Gdzie trzymać token: sekrety agenta poza czatem i poza plikiem

Poziom: srednio **Dotyczy: Windows 11, Linux, BookStack, MCP** **Czas: ~30 min**

Token ma trafić do programu, który go używa, a nie do rozmowy z modelem ani do pliku, który kiedyś zatwierdzimy do repozytorium. Pokazujemy to krok po kroku na przykładzie tokena do BookStacka: konto z najmniejszą rolą, wpis do magazynu poświadczeń, rejestracja serwera w agencie, wariant dla Linuksa i Dockera oraz uczciwa lista tego, czego taki magazyn nie chroni.

Skróty użyte na tej stronie (10)

Skrót	Po angielsku	Po polsku i co znaczy
AES256	Advanced Encryption Standard, 256-bit key	zaawansowany standard szyfrowania z kluczem 256-bitowym
AI	Artificial Intelligence	sztuczna inteligencja: programy, które naśladują część ludzkiego rozumowania i rozmawiają w języku naturalnym
API	Application Programming Interface	interfejs programistyczny aplikacji: sposób, w jaki programy rozmawiają ze sobą
CLI	Command Line Interface	interfejs wiersza poleceń: praca przez wpisywanie komend w terminalu
HTTP	Hypertext Transfer Protocol	protokół przesyłania hipertekstu: język, którym przeglądarki i programy rozmawiają z serwerami WWW
JSON	JavaScript Object Notation	zapis obiektów w stylu JavaScript: tekstowy format danych z klamrami i cudzysłowami
MCP	Model Context Protocol	protokół kontekstu modelu: otwarty standard podłączania narzędzi i danych do modeli AI
OAuth	Open Authorization	otwarta autoryzacja: standard, w którym jedna usługa dostaje ograniczony dostęp do drugiej przez token, bez poznawania hasła
PyPI	Python Package Index	indeks pakietów Pythona: publiczny rejestr bibliotek, z którego pobiera pakiety pip
TPM2	Trusted Platform Module 2.0	moduł zaufanej platformy: układ w komputerze, który przechowuje klucze szyfrujące

Leksykon: nazwy i pojęcia z tej strony (35)

Nazwa	Co to jest
agent	Model z dostępem do narzędzi, który w pętli planuje, wywołuje narzędzia, czyta wyniki i decyduje o kolejnym kroku. Tym różni się od czatu, że jego działania mogą zmieniać stan systemów.
allowlist	Lista dozwolonych: wszystko, czego na niej nie ma, jest zabronione. Jest bezpieczniejsza od listy zakazanych, bo o nowych zagrożeniach nie trzeba pamiętać.
Antigravity CLI	Narzędzie Google do pracy z agentami kodującymi, w wersji uruchamianej z terminala. Konfigurację serwerów MCP trzyma w pliku w profilu użytkownika.
Bearer	Schemat nagłówka HTTP Authorization, w którym po słowie Bearer idzie token. Kto go okaże, zostaje wpuszczony, więc token trzeba chronić jak hasło.
BookStack	Nasza baza wiedzy w przeglądarce, ułożona w półki, książki, rozdziały i strony. Ma API, przez które agent może czytać strony.
Claude	Rodzina modeli firmy Anthropic.
Claude Code	Agent kodujący firmy Anthropic działający w terminalu. Czyta CLAUDE.md, ma system uprawnień i hooki.
czat	Rozmowa z modelem, w której on tylko odpowiada tekstem. Sam niczego nie uruchamia ani nie zmienia, dopóki nie damy mu narzędzi.
Docker i Docker Compose	Docker uruchamia programy w odizolowanych kontenerach. Compose opisuje kilka kontenerów jednym plikiem.
FastMCP i MCPServer	Klasa z pakietu mcp do pisania serwerów MCP w Pythonie. W wersji 1 pakietu nazywa się FastMCP, w wersji 2 MCPServer.
Gemini	Rodzina modeli i czat firmy Google.
Gitea	Własny serwer repozytoriów Git z API. Jego serwer MCP (gitea-mcp) podłączamy do agenta w trybie tylko do odczytu.
guardrail	Mechanizm poza modelem, który ogranicza, co model lub agent może zrobić: filtr, lista dozwolonych narzędzi, limit kroków, zgoda człowieka. Prośba w prompcie guardrailem nie jest.
keyring	Biblioteka Pythona, która zapisuje i odczytuje sekrety w systemowym magazynie. W Windows jest to Menedżer poświadczeń.
log audytowy	Zapis, kto, kiedy i o co poprosił oraz co zrobił program, do późniejszego sprawdzenia.
Menedżer poświadczeń	Wbudowany w Windows magazyn haseł i tokenów, zaszyfrowany dla konta użytkownika. Chroni przed zwykłym przeglądaniem plików i przed innymi kontami, ale nie przed programem uruchomionym na tym samym koncie.
piaskownica	Odgrodzone środowisko (kontener, osobne konto, maszyna wirtualna), w którym agent może coś zepsuć, nie dotykając reszty systemu.
pip i PyPI	pip to instalator pakietów Pythona, a PyPI to publiczne składowisko tych pakietów.
poświadczenia systemd	Mechanizm systemd w Linuksie: usługa dostaje sekret jako plik dostępny tylko dla niej (LoadCredential), a plik na dysku może być zaszyfrowany (systemd-creds).

