B4B33PSY: Počítačové systémy

Lekce 08. Bezpečný vzdálený přístup

Petr Štěpán

stepan@fel.cvut.cz

1. listopadu 2025

Motivace

- Poznat jakým způsobem spolu počítače komunikují
- Porozumět architektuře a protokolu SSH
- Naučit se bezpečnou autentizaci (klíče, agent)
- Praktické techniky: tunelování, SCP, SSHFS

Historie a motivace

- Před SSH: Telnet, rlogin — text bez šifrování
- Potřeba důvěrnosti, integrity a autentizace
- SSH jako nástroj pro zabezpečený shell, tunely a přenos souborů

Telnet

Telnet:

- je aplikační protokol typu klient-server, který poskytuje přístup k virtuálním terminálům vzdálených systémů v lokálních sítích nebo na internetu.
- jedná se o protokol pro obousměrnou komunikaci.
- hlavním cílem je propojit terminálová zařízení a terminálově orientované procesy.
- v podstatě se jedná přesměrování standardního vstupu a výstupu přes internet do jiného počítače.

Termmín Telnet odkazuje na dvě věci:

- protokol, který určuje, jak mají dvě strany komunikovat
  - protokol je specifikovaný jako RFC 854 - Network Working Group – Request for Comments 854
- program, který implementuje tento protokol jako službu na blízkém a vzdáleném počítači.

Telnet

Telnet:

- uživatelská data se přenášejí v 8bitovém orientovaném datovém připojení přes TCP - Transmission Control Protocol
  - podrobnosti o TCP se dozvíte v předmětu Počítačové sítě PSIB ve 2. semestru
  - důležitou vlastností je, že se jedná o spojované a potvrzované zasílání dat mezi dvěma počítači
    - data se nemohou ztratit
    - data nemohou předbíhat, je zaručeno pořadí dat
- Telnet přenáší všechny informace, včetně uživatelských jmen a hesel, v prostém textu
  - Informace jsou viditelné pro všechny po cestě

Telnet

- Praktický příklad připojení na webový server

```
telnet example.com 80
```

- připojení na webový server a získání obsahu webové stránky

Získání webové stránky:

- zaslání zprávy `GET /index.html HTTP/1.0\n\n`

Úvod do kryptografie v SSH

- SSH kombinuje **symetrické** a **asymetrické** šifrování.
- Cílem je:
  - Zajistit **důvěrnost** přenášených dat
    - Nikdo kromě vysílající a přijímající strany data nemůže dekódovat
  - Ověřit **identitu** obou stran.
    - Ověřit, že mluvím se správnou stranou, že svoje data neposílám někomu jinému
  - Zabránit **manipulaci** s přenosem
    - Odolnost vůči tzv. útoku man-in-middle
    - Někdo uprostřed, kdo na první pohled není vidět, ale přijímá vysílaná data a přeposílá je cílovému adresátovi, přičemž může data číst

Symetrické šifrování

Co je **symetrické šifrování**:

- Používá **jeden klíč** pro šifrování i dešifrování.
- Rychlé a efektivní pro velké objemy dat.

Jednoduchým symetrickým kódováním je Ceasarova šifra

- posun písmen abecedy o přesně zvolený počet písmen
- klíč je počet o kolik písmen se abeceda posune
- stejný klíč pro šifrování i dešifrování
- příklad proudové šifry, kóduje se znak po znaku, bajt po bajtu

Příklad kódovací a dekódovací tabulky pro posun klíč 5:

A	B	C	D	E	...	V	W	X	Y	Z
F	G	H	I	J	...	A	B	C	D	E

Symetrické šifrování - bloková šifra

Složitějším příkladem symetrického kódování je Ceasarova šifra s heslem nazývaná Vigenèrova šifra:

- posun písmen abecedy se mění podle pozice písmene v textu
  - pod kovaný text napíšeme opakovaně heslo
  - písmeno hesla určuje o kolik se posouvá abeceda, A-0, B-1, C-2, ...
- klíč je slovo nebo řetězec libovolné délky
- stejný klíč se použije pro šifrování i dešifrování
- příklad blokové šifry, délka hesla určuje velikost bloku

Příklad kódovací a dekódovací tabulky pro klíč PSY:

text	P	S	Y	J	E	F	A	J	N
klíč	P	S	Y	P	S	Y	P	S	Y
kód	E	K	W	Y	W	D	P	B	L

Symetrické šifrování

- Nejznámější používané blokové symetrické šifry:
  - AES, DES, Triple DES
- Nejznámější proudové symetrické šifry:
  - ChaCha20
    - moderní algoritmus využívající sofitikovaný generátor náhodných čísel
    - zjednodušeně přičítá náhodně generovaná čísla ke kódovanému textu
- Symetrické šifry byly náchilné k útoku KPA - Known Plaintext Attack
  - Známe originál textu i jeho zakódovanou hodnotu
  - Šifry se snaží, aby je nešlo prolomit jinak, než vyzkoušením všech klíčů
  - Moderní šifry používají 256 bitový klíč, tedy museli bychom vyzkoušet 2^{256} různých klíčů

Symetrické šifrování

Symetrické šifry jsou velmi jednoduché a rychle spočítatelné:

- kódování zápisů na disk v reálném čase i pro velmi velké soubory

Základní problém:

- **Jak** zajistit bezpečně stejný klíč pro obě strany?
  - Zaslat ho přes internet nelze bez šifrování
  - Jeden veřejně dostupný klíč pro šifrování klíče by nic neřešil

Řešení – Asymetrické kódování

Asymetrické šifrování – princip

Základní princip asymetrického šifrování:

- Používá **dva klíče**:
  - **Veřejný klíč (public)** — sdílený s ostatními.
  - **Soukromý klíč (private)** — uchovávaný v tajnosti.
- Pro kódování se použije veřejný klíč a pouze držitel soukromého klíče je schopen zprávu dekódovat.
  - Pokud chci někomu poslat tajnou zprávu (třeba klíč pro symetrické kódování) použiji jeho veřejný klíč
  - Pro vzájemně utajenou komunikaci potřebují obě strany svoji dvoji veřený–soukromý klíč.
  - Veřejné klíče si pošlou nezašifrovaně veřejným kanálem
  - Každý zakóduje svoji zprávu veřejným klíčem příjemce
- Umožňuje:
  - Ověřit identitu serveru a klienta.
  - Bezpečně vyměnit symetrický klíč.

