` bez tego pliku jest bezużyteczna do odtworzenia poświadczeń, a sam plik nigdy nie powinien trafić do repozytorium razem z resztą kopii zapasowej bez dodatkowego zabezpieczenia.

Typy poświadczeń

Typ	Zastosowanie
Secret text	pojedynczy token — API key, webhook secret
Username with password	para login/hasło — dostęp do rejestru obrazów, systemu zewnętrznego
SSH Username with private key	klucz prywatny SSH — połączenie z agentem, checkout z repozytorium po SSH
Secret file	cały plik jako sekret — plik `.kubeconfig`, certyfikat, plik z wieloma zmiennymi

Zakresy poświadczeń

> **Pojęcie: zakres (scope) poświadczenia.** Poświadczenie dodane na poziomie **Global** jest widoczne dla wszystkich zadań w całej instalacji Jenkinsa — wygodne, ale oznacza, że każdy, kto ma prawo edytować dowolny Jenkinsfile, może się nim posłużyć. Poświadczenie dodane w zakresie konkretnego folderu jest widoczne wyłącznie dla zadań wewnątrz tego folderu, co pozwala odseparować sekrety jednego zespołu od drugiego bez rozdzielania ich na osobne instalacje Jenkinsa.

Ochrona przed skryptami Groovy

>

> **Pojęcie: script approval.** Pipeline skryptowy i część kroków deklaratywnych pozwalają wywołać dowolny kod Groovy, w tym metody spoza bezpiecznej piaskownicy (np. odczyt pliku systemowego, wywołanie procesu). Jenkins domyślnie blokuje takie wywołania i wymaga ręcznego zatwierdzenia przez administratora w **Manage Jenkins → In-process Script Approval** — dopóki administrator nie zatwierdzi konkretnego wywołania, build kończy się błędem `RejectedAccessException` zamiast po cichu wykonać potencjalnie niebezpieczny kod.

Zatwierdzenie oczekującego skryptu Groovy

```
Manage Jenkins > In-process Script Approval


Signature: method java.io.File delete
Requested by: nazwa_usera
[ Approve ] [ Deny ]
```

Zatwierdzamy pojedynczy podpis metody, a nie cały skrypt — kolejne, inne wywołanie tej samej metody z innego Jenkinsfile nie wymaga ponownej zgody, ale nowe, wcześniej niewidziane wywołanie już tak.

CSRF i bezpieczeństwo agent-to-controller

> **Pojęcie: ochrona CSRF (crumb).** Jenkins domyślnie wymaga do każdego żądania modyfikującego stan (np. wywołania buildu przez REST API) dodatkowego tokenu „crumb", pobieranego osobnym zapytaniem — zabezpiecza to przed wymuszeniem akcji w imieniu zalogowanego administratora przez złośliwą stronę otwartą w tej samej przeglądarce. Wyłączenie tej ochrony bywa kuszące przy integracji z zewnętrznymi systemami niewspierającymi crumba, ale zawęża powierzchnię ataku dokładnie w miejscu, gdzie Jenkins ma pełne uprawnienia administracyjne.

> **Pojęcie: agent-to-controller security.** Domyślnie w nowszych wersjach Jenkinsa agent nie ma pełnego zaufania kontrolera — nie może np. odczytać dowolnego pliku z dysku kontrolera ani wywołać dowolnej wewnętrznej metody, tylko wąski zestaw operacji potrzebnych do normalnej pracy. To ograniczenie ma znaczenie zwłaszcza wtedy, gdy agent wykonuje kod z niezaufanego repozytorium (fork zewnętrznego kontrybutora) — nawet przejęcie samego agenta nie daje automatycznie pełnego dostępu do kontrolera.

Częste błędy

Poświadczenie w zakresie Global tam, gdzie wystarczyłby zakres folderu. Domyślne dodawanie wszystkiego jako Global jest wygodne na start, ale z czasem każdy zespół w instalacji dostaje dostęp do sekretów każdego innego zespołu.

Ręczne wypisywanie sekretu przez `sh 'echo $HASLO'` do debugowania. Maskowanie działa na literalną wartość podstawioną przez `credentials()` — jeśli wartość trafi do zmiennej pośredniej albo zostanie przetworzona (np. zakodowana), maskowanie jej nie rozpozna i sekret trafi do logu jawnym tekstem.

Zatwierdzanie w Script Approval bez zrozumienia, co dana metoda robi. Zatwierdzenie to świadoma decyzja bezpieczeństwa administratora, nie formalność do „odklikania", żeby build przestał się wywalać.

□□ Ćwiczenia

> Zadania sprawdzające.

1. Wyjaśniamy różnicę między zakresem Global a zakresem folderu dla poświadczenia i podajemy sytuację, w której zakres folderu jest wyraźnie lepszym wyborem.
2. Pipeline używa `credentials('token-api')` typu Secret text i loguje wartość zmiennej przez `echo`. Wskazujemy, co się wyświetli w logu builda i dlaczego.
3. Build kończy się błędem `RejectedAccessException` przy wywołaniu metody Groovy spoza piaskownicy. Opisujemy, co administrator musi zrobić, żeby build przeszedł, i czy dotyczy to też innych, przyszłych zadań.
4. Wyjaśniamy jednym zdaniem, dlaczego wyłączenie ochrony CSRF w Jenkinsie jest ryzykowniejsze niż w przeciętnej aplikacji webowej.

Powiązane: Architektura Jenkinsa: kontroler, agenci, workspace · Pipeline: Jenkinsfile, etapy, artefakty · Agenci: dodanie, diagnostyka, utrzymanie

---

Revision #36
Created 2026-09-25 07:39:55 UTC by Tomasz Konieczny
Updated 2026-10-26 14:15:14 UTC by Tomasz Konieczny