Nazwa	Co to jest
PowerShell	Powłoka i język skryptowy Windows. W nim wykonujemy polecenia z naszych stron dla Windowsa.
Python	Język programowania, w którym piszemy klienty, serwery MCP i skrypty.
rotacja tokenu	Regularna wymiana tokenu lub hasła na nowe. Dzięki temu wyciek ma ograniczony czas życia.
sekret i poświadczenie	Informacja, która otwiera dostęp: hasło, klucz API, token. Trzymamy ją w magazynie sekretów, a nie w czacie, pliku konfiguracyjnym ani repozytorium.
serwer MCP	Program, który udostępnia agentowi narzędzia i dane w jednolity sposób, według protokołu MCP. Działa lokalnie jako podproces albo zdalnie przez HTTP.
śmiertelna trójka	Nazwa Simona Willisona (czerwiec 2025) dla trzech zdolności, które razem otwierają drogę do kradzieży danych: dostępu do danych prywatnych, kontaktu z niezaufaną treścią i możliwości wysyłania danych na zewnątrz. Zabezpieczenie polega na odjęciu jednej z nich.
stdio	Transport MCP, w którym klient uruchamia serwer jako własny podproces i rozmawia z nim przez standardowe wejście i wyjście. Poświadczenia serwer dostaje ze środowiska, a nie z sieci.
systemd	Menedżer usług w Linuksie: uruchamia i pilnuje usługi, ma też mechanizm przekazywania sekretów.
test akceptacyjny	Sprawdzenie opisane z góry, które faza musi przejść, żeby uznać ją za skończoną.
token	Słowo o dwóch znaczeniach. W modelu to kawałek tekstu (słowo, jego część albo znak), na które model dzieli wszystko, co czyta i pisze; od liczby tokenów zależą szybkość i koszt. W dostępie to ciąg znaków, który zastępuje hasło i otwiera drzwi do usługi.
tylko do odczytu	Tryb, w którym narzędzie może czytać, ale nie może zmieniać. Każde nowe podłączenie zaczynamy od niego.
uwierzytelnianie i autoryzacja	Uwierzytelnianie sprawdza, kim jesteś. Autoryzacja sprawdza, co ci wolno.
wywołanie narzędzia	Prośba modelu o uruchomienie funkcji, na przykład odczytu pliku albo zapytania do BookStacka. Model tylko prosi, a wykonuje program, który może odmówić.
zakres tokenu	Lista uprawnień przypisana tokenowi, czyli co wolno nim zrobić, na przykład tylko odczyt repozytorium.
Zielona	Fikcyjna firma, w której dzieją się nasze przykłady; opisuje ją strona „Świat Zielonej”. Tomek jest młodszym administratorem, Ola programistką, pani Halina główną księgową.
zmienna środowiskowa	Nazwana wartość przekazywana procesowi przy starcie. Tak serwer MCP typu stdio dostaje token, ale nie jest to tajemnica przed innymi programami tego samego konta.