Jednosměrné funkce

- Pro generování dvojice veřejný – soukromý klíč se používají tzv. jednosměrné funkce
  - Funkce $f(x)$ je jednosměrná, pokud je velmi obtížné spočítat pro konrétní hodnotu $f(a)$ jaká je hodnota a.
  - Zjednodušeně: Potom náhodné číslo a je privátní klíč a $f(a)$ je klíč veřejný
- Nevýhody asymetrického šifrování:
  - Správné volby funkce f a soukromého klíče a jsou výpočetně hodně náročné

RSA - výměna klíče

RSA (zkratka pro Rivest–Shamir–Adleman)

- je šifra s veřejným klíčem založená na faktorizaci - rozkladu na prvočísla.
- vytvořena již v roce 1977 a byla tehdy prvním známým algoritmem, který umožňoval zároveň vytvářet elektronický podpis (jako např. DSA) a zároveň i šifrovat.
- Nejrozšířenější využití RSA šifry je ve webových prohlížečích pro HTTPS (protokol SSL/TLS pro zabezpečený přístup k webovým stránkám).
- Šifra RSA čelila v minulosti několika bezpečnostním problémům, které způsobily nutnost prodloužení šifrovacích klíčů, doporučuje se použít více než 3072 bitů.
- Jejím nástupcem pro webové digitální certifikáty je šifra ECDSA.

Ověření totožnosti

Ověření totožnosti, lze použít vlastnosti dvojice klíčů:

- kódování privátním klíčem vytvoří obdobu podpisu
- každý může ověřit správnost tak, že dekódováním veřejným klíčem
- tím je možné podepsat dokumenty, případně i autorizovat ssh připojení

Asymetrické šifry v SSH

Nejčastější algoritmy:

- RSA — tradiční, bezpečný při dlouhých klíčích (≥ 3072 bitů).
- ECDSA — efektivní, založený na eliptických křivkách.
- Ed25519 — moderní, rychlý, bezpečný, doporučený.
- Použití:
  - Autentizace klienta/serveru.
  - Výměna klíčů (Diffie–Hellman, ECDH, Curve25519).
- Uvedené algoritmy mají rozdíly v rychlosti a odolnosti vůči kvantovým útokům.

SSH-1 — první generace

- Vznik: 1995 (Tatu Ylönen)
- Základní princip: zabezpečení vzdáleného shellu šifrováním.
- Slabiny:
  - Kryptografické chyby (špatná výměna klíčů, integrita).
  - Neoddělené vrstvy → problém při rozšiřování.
  - Zastaralé algoritmy (DES, MD5).
  - Dnes se doporučuje **nepoužívat** — většina implementací jej zcela odstranila.

SSH-2 — moderní standard

- Definován v RFC 4251–4256 (2006)
- Vylepšení oproti SSH-1:
- Oddělené vrstvy (transport, auth, connection)
- Bezpečnější výměna klíčů (Diffie–Hellman, ECDH)
- Silné šifry (AES, ChaCha20) a HMAC pro integritu
- Možnost více relací na jedno spojení
- Podpora moderních metod autentizace (klíče, certifikáty, GSSAPI)
- Dnes **de facto standard** všech implementací (OpenSSH, Dropbear, libssh…)

Základní přehled protokolu SSH

SSH-2:

- protokol řeší základní nedostatek šifrování při komunikaci mezi dvěmi počítači
- Tři vrstvy: Transport, Authentication, Connection
  - **Transport**: zabezpečený kanál, navazuje šifrované spojení, ověřuje server a provádí výměnu klíčů
  - **Authentication**: ověřuje klienta buď heslem, nebo klíčem
  - **Connection**: vytváří kanály pro přenos dat (shell, soubory, tcp-forwarding)
    - i více kanálů současně - tzv. multiplexing

Praktická ukázka

Co potřebujete:

- na straně serveru
  - program sshd - secure shell demon
    - program, který přijímá spojení na domluveném portu
    - obecně je k ssh připojení určen port 22
    - pro zmenšení šance na útok zvenčí můžete použít jiné číslo
      - pozor na firewally a další síťová zabezpečení, která mohou komunikaci ne vybraném portu blokovat
- na straně klienta
  - program ssh
    - program naváže komunikaci se zadaným serverem na defaultním, nebo zadaném portu

Základní připojení

- Příkaz: `ssh user@host` nebo `ssh host` a pak se jako user použije aktuální uživatelské jméno
  - Port: defaultně 22 — přepínač `-p` pro jiný port
  - Podrobnější výpisy: `-v`, `-vvv` pro ladění
- První připojení: kontrola otisku - fingerprint host key
  - Při prvním připojení dostanete hlášku:

```
petr@lenovo:~$ ssh stepan@postel.felk.cvut.cz
The authenticity of host 'postel.felk.cvut.cz (147.32.87.190)'
can't be established.
ECDSA key fingerprint is SHA256:z4VO4A6g76nrNlz/
DUsCRPpTtaIkzPpS8O7dyTH2J5o.
Are you sure you want to continue connecting (yes/no/
[fingerprint])? yes
Warning: Permanently added 'postel.felk.cvut.cz' (ECDSA) to the
list of known hosts.
```

