Engineer

SSH

зашифрована оболонка на іншій машині

Що це таке

SSH (Secure Shell) — це мережевий протокол для входу на інший комп'ютер і виконання на ньому команд через мережу, якій не можна довіряти, причому трафік шифрується, а обидві сторони доводять, хто вони, ще до початку роботи. Під цією назвою ховаються три речі: сам протокол, який IETF опублікувала в січні 2006 року як набір RFC на чолі з RFC 4251; OpenSSH — реалізація, яку підтримує проєкт OpenBSD і на якій побудовано приклади в цій статті; і команда `ssh`, що спілкується з серверним демоном `sshd`.

RFC 4251 описує три рівні поверх одного з'єднання, зазвичай TCP на порт 22 — номер, який протоколу призначила IANA. Транспортний рівень проводить обмін ключами, доводить особу сервера його ключем хоста, а далі шифрує все, що проходить, і перевіряє його цілісність. Рівень автентифікації користувача працює всередині цього тунелю й визначає, хто такий клієнт: за відкритим ключем, підтримка якого обов'язкова для кожної реалізації, за паролем, який лишається необов'язковим, або через довіру на рівні хоста. Рівень з'єднань ділить тунель на канали, тож один вхід може одночасно нести інтерактивну оболонку, окрему команду, передавання файлів через SFTP і кілька переспрямованих портів.

SSH замінив telnet і rlogin, які передавали сеанси й паролі мережею відкритим текстом. Сьогодні використовують версію протоколу 2.

Застосуйте поняття до рішення

Застосуйте «Що таке SSH?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

На сервері SSH — це вхідні двері. До машини в дата-центрі чи хмарі рідко хтось може підійти з клавіатурою, тож адміністрування, розгортання, копіювання файлів і автоматизоване налаштування приходять через SSH, а той, хто входить із достатніми правами, фактично володіє машиною.

Автентифікація за відкритим ключем робить SSH водночас зручним і надійним. Користувач тримає закритий ключ на своїй машині й кладе відповідний відкритий ключ на сервер, типово у `~/.ssh/authorized_keys`. Під час входу клієнт підписує дані, прив’язані саме до цього сеансу, тож сам ключ ніколи не передається мережею. Викраденим паролем можна скористатися звідусіль, а сервер, що приймає паролі, може атакувати підбором кожен, хто дістався до його порту, і автоматизовані інструменти саме це й роблять. OpenSSH типово приймає паролі, тож вимкнути їх — рішення, яке доводиться ухвалювати для кожного сервера.

Довіра потрібна й у зворотний бік: клієнт має знати, що потрапив саме на той сервер. Кожен сервер має ключ хоста, і RFC 4251 пропонує два способи йому довіряти: локальний список на клієнті, що пов’язує кожне ім’я хоста з його ключем, або центр сертифікації, який підписує ключі хостів, тож клієнтові достатньо мати лише ключ центру. Більшість користується першим у найвільнішій формі, яку RFC допускає заради зручності: прийняти ключ без перевірки під час першого з’єднання, зберегти його в `~/.ssh/known_hosts` і звіряти з ним під час кожного наступного. Це зазвичай називають довірою під час першого використання (trust on first use). Перший вхід захищений лише настільки, наскільки захищена мережа, якою він іде, тому OpenSSH питає дозволу, перш ніж зберегти новий ключ, і категорично відмовляє, коли відомий хост показує інший.

SSH — це ще й транспорт, на якому будуються інші інструменти. Git надсилає й отримує зміни через нього, `scp` і `sftp` копіюють ним файли, Ansible дістається через нього до своїх хостів, а переспрямування портів дає змогу з ноутбука дістатися до бази даних, що слухає лише loopback-інтерфейс сервера, не відкриваючи її в інтернет.

Як це виглядає на практиці

Власний хост компанії приймає інтерактивні входи лише через SSH, і його захист перевіряється у три етапи щоразу, коли запускається роль автоматизації, що відповідає за конфігурацію. Ці перевірки з’явилися у вересні 2026 року, коли звіт про доступ порівняв налаштування, які остаточно визначив демон, із тими, що записувала роль, і два з них розійшлися: типові значення дистрибутива стояли активними рядками, які читалися раніше за файл ролі, і sshd залишав перше значення. Нова назва файлу, що сортується перед файлом постачальника, закрила цю прогалину. Тепер новий файл перевіряють на синтаксис до встановлення, усю конфігурацію — після того, як вона на місці, а остаточні налаштування демона порівнюють із 24 очікуваними директивами, тож значення, яке програло ранішому рядку, зупиняє запуск. Дві директиви навмисно пропущено, бо чинна версія демона обидві вже не підтримує.

