Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Bachelor's thesis
HS16 Studiengang Informatik

Design and Implementation of an Alternative to SSH

Authors Raphael Emberger, Kal-El,
Musashi Miyamoto

Date May 2, 2019

Abstract

Preface

The *Secure Shell*(later referred to as SSH) protocol (Moorer 1971, Bider & Baushke 2012, Baushke 2017, Bider 2018*a*,*b*) is a system that allows a user to log in on a remote machine and perform tasks on that remote machine via a *Command Line Interface*(later referred to as CLI). *SSH* is widely known and used in everyday tasks. However: It is now over twelve years old in its current form. One of the problems with *SSH* is its complexity, both in the initial phase when key material is exchanged, but also later, for example because the server must always decide whether to return a character that has been sent to it or not (echo).
The goal of this work is a radically simplified protocol, which in its functions is similar to *SSH* (N.B. the similarity concerns the functions, not necessarily the protocol details). I develop the protocol, as well as a client and a server - all in *Go/Golang*(later referred to as Go). I demonstrate that the software can replace *SSH* by showing that it can handle several common use cases, among them:

- Interactive session
- `Rsync` with my solution as transport protocol

This bachelor thesis was proposed by Dr. Stephan Neuhaus (Neuhaus 2018) and aroused my interest as it is a challenge in the domain of information security and will produce a palpable result.
On this note I would like to thank *Zurich University of Applied Sciences*(later referred to as ZHAW) for granting me the opportunity to do my Bachelors thesis here and Dr. Stephan Neuhaus for helping me along the way of this Bachelors thesis.

This thesis has been worded with technically literate readers in mind. However: For core concepts and special terms, a glossary can be found at 6.

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

DECLARATION OF ORIGINALITY
Bachelor's Thesis at the School of Engineering

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:

.....

Signature:

.....

.....

.....

The original signed and dated document (no copies) must be included after the title sheet in the ZHAW version of all Bachelor thesis submitted.

Contents

1. Introduction

1.1. Initial Position

There was no thesis done on this subject that could have been used as reference. There are however several software projects that deal with a similar problem.

1.1.1. OpenSSH

The most noteworthy work to mention is of course *SSH* itself. *OpenSSH* (1999) is the name of the open source project which provides millions of administrators and developers with the ability to securely connect to a remote host. It replaces the up until then widely used protocols like `telnet`1.1.2 and `rlogin`/`rsh`1.1.3.
SSH uses *Transport Layer Security*(later referred to as TLS) to secure the communication channel between two peers and has earned itself a spot on the low end of the port table: It occupies port 22.
*SSH*s features can be used very flexibly: After it builds up a secure connection between a client and a server, it can be used to remotely login and use a terminal on that machine. It can also forward traffic on local ports to the remote host through the secure channel. This is also used by third party programs such as `rsync`.
When it comes to the log in procedure itself, *SSH* allows for standard user log in using the *Application Programming Interface*(later referred to as API) of the *Pluggable Authentication Modules*(later referred to as PAMs). Another feature is whitelisting of clients via their public keys, which barres intrusion via hijacked user-password-credentials.
After a secure connection could be established, there are multiple possibilities to use the opened channel. One is to forward the *Graphical User Interface*(later referred to as GUI) of a remote program to the client. Another one is to use this channel to tunnel more connections through it: For example can the traffic of an application which uses a specific port be forwarded to the remote host. This can obscure and secure this traffic between the host and the server.

1.1.2. Telnet

Telnet(C. Stephen 1969, Postel & Reynolds 1983) is an old(1969) and deprecated communication protocol which doesn't feature any security. However, in other implementations, *Telnet Secure*(later referred to as TELNETS) was proposed, which features encryption over the communication channel.
Telnet still has 23 as its very own port assigned to it.

Go-Telnet

Go-Telnet(Krempeaux 2016) is a *TELNETS* supporting client-server-application which has been implemented in *Go*.

1.1.3. Berkeley r-commands

The Berkley r-commands are a set of commands to do certain tasks on remote hosts. Those tasks are similar to their counterparts without a leading "`r`".

- `rlogin`
  This command connects to the host and performs a `login` command, which includes authentication and if successful, spawning a user Shell.