Základní připojení

- Všechny potvrzené otisky se ukládají do souboru known-hosts v adresáři .ssh
- ssh-keygen -R IP.adresa
- ssh-keygen -R hostname
- nebo smazat celý soubor known-hosts

```
ssh-keygen -H -F postel.felk.cvut.cz
# Host postel.felk.cvut.cz found: line 19
|1|b07c8v9fFL39FH8V5L5tZStBSE4=|fGEse+uaeF0URy+y2xiscW2M3sA=
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAo9BBGCRi1ZteNC
```

Základní připojení

- Po ověření veřejného klíče sshd programu na počítači postel.felk.cvut.cz musíte autorizovat svoje přihlášení
  - Autorizace je možná pomocí hesla – Vaše ČVUT heslo (nedoporučeno, ale jediná možnost před nastavením klíče)
  - nebo pomocí Vaší dvojice veřejného a soukromého klíče (doporučeno)
- Autorizace pomocí hesla má slabinu v tom, že útočník může zkoušet kombinace běžně užívaných hesel a tím prolomit Vaše heslo
  - Navíc je autorizace heslem nutná při každém přihlášení
    - pokud bude Váš počítač napadnut virem, který bude sledovat zadaný text přes klávesnici, tak virus přečte Vaše heslo a odešle ho útočníkovi

Co je tedy nutné udělat pro přihlášení klíčem?

Generování klíčů (prakticky)

- Generování klíčů programem `ssh-keygen`
  - pod Windows využijte buď WSL, tedy Ubuntu pod Windows, nebo program PuTTYgen.exe
- Parametry programu `ssh-keygen`:
  - typ algoritmu -t
    - rsa - defaultní algoritmus
      - pro RSA algoritmus je defaultně nastavena velikost pouze 2048 bitů, což není dostatečné, pokud chcete použít RSA algoritmus, pak doporučujeme kombinovat ho s přepínačem -b 4096, který nastaví velikost klíče na 4096 bitů
    - ed25519 - využití eliptických křivek, moderní rychlý algoritmus, stačí defaultně nastavený počet bitů
- Při generování klíče Vás program vyzve
  - k zadání jména souboru a adresáře kam se klíč uloží
  - k zadání hesla pro jeho zašifrování v rámci Vašeho počítače
    - pokud si myslíte, že nikdo nepronikne do Vašeho počítače a nebude si moci Váš soukromý klíč zkopírovat, heslo zadávat nemusíte

Generování klíčů (prakticky)

Příklad generování klíče s eliptickou křivkou:

```
ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/petr/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/petr/.ssh/id_ed25519
Your public key has been saved in /home/petr/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:qsQio/AlOCAMK7cmyWcaceAgngV1XwUm6X650pKRk9w petr@lenovo
The key's randomart image is:
+--[ED25519 256]--+
| ... . ..+o.     |
|  . . ..+        |
|+. .  ..         |
|B.+    .         |
|=*..  o S .      |
|=+oo   O E       |
|Oo=o+ . * .      |
|oB== . + o       |
|... .   o        |
+----[SHA256]-----+
```

Generování klíčů (prakticky)

Generování klíčů vytvořilo dva soubory:

```
petr@lenovo:~/.ssh$ ls -l
total 44
-rw------- 1 petr petr   399 lis  6 21:20 id_ed25519
-rw-r--r-- 1 petr petr    93 lis  6 21:20 id_ed25519.pub
```

- Pro použití klíče systém vyžaduje, aby přístup k soukormému klíči (soubor id_ed25519) neměl nikdo jiný, než vlastník souboru
  - To může být problémem ve WSL pod Windows, kdy klíč musí být uložen v souborovém systému, který umožní tato práva správně nastavit
- Přístup k veřejnému klíči není omezen, protože ten je veřejný a bude veřejně sdílen se všemi

Generování klíčů (prakticky)

Co je v souborech uloženo?

V souboru id_ed25519.pub je veřejný klíč zakódován textem se znaky anglické abecedy, čísel a běžných symbolů

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINmdYuQJZD1en5H8uNT8XmpL1gXboYSBNHsHgRt6J3Wg
petr@lenovo
```

V souboru id_ed25519 je soukromý klíč, zakódován opět textem se znaky anglické abecedy, čísel a běžných symbolů (tento soukromý klíč byl pozměněn)