□□ ZMIENIA SYSTEM

Wpisanie tokena do magazynu poświadczeń (Windows)

Token wpisujemy w ukrytym polu programu, nie w linii poleceń, więc nie zostaje w historii powłoki. Zapisuje go biblioteka `keyring` w Menedżerze poświadczeń Windows pod nazwą usługi `bookstack-ro`.

```
python -m pip install keyring
python -c "import getpass, keyring; keyring.set_password('bookstack-ro', 'token',
getpass.getpass('Token id:sekret (nie będzie widoczny): '))"
```

pip = Pip Installs Packages: menedżer pakietów Pythona · **keyring** = biblioteka Pythona do zapisu i odczytu sekretów w magazynie poświadczeń systemu · **getpass** = get password: moduł Pythona, który wczytuje tekst bez wyświetlania go na ekranie · **python -c** = uruchomienie kodu podanego w cudzysłowie, bez osobnego pliku

TYLKO ODCZYT

Sprawdzenie, że wpis istnieje, bez wypisywania wartości

Polecenie wypisuje tylko słowo „jest” albo „brak”, więc można je bezpiecznie uruchomić na oczach innych osób.

```
python -c "import keyring; print('jest' if keyring.get_password('bookstack-ro', 'token')
else 'brak')"
```

get_password = funkcja biblioteki keyring, która odczytuje wpis z magazynu poświadczeń po nazwie usługi i użytkownika

OSTROŻNIE · NIEODWRACALNE

Unieważnienie tokena w BookStacku

Po usunięciu tokena w profilu konta nie da się go przywrócić, a każdy program, który go używał, dostanie błąd 401. Robimy to od razu po każdym podejrzeniu wycieku i dopiero potem wystawiamy nowy token.

Zanim zaczniemy

Potrzebujemy działającego BookStacka, w którym możemy utworzyć rolę i użytkownika, Pythona 3.12 z biblioteką `keyring` i serwera MCP z następnej strony, Własny serwer MCP w Pythonie: BookStack tylko do odczytu. Całość działa na komputerze, na którym pracuje agent, i dotyczy konta Windows, na którym agent jest uruchamiany. Magazyn poświadczeń jest osobny dla każdego

konta, więc token wpisany jako jedno konto jest niewidoczny dla drugiego.

Zasady, według których to robimy, zebraliśmy na stronie Token w czacie: opowieść o jednym wklejeniu: sekret nie trafia do rozmowy ani do repozytorium, ma najmniejszą potrzebną rolę i da się go wymienić bez żalu.

> **Z ŻYCIA WZIĘTE**
>
> Po wtorkowym incydencie Tomek dostał zadanie: przejść całą procedurę od początku do końca na swoim komputerze i zapisać, co było niejasne. Zauważył, że token „jest tylko do odczytu” brzmiał dla niego uspokajająco, dopóki nie zobaczył, że rola konta serwisowego może czytać całą bazę wiedzy, razem z procedurami awaryjnymi. Dopisaliśmy więc do kroku pierwszego ograniczenie roli do dwóch książek, których agent faktycznie potrzebuje.

Krok 1: konto serwisowe i rola tylko do odczytu

Agent nie dostaje tokena żadnej osoby. Zakładamy w BookStacku osobnego użytkownika, na przykład `agent-odczyt`, z adresem w domenie `zielona.example`, i przypisujemy mu jedną, specjalnie przygotowaną rolę. Dzięki temu w dzienniku audytu widać, że zmiany albo odczyty pochodzą od agenta, a nie od Tomka.

Rola ma dwie rzeczy: systemowe uprawnienie do korzystania z API (bez niego token zwraca błąd 403) i prawo do wyświetlania treści. Prawa edycji, tworzenia i usuwania zostawiamy puste. Prawo do wyświetlania nadajemy na poziomie wybranych książek, a nie globalnie, żeby konto widziało tylko to, co ma widzieć.