- `rsh`
  `rsh` spawns a Shell without the log in process.
- `rexec`
  With this command, the user can log in to a remote machine and execute one command.
- `rcp`
  Using this command gives the user the ability to copy from and to a remote host.
- `rwho`
  This command tells the user what users are currently logged in on the remote machine.
- `rstat`
  `rstat` displays file system information from remote hosts.
- `ruptime`
  With this command, the user can see the uptime, number of logged in users and current work load of the remote machine.

1.2. Task

The objective of this thesis is to create a simplified communication protocol between a client and a server to allow the client to log in on the server.

2. Theoretical Principles

3. Method

4. Results

5. Discussion And Prospects

6. Index

6.1. Bibliography

Baushke, M. D. (2017), 'More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)', *RFC* **8268**, 1–8.
URL: https://doi.org/10.17487/RFC8268 3

Bider, D. (2018*a*), 'Extension Negotiation in the Secure Shell (SSH) Protocol', *RFC* **8308**, 1–14.
URL: https://doi.org/10.17487/RFC8308 3

Bider, D. (2018*b*), 'Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol', *RFC* **8332**, 1–9.
URL: https://doi.org/10.17487/RFC8332 3

Bider, D. & Baushke, M. D. (2012), 'SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol', *RFC* **6668**, 1–5.
URL: https://doi.org/10.17487/RFC6668 3

C. Stephen, C. (1969), 'Network subsystem for time sharing hosts', *RFC* **15**, 1–5.
URL: https://doi.org/10.17487/RFC0015 6

Krempeaux, C. I. (2016), 'go-telnet', Github.
URL: https://github.com/reiver/go-telnet 6

Moorer, J. A. (1971), 'Second Network Graphics meeting details', *RFC* **253**, 1.
URL: https://doi.org/10.17487/RFC0253 3

Neuhaus, S. (2018), 'Bachelorarbeit 2019 - FS: BA19_neut_03'.
URL: https://tat.zhaw.ch/tpada/arbeit_vorschau.jsp?arbeitID=16096 3

OpenSSH (1999).
URL: https://www.openssh.com/ 6

Postel, J. & Reynolds, J. K. (1983), 'Telnet protocol specification', *RFC* **854**, 1–15.
URL: https://doi.org/10.17487/RFC0854 6

6.2. Glossary

Application Programming Interface
Accessible interface for developers to use external code. 6

Command Line Interface
A text based interface centered around commands to perform specific tasks. 3

Go/Golang
Google's programming language. 3

Graphical User Interface
Graphical interface for the user to visually interact with a program. 6

Pluggable Authentication Module
Modules for user authentication. 6

Secure Shell
An client-server-application that allows remote login and interaction with a Shell. See 1.1.1. 3

Secure Sockets Layer
Cryptographic protocol to secure the communication between two peers via symmetric cryptography. Deprecated. 13

Telnet Secure
Telnet with *Secure Sockets Layer*(later referred to as SSL) encryption. 6

Transport Layer Security
Newer and recommended version of *SSL*. 6

Zurich University of Applied Sciences
Name of my university of trust. 3

port A point for traffic to flow, represented by an unsigned integer of up to 2 bytes. The name was chosen as an analogy to ports for ships. 6

Shell A *CLI* program, that reads user input line-by-line and executes those commands. 6, 7, 13

6.3. List of Figures

6.4. List of Tables

6.5. List of Listings

6.6. Acronym Glossary

API *Application Programming Interface* 6, *See Application Programming Interface*

CLI *Command Line Interface* 3, 13, *See Command Line Interface*

GUI *Graphical User Interface* 6, *See Graphical User Interface*

Go *Go/Golang* 3, 6, *See Go/Golang*

PAM *Pluggable Authentication Module* 6, *See Pluggable Authentication Module*

SSH *Secure Shell* 3, 6, *See Secure Shell*

SSL *Secure Sockets Layer* 13, *See Secure Sockets Layer*

TELNETS *Telnet Secure* 6, *See Telnet Secure*

TLS *Transport Layer Security* 6, *See Transport Layer Security*

ZHAW *Zurich University of Applied Sciences* 3, *See Zurich University of Applied Sciences*

A. Appendix

A.1. Project Management

A.2. Others