Місяцем раніше, у серпні 2026 року, перевірка виявила три захисти від перебору паролів, які ніколи не працювали, від самого дня, коли хост зібрали. Механізм блокування стежив за портом 22, хоча порт 1986, на який демона переселили ще з першим налаштуванням, лишався поза його увагою. Обмеження частоти з’єднань у брандмауері стояло після ширшого правила, яке дозволяло той самий трафік, а оскільки спрацьовує перше правило, що підходить, до обмеження так і не дійшов жоден пакет, хоча у виводі стану воно й далі значилося. Дія блокування покладалася на утиліту пакетного фільтра, яку менеджер пакетів лише рекомендує, тож механізм помічав нападників, а потім завершувався помилкою щоразу, коли намагався когось заблокувати. Після перевірки лишилися два сталі правила: кожен засіб безпеки має підкріплюватися перевіркою, що доводить його дію, а правило брандмауера підтверджують за його місцем у списку, а не лише за наявністю.

Власна git-форджа компанії використовує SSH як транспорт, а не як спосіб входу. Її обрали, бо нею керують через SSH і вона не має вебпанелі, яку довелося б захищати. Анонімний доступ і доступ без ключа відхиляються, а кожен репозиторій приватний. Її SSH-сервер приймає з’єднання тільки на loopback, порт 23231, і брандмауер не відкриває для нього додаткового порту: клієнт ззовні спершу підключається до хоста-переходу й лише звідти до самого сервера.

Де виникають проблеми

Рядок захисту стоїть у файлі, а сервер його ігнорує. Для більшості ключових слів `sshd` залишає перше прочитане значення, а `Include` розгортає свої файли в лексикографічному порядку, тож коли основний файл першим підтягує каталог додаткових файлів, файл постачальника, що сортується раніше, перемагає ваш, і обидва файли при цьому виглядають правильно. Пошук у конфігурації доводить лише те, що рядок записано. Питайте демона: `sshd -t` перевіряє синтаксис, а `sshd -T` виводить налаштування, якими він справді користуватиметься.

```
/etc/ssh/sshd_config.d/10-vendor.conf:     PasswordAuthentication yes
/etc/ssh/sshd_config.d/50-hardening.conf:  PasswordAuthentication no

$ sudo sshd -T | grep -i '^passwordauthentication'
passwordauthentication yes
```

Змінений ключ хоста видаляють і приймають. Коли відомий сервер показує інший ключ хоста, `ssh` виводить попередження й відмовляється з'єднуватися. Рефлекторна реакція — `ssh-keygen -R` для імені хоста, повторне з'єднання і набране yes, а це все, що потрібно для атаки «людина посередині». Спершу звірте новий відбиток іншим каналом, наприклад командою `ssh-keygen -lf` для відкритого ключа хоста, запущеною з консолі сервера. У скриптах `StrictHostKeyChecking no` небезпечніше, ніж здається: воно не лише мовчки зберігає нові ключі, а й дозволяє, з певними обмеженнями, з'єднання з хостами, чиї ключі змінилися. Значення `accept-new` зберігає невідомі хости й далі відмовляє, коли ключ змінився.

Вхід за паролем лишається відкритим для всього інтернету. `PasswordAuthentication` типово має значення yes, як і автентифікація keyboard-interactive, що може попросити той самий пароль іншим шляхом. Щойно в кожного користувача є ключ, встановіть обом значення no. `PermitRootLogin` уже типово має значення `prohibit-password`, яке не пускає root за паролем, але пускає за ключем. Перенесення демона з порту 22 змінює те, де він слухає, а не те, хто може підбирати пароль, і кожен засіб захисту, що згадує порт, має переїхати разом із ним, як показує механізм блокування вище.

Переспрямування агента позичає ваші ключі серверу. Коли увімкнено `ForwardAgent` або використано `ssh -A`, будь-хто, хто здатен обійти права доступу до файлів на віддаленому хості, зокрема root, може через ваш агент

автентифікуватися деінде від вашого імені, доки ви підключені. Скопіювати ключ він не може, але підписувати ним може. Щоб дістатися до машини за бастіоном, використовуйте хост-перехід: `ssh -J bastion.example.com app.internal` прокладає TCP-з'єднання через бастіон і домовляється про сеанс із цільовою машиною наскрізно, тож бастіон пересилає байти, яких не може прочитати, а ключ лишається на вашому ноутбуці.

Що почитати далі

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — три рівні, дві моделі довіри до ключів хоста й розділ про безпеку.
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — обмін ключами, перевірка ключа хоста й рядок версії, який кожна сторона надсилає першим.
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — як на рівні протоколу працюють входи за відкритим ключем, паролем і через довіру хоста.
- sshd_config(5) — усі ключові слова сервера з типовими значеннями й правило, за яким діє перше отримане значення.
- ssh_config(5) — бік клієнта, зокрема StrictHostKeyChecking, ProxyJump і ForwardAgent.
- sshd(8) — тестові режими `-t` і `-T` та формати файлів authorized_keys і known_hosts.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке SSH?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/ssh/