Token tworzymy w sekcji tokenów API konta serwisowego. Podajemy nazwę, która mówi, do czego służy, na przykład `agent-odczyt-komputer-toma`, i datę wygaśnięcia za kilka miesięcy. BookStack pokazuje parę identyfikator i sekret tylko raz, więc przepisujemy ją prosto do ukrytego pola z kroku drugiego, a nie do notatnika.

> **Dlaczego dwie warstwy ograniczeń.** Rola w BookStacku ogranicza to, co token może zrobić po stronie usługi. Lista książek w zmiennej `BOOKSTACK_KSIAZKI`, którą czyta nasz serwer, ogranicza to, co agent dostanie z tego, co token mógłby zobaczyć. Gdy jedno ograniczenie zawiedzie przez pomyłkę w konfiguracji, drugie nadal działa.

Krok 2: wpisanie tokena bez czatu i bez pliku

Uruchamiamy pierwszą z fiszek powyżej w terminalu zwykłego użytkownika, nie jako administrator. Polecenie pyta o token w ukrytym polu: nic nie pojawia się na ekranie, a naciśnięcie Enter zapisuje wpis. Biblioteka `keyring` na Windowsie korzysta z Menedżera poświadczeń i zapisuje wpis jako ogólne poświadczenie o nazwie `bookstack-ro` i nazwie użytkownika `token`.

Wpis można obejrzeć w Panelu sterowania, w Menedżerze poświadczeń, w poświadczeniach systemu Windows, w sekcji poświadczeń ogólnych. Zobaczymy tam nazwę wpisu i możliwość jego usunięcia, a samo hasło zobaczy tylko ktoś, kto kliknie pokaż i zna hasło do konta Windows. Drugą fiszką sprawdzamy, że wpis istnieje, bez wypisywania wartości.

Krok 3: rejestracja serwera w agencie

Do konfiguracji agenta wpisujemy wyłącznie polecenie uruchomienia serwera i ustawienia, które nie są tajne. W Claude Code służy do tego `claude mcp add`; opcje `--env`, `--transport` i `--scope` stoją przed nazwą serwera, a po podwójnym myślniku zaczyna się polecenie uruchamiające serwer. Opcja `--env` nie może stać bezpośrednio przed nazwą serwera, bo wtedy nazwa zostałaby odczytana jako kolejna para, więc między nimi wstawiamy `--transport stdio`.

```
claude mcp add --scope user --env BOOKSTACK_URL=https://wiedza.zielona.example --env
BOOKSTACK_KSIAZKI=19,829 --transport stdio bookstack-ro -- python C:\AI\mcp\bookstack_ro.py
claude mcp list
```

Polecenie `claude mcp list` pokazuje stan serwera: połączony, wymaga uwierzytelnienia, błąd połączenia albo oczekuje zatwierdzenia. W Antigravity CLI ten sam wpis trafia do pliku `.gemini/config/mcp_config.json` w profilu użytkownika, a konfiguracja wyłącznie dla jednego projektu do pliku `.agents/mcp_config.json` w katalogu projektu.

```
{
  "mcpServers": {
    "bookstack-ro": {
      "command": "python",
      "args": ["C:\\AI\\mcp\\bookstack_ro.py"],
      "env": {
```

```
        "BOOKSTACK_URL": "https://wiedza.zielona.example",
        "BOOKSTACK_KSIAZKI": "19,829"
      }
    }
  }
}
```

Ani w poleceniu, ani w pliku nie ma tokena. Serwer czyta go przy starcie z Menedżera poświadczeń, a jeśli go tam nie znajdzie, kończy pracę z komunikatem, co zrobić.

Krok 4: wariant dla Linuksa, gdy serwer działa jako usługa

Na Linuksie ten sam wzorzec ma dwa warianty. Gdy agent działa w terminalu użytkownika z pulpitem, biblioteka `keyring` korzysta z usługi sekretów środowiska graficznego, więc wystarczy ten sam program i to samo polecenie. Gdy ten sam wzorzec stosujemy do własnej usługi uruchamianej przez systemd, na przykład na `app01`, właściwym miejscem na sekret są poświadczenia systemd, a funkcja `wczytaj_token` w naszym serwerze obsługuje tę drogę jako pierwszą.