```
-----BEGIN OPENSSH PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
-----END OPENSSH PRIVATE KEY-----
```

Nahrání veřejného klíče na server

Nyní pro přihlášení bez hesla na počítač potřebujeme nahrát vytvořený veřejný klíč do počítače, kam se chceme přihlásit

- to můžeme udělat dvěma způsoby:
  - jednodušší postup využívá příkaz `ssh-copy-id user@host`
    - program se zkusí přihlásit všemi klíči v adresáři `.ssh`, zjistí, které klíče neuspěly a po ssh přihlášení na daný počítač, zkopíruje všechny veřejné klíče do souboru .ssh/authorized_keys
    - při zadání -i cesta/soubor se instaluje pouze zadaný klíč
  - složitější způsob:
    - ručně přidat obsah veřejného klíče do souboru `~/.ssh/authorized_keys`
    - pokud tento soubor neexistuje je nutné ho vytvořit se správnými přístupovými právy
      - `chmod 700 ~/.ssh` a `chmod 600 authorized_keys`

SSH agent a správa klíčů

- Pokud jsme si pro soukromý klíč zvolili při generování heslo, pak při každém jeho použití, budeme vyzváni k zadání hesla
  - to je nepraktické, protože klíč se může využívat velmi často, např. každý git pull a git push příkaz
  - můžeme využít program `ssh-agent`, který si uloží odemčené heslo
    - uložení hesla je jen dočasné, pouze v paměti, takže nehrozí jeho přečtení žádným programem
    - při restartu počítače zadáme heslo pouze jednou, pak si ho již ssh-agent pamatuje
    - pro vložení hesla do ssh-agenta použijeme program `ssh-add`
      - program nám s přepínačem `ssh-add -l` zobrazí veřejné části klíčů, vložených do ssh-agenta

SSH forwarding

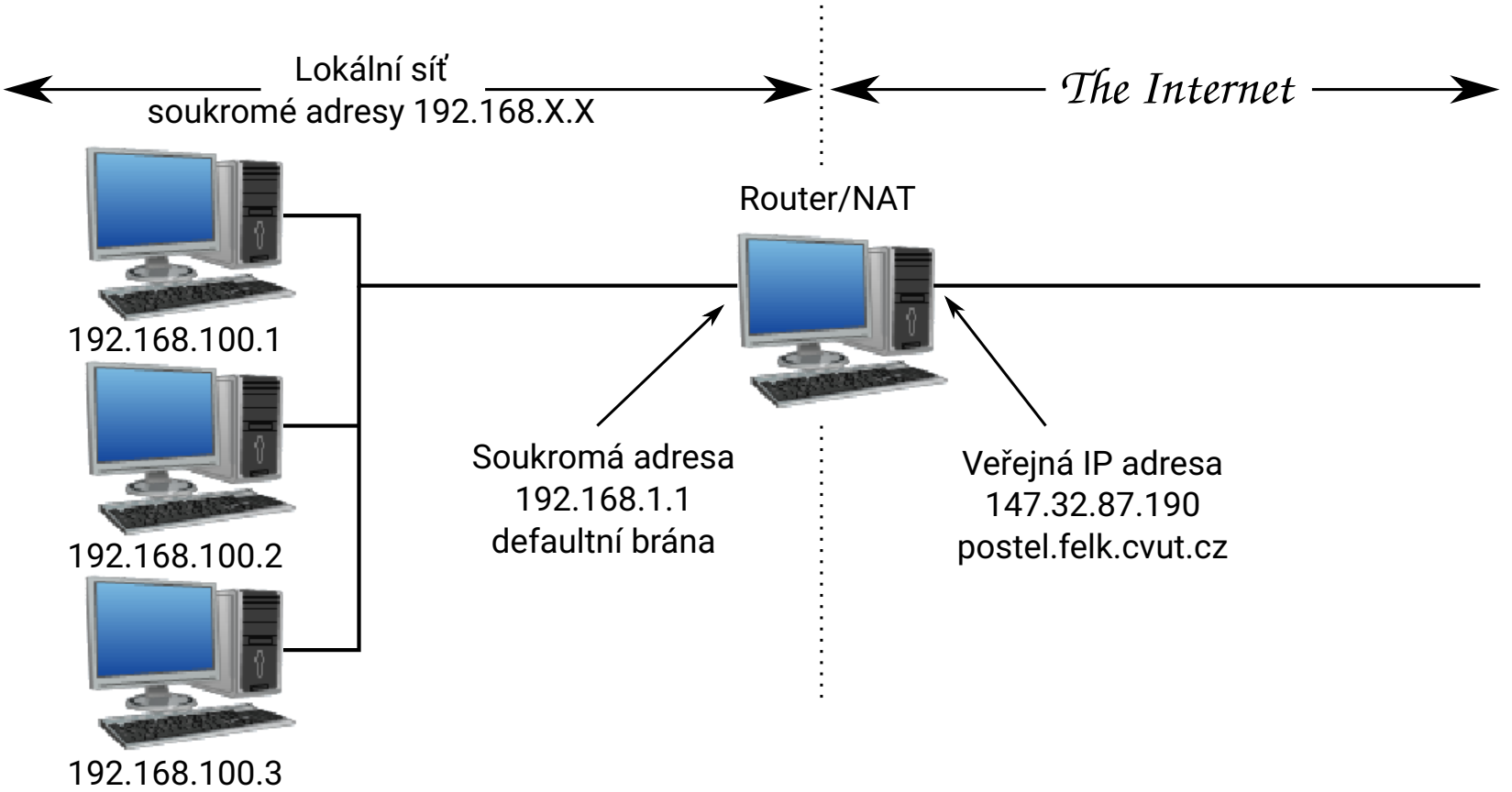
- Pokud chcete využívat svůj klíč i na počítači, kam jste se přihlásili přes ssh, můžete svoje klíče “přeposlat” - forward
  - přeposlání klíčů se nastaví příznakem -A při přihlašování, tedy

