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.

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 official formulation of the tasks can be found in the appendix A.1.1.
The objective of this thesis is as follows:

- Design and implement a client-server protocol that can manage interactive sessions.
- Design and implement a privilege-separation architecture on the server side that allows safe dropping of privileges once a client establishes a connection.

For a passing grade (4.0), the work must contain at least the following:

- In the thesis, an introduction to the problem and why the envisaged solution will solve it.
- In the thesis, a survey of related work in the area.
- In the thesis, a detailed design of the solution.
- In the thesis, an evaluation of the performance of the implemented solution.
- In the software, a privilege-separation architecture.

Incorporating the following components will improve the grade:

- In the related work section of the thesis, a comparison of all the related work with the envisaged solution, outlining why the envisaged solution is better.
- In the thesis, a detailed analysis of the security of the solution, including possible attacks and defenses.
- Use of TLS as the transport layer.
- A proof-of-concept client that can handle interactive sessions.
- A proof-of-concept client that works as a transport for `rsync`.

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. Used acronyms are listed in 6.5.

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.1.1. Official Statement of Tasks

Bachelor Thesis
Preventing Supply Chain Insecurity by Authentication on Layer 2

Stephan Neuhaus

2017-06-15

1 Introduction

The SSH protocol [RFC253, RFC6668, RFC8268, RFC8308, RFC8332] 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). You develop the protocol, as well as a client and a server. You demonstrate that your software can replace SSH by showing that it can handle several common use cases, among them:

- Interactive session
- Rsync with the SSH replacement as transport protocol

2 Task

To this end, this thesis will

- design and implement a client–server protocol that can manage interactive sessions
- design and implement a privilege-separation architecture on the server side that allows safe dropping of privileges once a client establishes a connection

For a passing grade (4.0), the work must contain at least the following:

- in the thesis, an introduction to the problem and why the envisaged solution will solve it;
- in the thesis, a survey of related work in the area;
- in the thesis, a detailed design of the solution;
- in the thesis, an evaluation of the performance of the implemented solution; and

- in the software, a privilege-separation architecture.

These requirements do not contain anything related to security. This is not an accident.

Incorporating the following components will improve the grade. The more components are included, the better the grade will be.

- In the related work section of the thesis, a comparison of all the related work with the envisaged solution, outlining why the envisaged solution is better;
- in the thesis, a detailed analysis of the security of the solution, including possible attacks and defenses;
- use of TLS as the transport layer;
- the software in a public repository, ideally on Github;
- a proof-of-concept client that can handle interactive sessions;
- a proof-of-concept client that works as a transport for rsync;

ZHAW's School of Engineering no longer provides formal language lessons for its students as part of the curriculum. I am therefore giving notice that submitting a thesis with large amounts of orthographical or grammatical errors lead to a lower grade.

The thesis can be submitted in German or English. English is prefered, but submitting in German will not lead to a lower grade.

A.1.2. Project Plan

Revised Project Plan

March 2019

1 Introduction

The work on the final thesis is divided into several sub tasks. The individual tasks and respective time planning was defined early.
As this is a field of work, we as a team are not familiar with, we have decided to change our original plan: "Project Plan" to a revised version.
The biggest difference is that the Protype is developed earlier but we are removing some security measures respectively moving it to an optional goal We are subdividing the following area of development:

- Technical research: The research starts with a collection of examined current solution to a secure data transference, followed by a list of pro and cons for the approaches that includes a preferred selection. Moreover, we would survey the current state development within that field.
- Conceptualising: Look for alternative solutions and compare them. Plan the development.
- Prototype: A unsecure SSH Client
- Testing: Do generalized tests, identify possible security vulnerability
- Rework of the secure shell: Modify the secure shell based on the newly discovered needs

Furthermore there are specific milestone within the project process, which we would use to realign and discuss our time division.

2 Visualization

The project plan is documented in the form of a chart and is updated throughout the project. This way, deviations can be detected early and can be discussed with the supervisor and within our team.