Poświadczenie zapisujemy zaszyfrowane, wpisując token w ukrytym polu powłoki, a nie w argumencie polecenia. `printf` jest poleceniem wbudowanym powłoki, więc token nie pojawia się na liście procesów.

```
sudo install -d -m 700 /etc/zielona
read -rs -p 'Token id:sekret: ' T && printf '%s' "$T" | sudo systemd-creds encrypt --
name=bookstack-ro - /etc/zielona/bookstack-ro.cred; unset T
```

W pliku usługi wskazujemy zaszyfrowane poświadczenie. Systemd odszyfrowuje je tuż przed startem usługi i udostępnia jako plik w katalogu wskazanym przez zmienną `CREDENTIALS_DIRECTORY`, a nasz serwer czyta ten plik jako pierwszą z trzech możliwości.

```
[Service]
LoadCredentialEncrypted=bookstack-ro:/etc/zielona/bookstack-ro.cred
ExecStart=/usr/bin/python3 /opt/mcp/bookstack_ro.py
```

Według dokumentacji systemd dostęp do poświadczenia ma tylko użytkownik usługi, dane nie są przekazywane procesom potomnym przez środowisko, a katalog z poświadczeniami leży w pamięci, która nie trafia do pliku wymiany, o ile pozwalają na to uprawnienia. Szyfrowanie używa AES256-

GCM, a klucz pochodzi z modułu TPM2, z pliku `/var/lib/systemd/credential.secret` albo z obu naraz, więc poświadczenia nie da się przygotować na jednej maszynie i odszyfrować na innej [1].

Krok 5: wariant dla Dockera

W Docker Compose sekret montowany jest jako plik w katalogu `/run/secrets`, a jego nazwa to nazwa sekretu z pliku Compose. Nasz serwer szuka pliku `bookstack-ro` w katalogu ze zmiennej `CREDENTIALS_DIRECTORY`, więc w kontenerze wystarczy nazwać sekret `bookstack-ro` i ustawić tę zmienną na `/run/secrets`. Plik z samym tokenem trzymamy poza repozytorium, a w pliku Compose stoi tylko ścieżka do niego.

Gdy serwer jest zdalny: zmienne i pomocnik nagłówków

Nie każdy serwer MCP możemy napisać sami. Gdy łączymy się z cudzym serwerem przez HTTP i potrzebny jest statyczny token, Claude Code pozwala podstawić zmienną środowiskową do nagłówka, a plik `.mcp.json` zawiera wtedy tylko jej nazwę.

```
{
  "mcpServers": {
    "gitea": {
      "type": "http",
      "url": "https://mcp.zielona.example/mcp",
      "headers": { "Authorization": "Bearer ${GITEA_TOKEN}" }
    }
  }
}
```

Rozwijanie zmiennych działa w polach `command`, `args`, `env`, `url` i `headers`. Gdy zmiennej nie ma, a nie podano wartości domyślnej, serwer się ładuje, Claude Code wypisuje ostrzeżenie i używa tekstu `${GITEA_TOKEN}` dosłownie, więc połączenie kończy się odrzuceniem, a nie pustym nagłówkiem [2].

Zmienna musi jednak być ustawiona w środowisku procesu agenta, a to oznacza, że agent z powłoką może ją odczytać. Lepszy wybór daje pole `headersHelper`, które zamiast zmiennej wskazuje skrypt. Skrypt wypisuje nagłówki jako JSON i dostaje na to 10 sekund; uruchamia się przy każdym połączeniu, jego wynik nie jest zapamiętywany, a po odpowiedzi 401 albo 403 Claude Code uruchamia go ponownie i próbuje raz jeszcze. Jeśli skrypt pochodzi z pliku `.mcp.json` projektu, działa dopiero po zatwierdzeniu zaufania do katalogu, a Claude Code usuwa z jego środowiska

zmienne o nazwach zawierających `TOKEN`, `SECRET`, `PASSWORD`, `KEY` i `AUTH` [2].

```
# pobierz_naglowki.py: wypisuje nagłówki jako JSON, token bierze z magazynu poświadczeń
import json
import keyring

token = keyring.get_password("gitea-ro", "token")
print(json.dumps({"Authorization": f"Bearer {token}"}))
```

Czego taki magazyn nie chroni

Menedżer poświadczeń chroni token przed zwykłym przeglądaniem plików i przed innymi kontami Windows. Nie chroni go przed innym procesem tego samego użytkownika, bo ten może zawołać ten sam interfejs systemu, z którego korzysta biblioteka `keyring`. Agent z nieograniczoną powłoką, uruchomiony na tym koncie, mógłby więc wykonać polecenie podobne do fiszki odczytu, tylko z wypisaniem wartości zamiast słowa „jest”.