```
ssh -A stepan@postel.felk.cvut.cz
```

ProxyJump

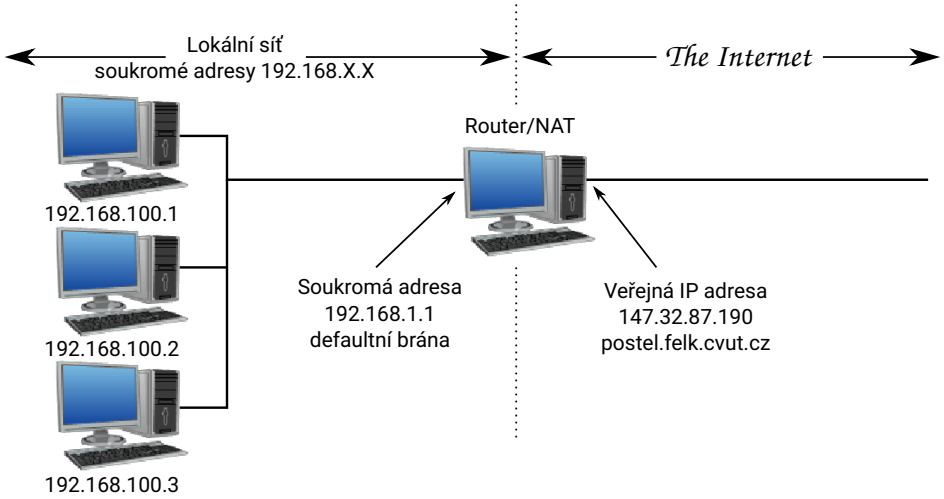
ProxyJump – připojování do lokální sítě

Co je to vlastně lokální síť?



ProxyJump

- Soukromé adresy nejsou viditelné z internetu
  - zvenku není možné se přímo připojit na počítač v lokální síti
- Nejdřív je nutné se připojit na router, který má veřejnou adresu
  - v našem případě počítač postel.felk.cvut.cz, který má IP adresu 147.32.87.190
  - z tohoto počítače již jsou dostupné lokální počítače v lokální síti



ProxyJump

ProxyJump nám pomůže vytvořit přímé spojení na lokální počítač přes jiný počítač

- `ssh -o 'ProxyJump=stepan@postel.felk.cvut.cz' apo@192.168.100.1`
  - Provede nejdřív ssh spojení na počítač postel.felk.cvut.cz s uživatelským jménem stepan
  - Z tohoto počítače provede ssh připojení na počítač v lokální síti s adresou 192.168.100.1 jako uživatel apo
  - Toto připojení bude užitečné při práci v předmětu APO - Architektury počítačů, kdy se budete moci připojit z domova na počítač v lokální síti

Konfigurace ssh klienta

- Pokud se budete často připojovat na nějaký počítač, můžete se parametry ssh připojení nastavit v souboru `~/.ssh/config`:
  - v souboru si pojmenujete seznam připojení a k nim specifická nastavení, použitá pro jednotlivá připojení
- Příklad:

```
Host lab23
  HostName 192.168.100.1
  User stepan
  IdentityFile ~/.ssh/id_ed25519
  Port 2222
  ForwardAgent no
  ProxyJump stepan@postel.felk.cvut.cz
```

- pak již stačí napsat `ssh lab23`
  - provede se: `ssh -p 2222 -i ~/.ssh/id_ed25519 -o 'ProxyJump=stepan@postel.felk.cvut.cz' stepan@192.168.100.1`

Bezpečný přenos souborů

Prvním nástrojem pro přenos souborů byl systém ftp - File Transfer Protocol

- první verze z roku 1971, revize 1980
- první implementace používaly pouze řádkový terminál pro výběr souborů pro přenos
- později začal být ftp protokol integrován do webových prohlížečů
  - pro ftp servery se používala notace ftp:
- ftp používá TCP dva porty 20, 21
- obdobně jako telnet, žádné údaje nejsou šifrované
  - uživatelská jména i hesla pro přihlášení jsou přenášeny jako prostý text
  - data souborů jsou také přenášena bez šifrování

SCP – secure copy

Kopírování souborů mezi počítači s využitím protokolu ssh

- program `scp` využívá vytvoření bezpečného ssh spojení mezi Vaším a cílovým počítačem
  - vytvořený bezpečný kanál je využit pro přenos dat souboru z nebo na Váš počítač
  - data jsou šifrovány symetrickým šifrováním, jehož bezpečný klíč byl dohodnut v rámci ssh
- kopírování souboru file na cizí počítač do adresáře path:
  - `scp file user@host:/path/`
- kopírování souboru file z adresáře path z cizího počítače do adresáře adres:
  - `scp user@host:/path/file adres`

SCP – secure copy

Scp lze použít pokud na vzdáleném počítači je spuštěn server sshd.

- program `scp` využívá vytvoření bezpečného ssh spojení mezi Vaším a cílovým počítačem