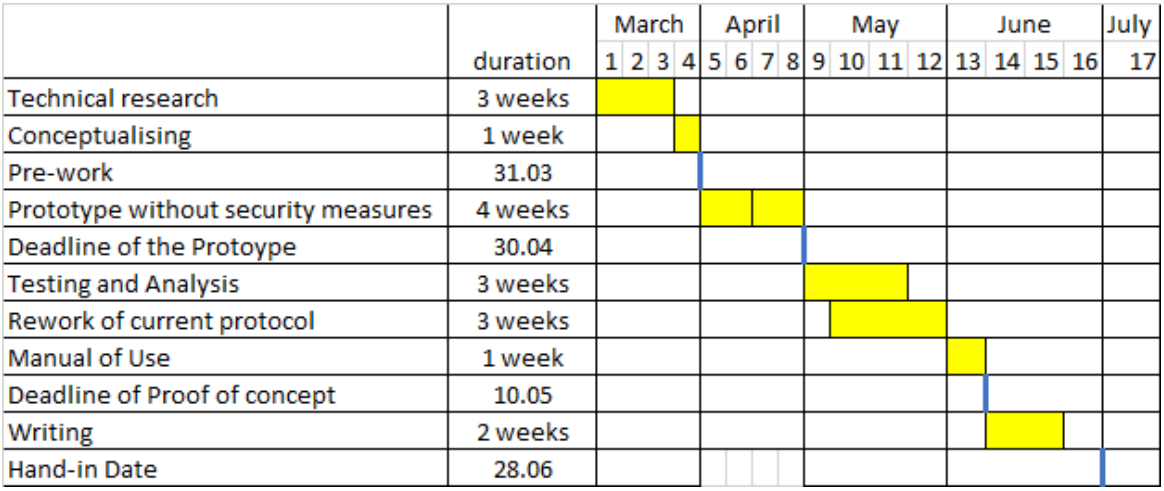
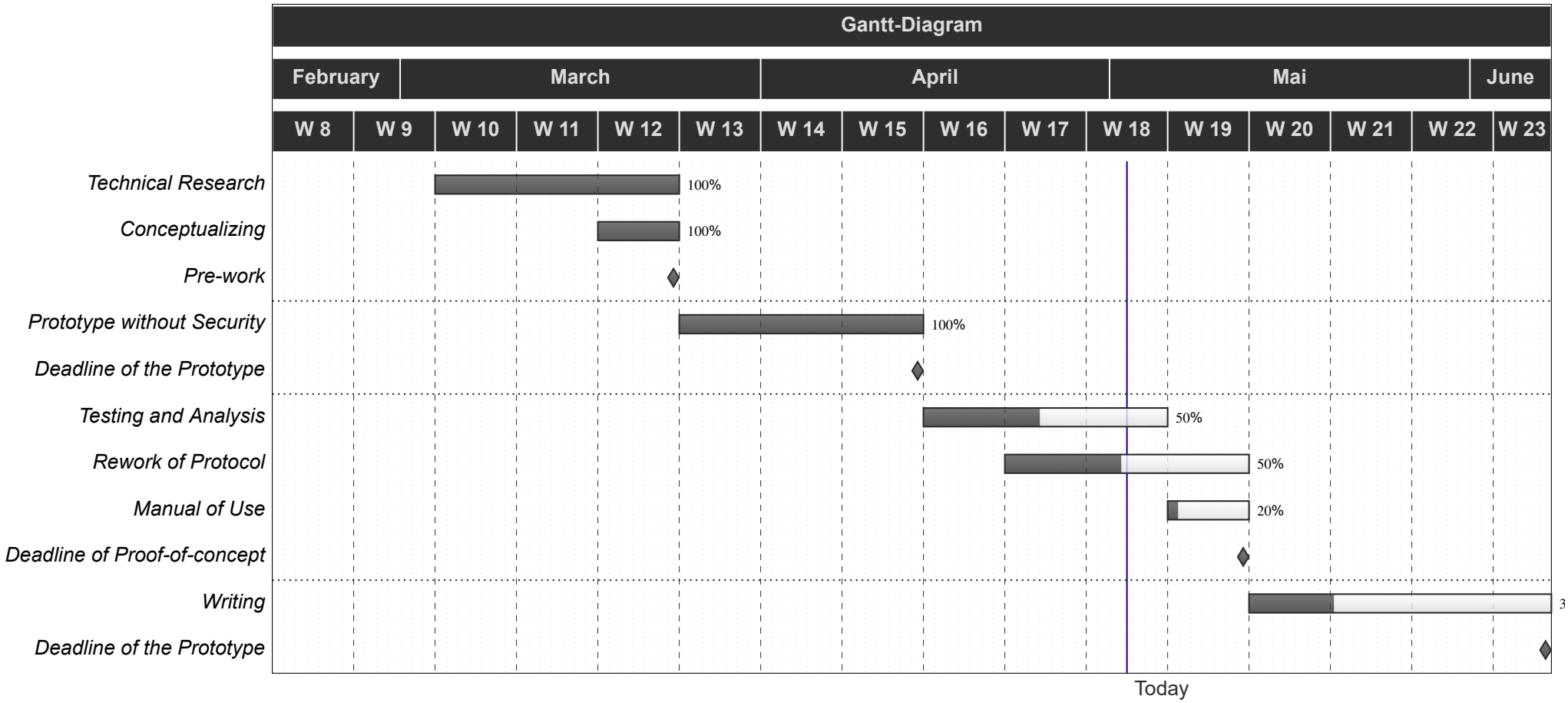


