Тема 5. Керування сервером з командного рядка

Для підключення до сервера або іншого мережевого обладнання, що керується операційною системою, з метою налаштування найчастіше використовується інтерфейс командного рядка (консоль).

Інтерфейс командного рядка (command-line interface, CLI) – різновид текстового інтерфейсу користувача та комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням текстових рядків (команд). Також відомий під назвою консоль. Інтерфейс командного рядка може бути протиставлений системам керування програмою на основі меню чи різних реалізацій графічного інтерфейсу. Формат виводу інформації в інтерфейсі командного рядка не регламентується; зазвичай це простий текстовий вивід, але може бути й графічним, звуковим виводом тощо.

Командна оболонка в UNIX є інтерфейсом командного рядка в Unix-подібних операційних системах, тобто виконує команди, які подає користувач, або які читаються з файлів. Такі файли з командами оболонки називаються сценаріями (скриптами, програмами) оболонки. Ці сценарії не компілюються, а інтерпретуються оболонкою. Це означає, що оболонка прочитує сценарій від початку до кінця, рядок за рядком, шукаючи зазначені там команди й виконуючи їх; на відміну від цього підходу, компілятор перетворює цілу програму до вигляду, придатного до виконання машиною – потім файл з таким кодом можна використати в сценарії оболонки. Характерна особливість мови оболонки – багато операцій, які в традиційних мовах програмування є вбудованими, виконуються з допомогою виклику зовнішніх програм.

Для встановлення захищеного з’єднання через інтерфейс командного рядка використовуються протоколи Telnet та SSH.

Telnet (TELetype NETwork) – мережевий протокол для реалізації інтерфейсу командного рядка по мережі (у сучасній формі – за допомогою транспорту TCP). Назву «telnet» мають також деякі утиліти, що реалізують клієнтську частину протоколу.

Призначення протоколу TELNET у наданні достатньо спільного, двонаправленого, восьмибітового байт-орієнтованого засобу зв'язку. Його основне завдання полягає в тому, щоб дозволити термінальним пристроям і термінальним процесам взаємодіяти один з одним. Передбачається, що цей протокол може бути використаний для зв'язку виду термінал-термінал («зв'язування») або для зв'язку процес-процес («розподілені обчислення»).

Хоча у сесії Telnet виділяють клієнтську і серверну сторону, протокол насправді повністю симетричний. Після встановлення транспортного з'єднання (як правило, TCP) обидва його кінця грають роль «мережевих віртуальних терміналів» (Network Virtual Terminal, NVT), які обмінюються двома типами даних:

- прикладними даними (тобто даними, які йдуть від користувача до текстового додатка на стороні сервера і назад);
- командами протоколу Telnet, окремим випадком яких є опції, службовці для з'ясування можливостей і переваг сторін.

Хоча Telnet-сесії, що виконується по TCP, властивий повний дуплекс, NVT повинен розглядатися як напівдуплексний пристрій, що працює за замовчуванням у буферизованому рядковому режимі.

Прикладні дані проходять через протокол без змін. Дані можуть зазнавати змін (наприклад може бути скинутий старший біт) у разі, якщо на вхід одного терміналу надійшли дані, допустимість приймання яких не була підтверджена, тобто на виході другого віртуального терміналу ми бачимо саме те, що було введено на вхід першого. З точки зору протоколу дані представляють просто послідовність байтів (октетів), за замовчуванням належать набору ASCII, але при включеній опції Binary – будь-яких. Хоча були запропоновані розширення для ідентифікації набору символів Telnet Charset Option, але на практиці ними не користуються. Всі значення октетів прикладних даних крім \ 377 (десяткове 255) передаються з транспорту як є. Октет \ 377 передається послідовністю \ 377 \ 377 із двох октетів. Це пов'язано з тим, що октет \ 377 використовується на транспортному рівні для кодування опцій.