  - vytvořený bezpečný kanál je využit pro přenos dat souboru z nebo na Váš počítač
  - data jsou šifrovány symetrickým šifrováním, jehož bezpečný klíč byl dohodnut v rámci ssh
- kopírování souboru file na cizí počítač do adresáře path:
  - `scp file user@host:/path/`
- kopírování souboru file z adresáře path z cizího počítače do adresáře adres:
  - `scp user@host:/path/file adres`
- cesta k souboru ve vzdáleném adresáři je buď absolutní, nebo relativní

SFTP

SFTP je bezpečná varianta starého protokolu ftp

- `sftp user@host` — interaktivní
- Důvody preferovat SFTP/SCP před FTP

SSHFS — souborový systém přes SSH

- Připojení vzdálené souborové složky přes FUSE
  - `sshfs user@host:/remote /mnt`
- Použití: jednoduchý přenos bez sync nástrojů
- Nevýhody: výkon a locking - zamykání vzdáleného adresáře
  - není vhodné pro paralelní změny v systému

SSH a Windows

- ssh, scp, sftp - jsou standardní součásti UNIXových systémů
- pro Windows můžete:
  - využít WSL - neboli Ubuntu pod Windows
    - pak můžete využívat veškeré nástroje jako v unixových systémech
  - využít PuTTY - pro ssh připojení, případně PuTTYgen pro generování klíčů
  - využít WinSCP - jako náhradu scp
    - objeví se vám okno, ve kterém můžete kopírovat soubory ze/na vzdálený počítač

Monitoring a logging

- Logy: `/var/log/auth.log`, `journalctl -u sshd`
- Audit: kdo a kdy se přihlásil
- Centralizovaný logging (ELK, Graylog)

Pokročilé: Certifikáty OpenSSH

- OpenSSH podporuje certifikáty (CA-signed keys)
- Výhody: centrální správa identity, expirace, principals
- Příklad použití `ssh-keygen -s ca -I id` atd.