Figure 1: Project plan


Gantt-Diagram
February
March
April
Mai
June
W 8
W 9
W 10
W 11
W 12
W 13
W 14
W 15
W 16
W 17
W 18
W 19
W 20
W 21
W 22
W 23
Technical Research
Conceptualizing
Pre-work
Prototype without Security
Deadline of the Prototype
Testing and Analysis
Rework of Protocol
Manual of Use
Deadline of Proof-of-concept
Writing
Deadline of the Prototype
100%
100%
100%
50%
50%
20%
3
Today

A.1.3. Meeting Minutes

The meeting minutes have a disruption in style and execution beginning from the 6th meeting. Reason for this is because in the beginning of the project, Mr Schwarz was responsible for keeping the minutes, but he opted out of the project.

Oh my Gosh - Meeting protocol

1 2nd Meeting

Participent
Bachelor thesis supervisor - Stephan Neuhausen
Bachelor student - Raphael Emberger
Bachelor student - Kevin Schwarz
Time duration of the meeting
1 hour and 15 minutes

1.1 Objectives

1. Keeping the bachelor thesis supervisor informed on the state of affairs
2. Getting an overview on how the progress relate to the scheduled progress

1.2 Summary

In this meeting the following things were achieved:

1. Decision towards ITC and UDP was achieved
2. A rough draft of a generalized process was finalized and presented to the supervisor
   (a) Smaller misconceptions were resolved
   (b) Fields where further research is warranted was shown
3. Reiteration of the project goal
4. Further Delimitation of the project extent.
   (a) Smaller misconceptions were resolved

1.3 Tasks and resources

The following were left as tasks or as a research subject for the next meeting in descending priority:

1. Understanding shell forwarding
2. Researching the limitation of IOCTL - raw and device specific output
3. Pseudo terminals
4. Persistence in relation to Environment variables

1.4 Next meeting

The next meeting plan were not changed, the formerly decided weekly scheduled date still stands.
The next meeting is dated: [14 / 03 / 19] on the first floor of the Zürich location of the ZHAW within the Room 0.03.

Oh my Gosh - Meeting protocol

1 3rd Meeting

Participent
Bachelor thesis supervisor - Stephan Neuhausen
Bachelor student - Raphael Emberger
Bachelor student - Kevin Schwarz
Time duration of the meeting
1 hour

1.1 Objectives

1. Keeping the bachelor thesis supervisor informed on the state of affairs
2. Getting an overview on how the progress relate to the scheduled progress

1.2 Summary

In this meeting the following things were achieved:

1. General overview of a PTY
2. Certain foundation towards developing a non-secure secure shell were explained:
   (a) Smaller misconceptions were resolved
   (b) Fields where further research is warranted was shown
3. Reiteration of the project goal
4. Further Delimitation of the project extent.
   (a) Smaller misconceptions were resolved

1.3 Tasks and resources

The following were left as tasks or as a research subject for the next meeting in descending priority:

1. Understanding shell forwarding
2. Researching the limitation of IOCTL - raw and device specific output
3. Pseudo terminals
4. Persistence in relation to Environment variables

1.4 Next meeting

The next meeting plan were not changed, the formerly decided weekly scheduled date still stands.
The next meeting is dated: [14 / 03 / 19] on the first floor of the Zürich location of the ZHAW within the Room 0.03.

Oh my Gosh - Meeting protocol

1 4th Meeting

Participent
Bachelor thesis supervisor - Stephan Neuhausen
Bachelor student - Raphael Emberger
Bachelor student - Kevin Schwarz
Time duration of the meeting
25 Minutes

1.1 Objectives

1. Keeping the bachelor thesis supervisor informed on the state of affairs
2. Gain an introduction to encryption

1.2 Summary

In this meeting the following things were achieved:

1. Introduction to Bash was given
2. Move up of the prototype deadline
3. Change of scheduled development plan:
   (a) Decrease of the SSH scope
   (b) Reduction of the security measures to an optional goal
   (c)

1.3 Tasks and resources

The following were left as tasks or as a research subject for the next meeting in descending priority:

1. Researching the infrastructure of GO-order
2. Researching Bash and PTY on Windows enviroment
3. First Server - Client Demo
4. Crafting a simple process diagram for the next meeting

1.4 Next meeting

The next meeting plan were not changed, the formerly decided weekly scheduled date still stands.
The next meeting is dated: [28 / 03 / 19] on the first floor of the Zürich location of the ZHAW within the Room 0.03.