Протокол надає за замовчуванням мінімальну функціональність, яка розширюється за рахунок опцій. Принцип обумовлених опцій вимагає проводити переговори при включенні кожної з опцій. Одна сторона ініціює запит, а інша сторона може або прийняти, або відкинути пропозицію. Якщо запит приймається, то опція негайно вступає в силу. Опції описані окремо від протоколу як такого, і їх підтримка програмним забезпеченням довільна. Клієнту протоколу (мережевому терміналу) пропонується відкидати запити на включення непідтримуваних і невідомих опцій.

У протоколі не передбачено використання ні шифрування, ні перевірки достовірності даних. Тому він вразливий для будь-якого виду атак, до яких вразливий його транспорт, тобто протокол TCP. Для функціональності віддаленого доступу до системи в наш час застосовується мережевий протокол SSH (особливо його версія 2), основною причиною створення якого були питання безпеки. Так що варто мати на увазі, що сесія Telnet дуже беззахисна, якщо тільки не здійснюється в повністю контрольованої мережі або з застосуванням захисту на мережевому рівні (різні реалізації віртуальних приватних мереж).

SSH (Secure SHell – «безпечна оболонка») – мережевий протокол рівня застосунків, що дозволяє проводити віддалене управління комп'ютером і тунелювання TCP-з'єднань (наприклад, для передачі файлів). Схожий за функціональністю з протоколом Telnet, проте шифрує весь трафік, в тому числі і паролі, що передаються.

Криптографічний захист протоколу SSH не фіксований, можливий вибір різних алгоритмів шифрування. Клієнти і сервери, що підтримують цей протокол, доступні для різних платформ. Крім того, протокол дозволяє не тільки використовувати безпечний віддалений shell на машині, але і туннелювати графічний інтерфейс – X Tunnelling (тільки для Unix-подібних ОС або застосунків, що використовують графічний інтерфейс X Window System). Так само ssh здатний передавати через безпечний канал (Port Forwarding) будь-який

інший мережевий протокол, забезпечуючи (при належній конфігурації) можливість безпечної пересилки не тільки X-інтерфейсу, але і, наприклад, звуку.

Підтримка SSH реалізована у всіх UNIX системах, і на більшості з них в числі стандартних утиліт присутні клієнт і сервер ssh. Існує низка реалізацій SSH-клієнтів і для не-UNIX ОС. Велику популярність протокол отримав після широкого розвитку сніферів, як альтернативне небезпечному телнету рішення для управління важливими вузлами.

Зараз відомо дві гілки версій – 1 і 2. Проте гілка 1 зупинена, оскільки наприкінці 90-х у ній було знайдено багато вразливостей, деякі з яких досі накладають серйозні обмеження на її використання, тому перспективною (такою, що розвивається) і найбезпечнішою є версія 2.

Перша версія протоколу, SSH-1, була розроблена в 1995 році дослідником Тату Ілоненом з Технологічного університету Гельсінкі, Фінляндія. SSH-1 був написаний для забезпечення більшої конфіденційності, ніж протоколи rlogin, telnet і rsh. У 1996 році була розроблена безпечніша версія протоколу, SSH-2, несумісна з SSH-1. Протокол набув ще більшої популярності, і до 2000 року у нього було близько двох мільйонів користувачів. В даний час під терміном «SSH» зазвичай мається на увазі саме SSH-2, тому що перша версія протоколу з огляду істотних недоліків зараз практично не застосовується.

У 2006 році протокол був затверджений робочою групою IETF як Інтернет-стандарт.

Проте, в деяких країнах (Франція, Росія, Ірак і Пакистан) до сих пір потрібен спеціальний дозвіл у відповідних структурах для використання певних методів шифрування, включаючи SSH. Див закон Російської Федерації «Про федеральних органах урядового зв'язку і інформації» (закон втратив силу з 1 липня 2003 року у зв'язку з прийняттям федерального закону від 30.06.2003 № 86-ФЗ).

Поширені дві реалізації SSH: пропріетарна комерційна та безкоштовна вільна. Вільна реалізація називається OpenSSH. До 2006 року 80 % комп'ютерів мережі Інтернет використовувало саме OpenSSH. Пропріетарна реалізація

розробляється організацією SSH Inc., Вона безкоштовна для некомерційного використання. Ці реалізації містять практично однаковий набір команд.