Prawdziwa obrona składa się więc z kilku rzeczy naraz. Token ma najmniejszą rolę i krótką ważność, więc jego wyciek kosztuje mało. Agent działa na osobnym koncie albo w piaskownicy, więc nie sięga do poświadczeń człowieka.

Agent nie ma nieograniczonej powłoki: nie dajemy modelowi bezpośredniego dostępu do `PowerShell.exe`, `cmd.exe`, `docker.exe`, `ssh.exe` ani do konta administratora, o czym mówi strona Projektowanie guardraili: jak ograniczyć, co model i agent mogą zrobić. Serwer po swojej stronie ma listę dozwolonych książek i dziennik wywołań.

> **Magazyn nie zastępuje ograniczeń.** Jeśli agent może uruchamiać dowolne polecenia na koncie, na którym leży token, to token nie jest chroniony przed agentem. Dlatego w naszym planie uprawnienia agenta i miejsce przechowywania sekretów projektujemy razem, a nie osobno.

Rotacja i unieważnienie

Rotacja jest zaplanowaną wymianą sekretu, zanim ktokolwiek go zobaczy, a unieważnienie reakcją na to, że ktoś go zobaczył. Obie robimy w tej samej kolejności, żeby usługa była bez tokena możliwie krótko.

1. Wystawiamy w BookStacku nowy token dla tego samego konta serwisowego.

2. Wpisujemy go do magazynu tym samym poleceniem z kroku drugiego; `set_password` zastępuje poprzednią wartość.
3. Restartujemy agenta i sprawdzamy, że serwer działa na nowym tokenie.
4. Usuwamy stary token w BookStacku.

Przy unieważnieniu po wycieku zmieniamy kolejność: najpierw usuwamy stary token, potem wystawiamy nowy, bo ważniejsze jest zamknięcie dziury niż ciągłość pracy.

Problemy i rozwiązania

Objaw	Przyczyna	Rozwiązanie
serwer kończy się komunikatem „Brak tokena”	wpis w magazynie założono na innym koncie Windows albo pod inną nazwą	uruchamiamy drugą fiszkę na tym samym koncie, na którym działa agent, i sprawdzamy nazwę usługi
błąd 401	token wpisano z białymi znakami albo bez formatu `id:sekret`	wpisujemy ponownie, bez spacji i znaków nowej linii
błąd 403	rola konta serwisowego nie ma uprawnienia do API	dodajemy do roli systemowe uprawnienie do korzystania z API
serwer działa w terminalu, ale nie z agenta	agent uruchamia serwer z okrojonym środowiskiem, bez ścieżki do właściwego Pythona	podajemy pełną ścieżkę do interpretera w konfiguracji
`CERTIFICATE_VERIFY_FAILED`	brak zmiennych firmowego urzędu certyfikacji w środowisku serwera	dopisujemy `SSL_CERT_FILE` w sekcji `env`
`NoKeyringError` na serwerze Linuksa bez pulpitu	brak usługi sekretów	używamy poświadczeń systemd z kroku czwartego

Test akceptacyjny

Pytamy agenta o stronę o kopiach zapasowych z naszej bazy wiedzy i obserwujemy, że wywołuje narzędzia `szukaj` i `czytaj_strone`, a odpowiedź zawiera treść strony. Prosimy o stronę z książki spoza listy i oczekujemy odpowiedzi „ODMOWA”. W pliku dziennika serwera widzimy wiersze z nazwą narzędzia i liczbą trafień, a w żadnym z nich nie ma tokena ani treści stron.