Konfigurace serveru

Konfigurace ssh serveru je nejčastěji v souboru `/etc/ssh/sshd_config`

- Důležitá nastavení serveru:
  - `PermitRootLogin no` - zákaz přihlášení pro uživatele root
  - `PasswordAuthentication no` - po nastavení klíčů zrušte možnost přihlášení heslem
    - je to otevřená brána k útokům
    - `AllowUsers` / `AllowGroups` - případně můžete ponechat přihlášení heslem jen pro uživatele, kteří si ještě klíč nenastavily
  - `Port`- zvažte jiný port pro připojení na ssh
- Po změně konfiguračního souboru musíte službu `sshd` restartovat: `systemctl restart sshd`

Problémy

Co dělat pokud se nemůžete přihlásit?

Zkontrolujte:

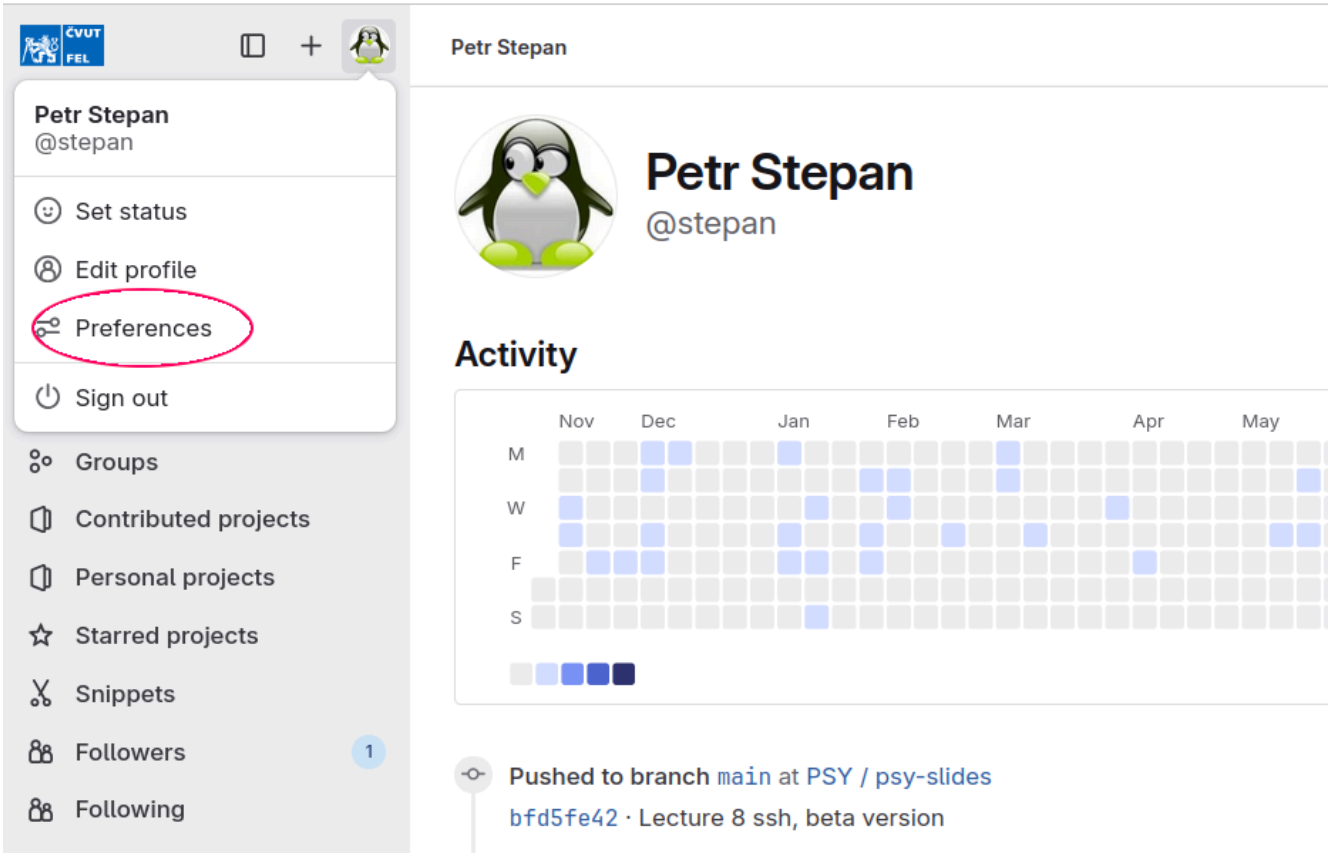
- výpis při zapnutých ladicích hlášeních `ssh -vvv user@host`
  - hlášení Permission denied — často špatné práva adresáře `~/.ssh` nebo soukromého klíče
- log programu `sshd` na servrovém počítači
- nastavení počítačové síte - firewally

SSH klíč pro gitlab

Chcete použít klíč pro přístup na gitlab?

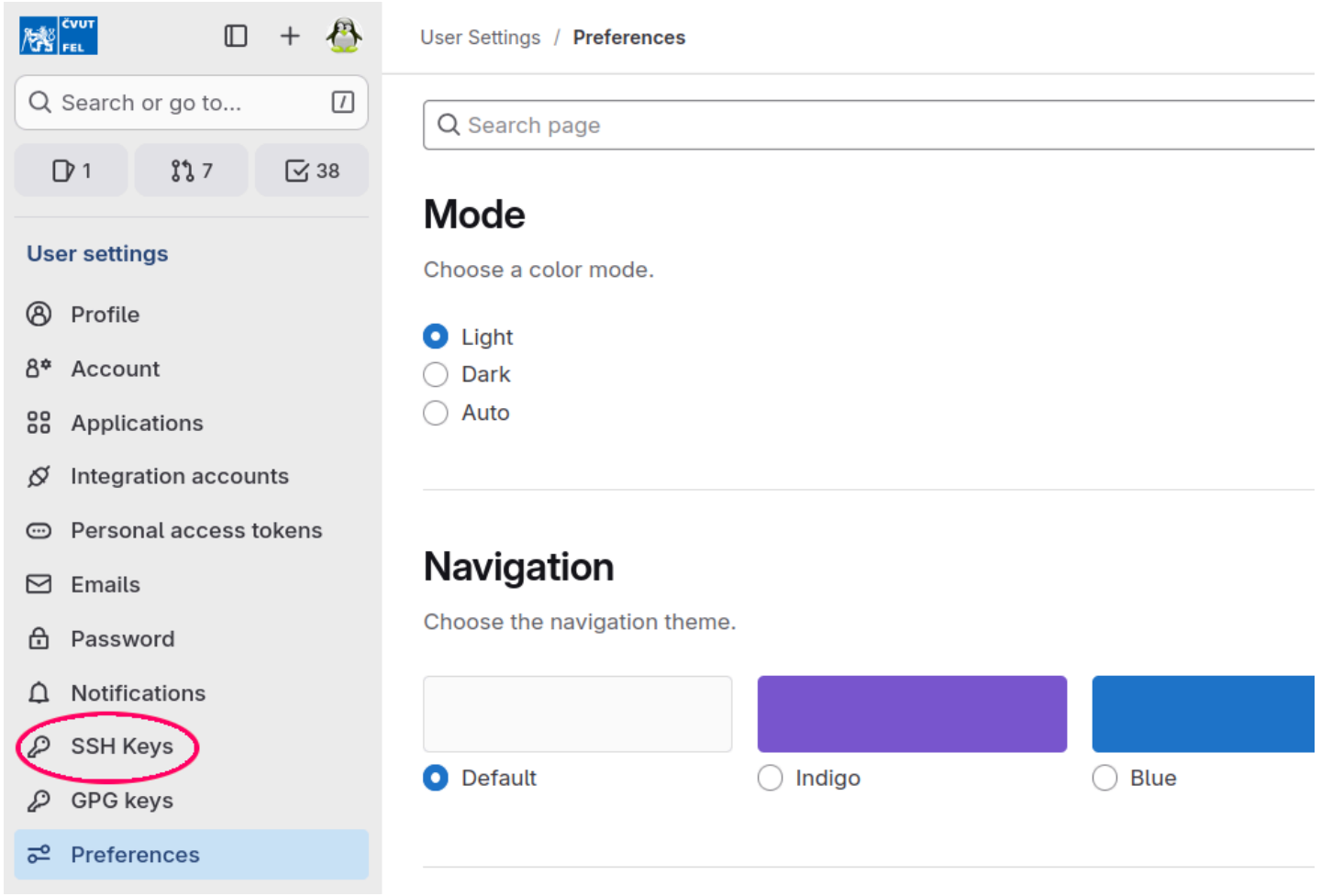
Prostě si ho zaregistrujte do gitlabu (nebo do githubu a jiných gitů).

Nejdříve si otevřete záložku preference v osobních nastaveních.



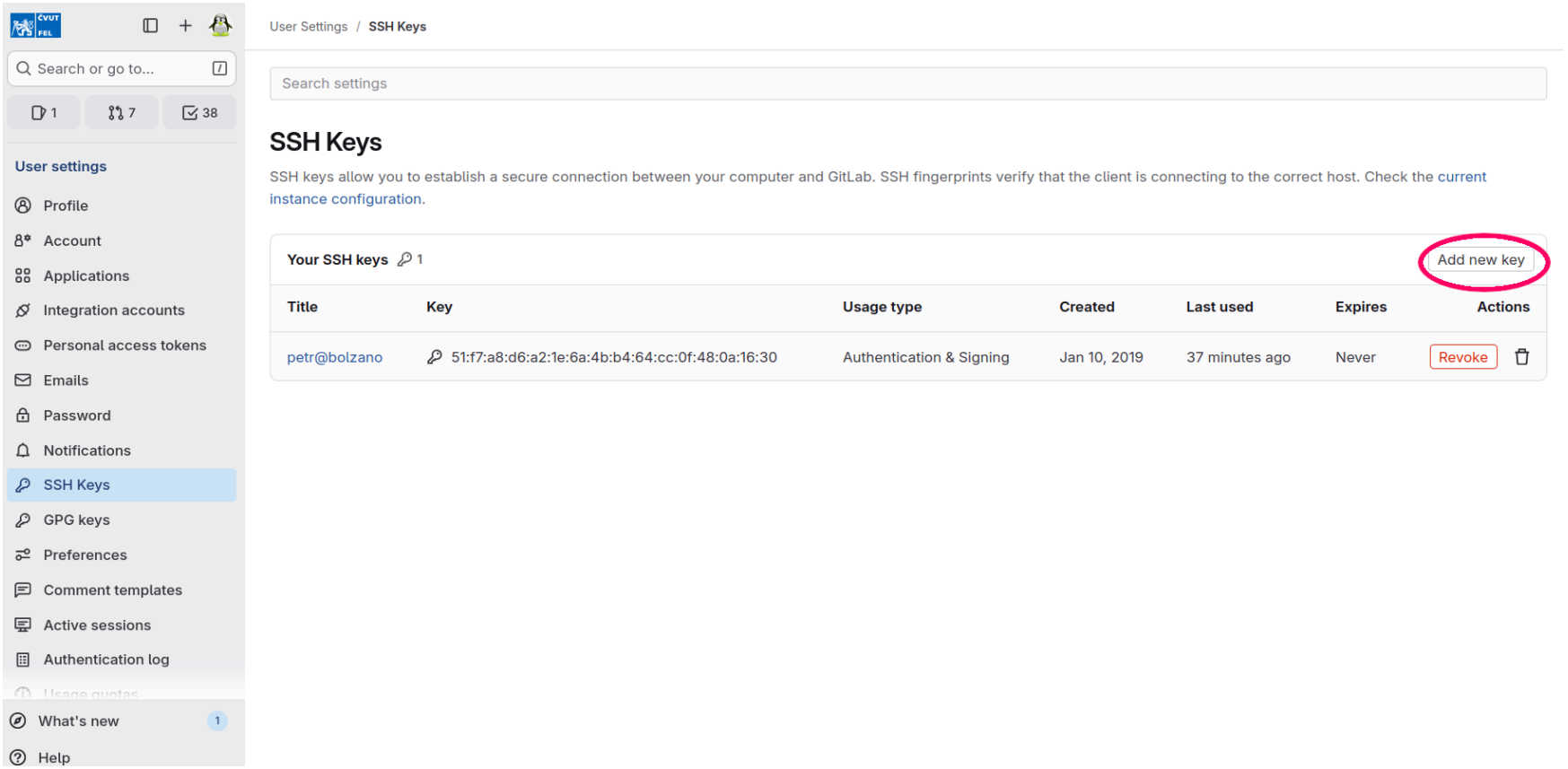
SSH klíč pro gitlab

Pak v levé části vyberte položku ssh-keys.



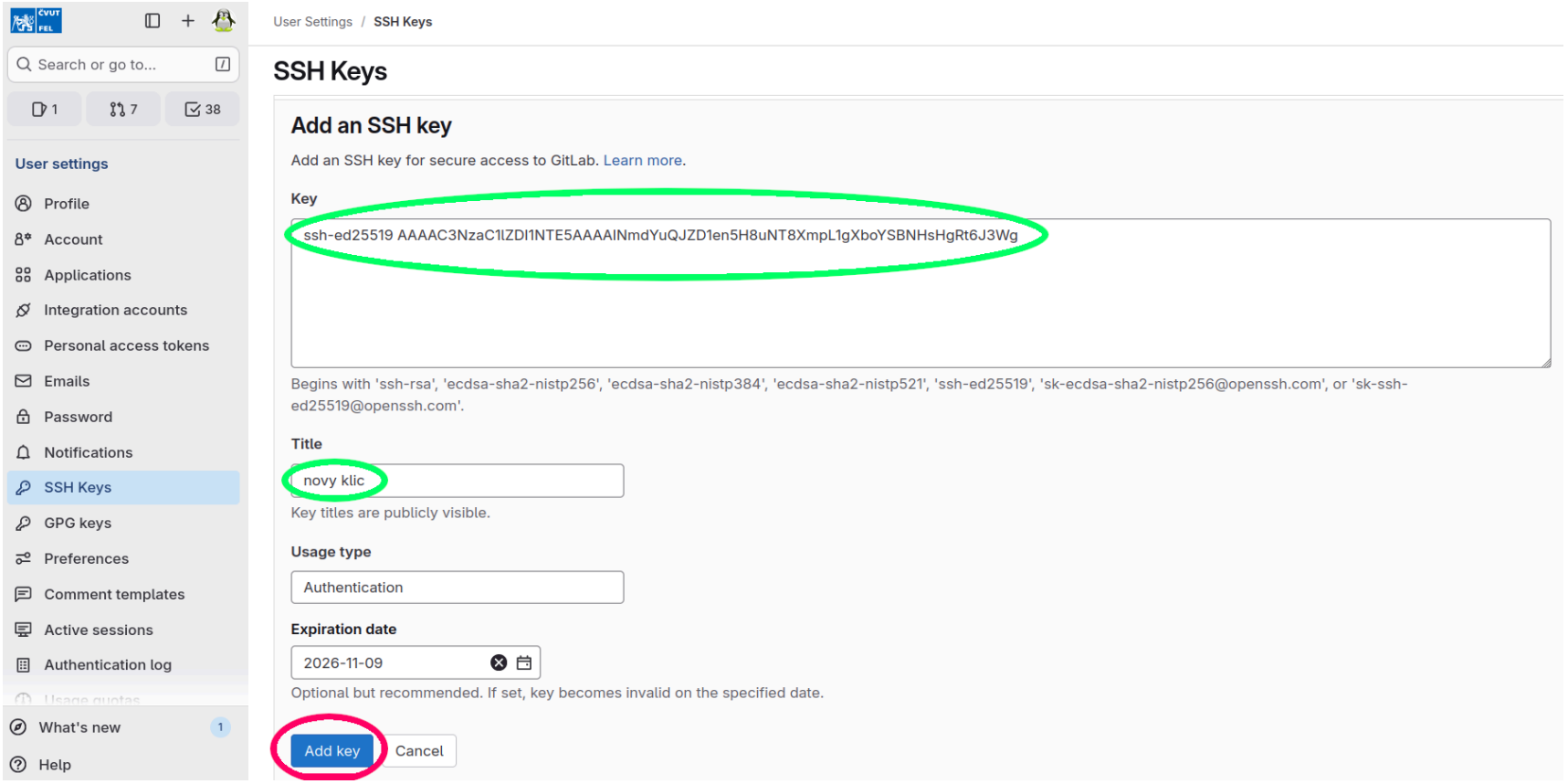
SSH klíč pro gitlab

Zde uvidíte všechny svoje registrované klíče a můžete si přidat další.



SSH klíč pro gitlab

Zkopírujte v textovém editoru text veřejného klíče a vložte ho do okna, klíč si pojmenujte a uložte.



Závěr a zdroje

To je dnes vše.

Na cvičení si to vyzkoušíte prakticky.