Протокол SSH-2, на відміну від протоколу telnet, стійкий до атак прослуховування трафіку («сніфінг»). Протокол SSH-2 також стійкий до атак шляхом приєднання посередині (англ. session hijacking) — неможливо включитися у вже встановлену сесію або перехопити її.

Для запобігання атак «людина посередині» при підключенні до хосту, ключ якого ще не відомий клієнту, клієнтське ПО показує користувачеві «зліпок ключа» (key fingerprint). Рекомендується ретельно перевіряти показуваний клієнтським ПО «зліпок ключа» (key fingerprint) зі зліпком ключа сервера, бажано отриманим по надійним каналах зв'язку або особисто.

Підтримка SSH реалізована у всіх UNIX-подібних системах, і на більшості з них в числі стандартних утиліт присутні клієнт і сервер ssh. Існує безліч реалізацій SSH-клієнтів і для не-UNIX ОС. Велику популярність протокол отримав після широкого розвитку аналізаторів трафіку і способів порушення роботи локальних мереж, як альтернативне небезпечному протоколу Telnet рішення для управління важливими вузлами.

Для роботи по SSH потрібен SSH-сервер і SSH-клієнт. Сервер прослуховує з'єднання від клієнтських машин і при встановленні зв'язку виробляє аутентифікацію, після чого починає обслуговування клієнта. Клієнт використовується для входу на віддалену машину і виконання команд.

Для з'єднання сервер і клієнт повинні створити пари ключів – відкритих і закритих – і обмінятися відкритими ключами. Зазвичай використовується також і пароль.

SSH це протокол, який може бути використаний для багатьох додатків на різних платформах, включаючи самі Unix варіантів (Linux, BSD, включаючи від Apple OS X, і Solaris), а також Microsoft Windows. Деякі програми можуть зажадати функції, які доступні тільки або сумісним з конкретними клієнтами SSH або серверів. Наприклад, з використанням протоколу SSH для реалізації VPN можна, але в даний час тільки з OpenSSH сервер і клієнт реалізації.

SSH-тунель — це тунель, який створюється за допомогою SSH-з'єднання і використовується для шифрування тунельованих даних. Використовується для того, щоб убезпечити передачу даних в Інтернеті (аналогічне призначення має IPsec). Особливість полягає в тому, що незашифрований трафік будь-якого протоколу шифрується на одному кінці SSH-з'єднання і розшифровується на іншому.

Практична реалізація може виконуватися декількома способами:

- Створенням Socks-проксі для програм, які не вміють працювати через SSH-тунель, але можуть працювати через Socks-проксі
- Використанням додатків, які вміють працювати через SSH-тунель.
- Створенням VPN-тунелю, підходить практично для будь-яких додатків.

Якщо програма працює з одним певним сервером, можна налаштувати SSH-клієнт таким чином, щоб він пропускав через SSH-тунель TCP-з'єднання, що приходять на певний TCP-порт машини, на якій запущений SSH-клієнт.

SSH – це протокол сеансового рівня. SSH-сервер зазвичай прослуховує з'єднання на TCP-порту 22. Специфікація протоколу SSH-2 міститься в RFC 4251. Для аутентифікації сервера в SSH використовується протокол аутентифікації сторін на основі алгоритмів електронно-цифрового підпису RSA або DSA. Для аутентифікації клієнта також може використовуватися ЕЦП RSA або DSA, але допускається також аутентифікація за допомогою пароля (режим зворотної сумісності з Telnet) і навіть ip-адреси хоста (режим зворотної сумісності з rlogin). Аутентифікація за паролем найбільш поширена, вона безпечна, тому що пароль передається по зашифрованому віртуального каналу. Аутентифікація по ip-адресою небезпечна, цю можливість найчастіше відключають. Для створення спільного секрету (сеансового ключа) використовується алгоритм Діффі – Хеллмана (DH). Для шифрування переданих даних використовується симетричне шифрування, алгоритми AES, Blowfish або 3DES. Цілісність переданих даних перевіряється за допомогою CRC32 в SSH1 або HMAC-SHA1/HMAC-MD5 в SSH2.