Potem usuwamy wpis z Menedżera poświadczeń i uruchamiamy serwer ponownie: powinien zakończyć pracę z komunikatem, a nie próbować dalej bez tokena. Na koniec przeszukujemy repozytorium i historię rozmów skanerem sekretów, na przykład `gitleaks`, żeby potwierdzić, że token nie pojawił się w żadnym pliku. Test uznajemy za zaliczony, gdy wszystkie pięć obserwacji zgadza się z opisem.

Ćwiczenia

1. Dlaczego token wpisujemy przez `getpass` w programie, a nie podajemy jako argument polecenia?

Odpowiedź

Argument polecenia widać na liście procesów i zostaje w historii powłoki. Ukryte pole programu przyjmuje wartość bezpośrednio z klawiatury, nie zapisuje jej w historii i nie wyświetla na ekranie.

2. Czy wpis w Menedżerze poświadczeń chroni token przed agentem z nieograniczoną powłoką, uruchomionym na tym samym koncie?

Odpowiedź

Nie. Inny proces tego samego użytkownika może odczytać wpis przez ten sam interfejs systemu. Ochronę zapewniają dopiero: token o najmniejszej roli, osobne konto albo piaskownica dla agenta, brak nieograniczonej powłoki i lista dozwolonych książek po stronie serwera.

3. Kiedy w kolejności rotacji stary token usuwamy jako pierwszy, a kiedy jako ostatni?

Odpowiedź

Przy zwykłej, zaplanowanej rotacji usuwamy go ostatni, po sprawdzeniu, że nowy działa, żeby usługa nie była bez tokena. Po wycieku usuwamy go pierwszy, bo zamknięcie dziury jest ważniejsze niż ciągłość pracy.

Źródła i dalsze lektury

Przypisy

[1] Poświadczenia usługi, `LoadCredential=`, `LoadCredentialEncrypted=`, `$CREDENTIALS_DIRECTORY`, dostęp tylko dla użytkownika usługi, szyfrowanie `systemd-creds`: systemd, Credentials.

[2] Rozwijanie zmiennych w `.mcp.json`, `headersHelper`, `claude mcp add`, stany serwerów, zatwierdzanie serwerów projektu: Claude Code, MCP.

[3] Magazyn poświadczeń Windows w bibliotece `keyring` (wpis ogólny o nazwie usługi, nazwa użytkownika zapisana w polu użytkownika): kod modułu `keyring/backends/Windows.py` w wydaniu 25.7.0, keyring na PyPI. Tokeny API w BookStacku, nagłówek `Authorization: Token id:sekret` i uprawnienie do API: sekcja „Getting Started” w dokumentacji API, dostępnej w każdej instancji pod adresem `/api/docs`.

Lektury

O tym, jak wybrać sposób połączenia, mówi Jak agent łączy się z serwerem MCP: dwa transporty i miejsce na sekret, a o ograniczaniu zasięgu szkód po stronie agenta Śmiertelna trójka: dane, obca treść i wyjście na zewnątrz.

Nota metodyczna: czego nie wiemy

Wpisów w Menedżerze poświadczeń Windows nie obejrzeliśmy w interfejsie, bo budowaliśmy serwer w środowisku bez Windowsa. Nazwę wpisu i pole użytkownika opisujemy na podstawie kodu biblioteki `keyring`, a nie zrzutu ekranu. Polecenie `systemd-creds encrypt` z wejściem ze standardowego wejścia opisujemy według dokumentacji, nie uruchomiliśmy go na `app01`, więc przed wdrożeniem warto je wypróbować na maszynie testowej.

Nie sprawdzaliśmy, jak Antigravity CLI traktuje zmienne w sekcji `env`, i dlatego przy nim polecamy serwer, który sam czyta sekret. Nie wiemy też, gdzie Claude Code zapisuje na Windowsie tokeny OAuth wystawione dla serwerów zdalnych: dokumentacja mówi tylko, że są przechowywane bezpiecznie i odświeżane automatycznie.

Revision #3
Created 2026-10-20 08:59:10 UTC by Tomasz Konieczny
Updated 2026-10-20 09:05:24 UTC by Tomasz Konieczny