Oh my Gosh - Meeting protocol

1 5th Meeting

Participent
Bachelor thesis supervisor - Stephan Neuhausen
Bachelor student - Raphael Emberger
Bachelor student - Kevin Schwarz
Time duration of the meeting
45 Minutes

1.1 Objectives

1. Keeping the bachelor thesis supervisor informed on the state of affairs
2. Demonstrate Demo

1.2 Summary

In this meeting the following things were achieved:

1. The Prototype was tested.
2. Process Diagram was explained
3. 4 open Problems were discussed:
   (a) PAM Struct and how they work
   (b) Generalized Certkey location
   (c) Use of the Prototype within Linux
   (d) Correct pipe lining and forking

1.3 Tasks and resources

The following were left as tasks or as a research subject for the next meeting in descending priority:

1. Further testing of both Linux, Apple and Windows environment
2. Solving of Pam Struct problem
3. Further development

1.4 Next meeting

The next meeting plan were modified, the formerly decided weekly scheduled date still stands.
The next meeting is dated: [5 / 04 / 19] on the first floor of the Zürich location of the ZHAW within the Room 0.13.

Design and Implementation of an Alternative to SSH

Meeting Minutes

1 Attendees

Present: Stephan Neuhaus, Raphael Emberger
Absent: Kevin Schwarz*(illness)*

2 Initiation

The meeting took place on the *Friday, 5th of April 2019, 13:00* in *ZL0.13, Lagerstrasse 45, Zürich*. Raphael Emberger was resonsible for the minutes.

3 Points of discussion

3.1 Process forking unsuccessful

Attempts on forking a sub-process were unsuccessful. The reason for this was that the standard library of Go doesn't allow such mechanics, as Go was designed with go-routines in mind instead.

Solution A quick test with cgo yielded a viable solution to the problem: Using the C-routine fork() a fork was successful.

3.2 Shell instantiating and forwarding

Attempts in forwarding the client connection to a server-side shell's stdin and its stdout and stderr to the connection of the client were unsuccessful.

Solution One quick tests showed that hooking up the std* pipes to a local shell process with Go worked just fine. Therefore it was deemed feasible to transfer the entire interface to the client.

3.3 Participation of Mr. Schwarz

Up until this date, the participation of Mr Schwarz was remarkable little in terms of writing on the code base of the project. The present parties agreed on this matter.

Solution It was decided to give Mr Schwarz a choice of action: Either he starts to participate heavily in the project from now on or he opts out of the project entirely.

4 Old Business

- **Login attempts in Linux fail:** This problem was deemed lower priority, as Login works on the WSL and can still be dealt with in later stages of the project.

Next Meeting

Friday, 5th of April 2019, 13:00 in *ZL0.13, Lagerstrasse 45, Zürich*

Design and Implementation of an Alternative to SSH

Meeting Minutes

1 Attendees

Present: Stephan Neuhaus, Raphael Emberger, Kevin Schwarz

2 Initiation

The meeting took place on the *Friday, 12th of April 2019, 13:00* in *ZL0.13, Lagerstrasse 45, Zürich*. Raphael Emberger was resonsible for the minutes.

3 Points of discussion

3.1 Participation of Mr. Schwarz

Mr Schwarz decided to opt out of the project because of time issues. Mr Neuhaus will therefore adapt the outline of the project.

3.2 Reading user data works

The new module to read user data via the getpwnam(3) API has been implemented using cgo. It can read all the required data(i.e. the user shell which wasn't supported in the go standard library).

3.3 Forking implemented, but causes problems

Forking has been implemented via cgo but after forking, the net.Conn object cannot be used by the child process. There is also the to further investigate, whether after forking a new process actually gets started, as a quick look at the processes didn't reveal that a fork has been processed.

Solution To counter this problem it is suggested to do the connection build up via cgo using the C-socket API. This returns an integer as a file descriptor, which shouldn't cause problems when forking.

3.4 Remote start and handling of a shell has issues

After successfully hooking up the channels from the client to the shell process, almost all mechanics work as expected with exception of missing characters like the PS{1,2,3,4} prompts.

Solution It is suggested to compare the environment variables of the child shell process and the usual terminals to see if there are deal breaking differences. Adjusting the child shells environment variables might fix the problem.

4 Old Business

- **Login attempts in Linux fail:** This problem was deemed lower priority, as Login works on the WSL and can still be dealt with in later stages of the project.

Next Meeting

Friday, 26th of April 2019, 13:00 in *ZL0.13, Lagerstrasse 45, Zürich*

Design and Implementation of an Alternative to SSH

Meeting Minutes

1 Attendees

Present: Stephan Neuhaus, Raphael Emberger

2 Initiation

The meeting took place on the *Friday, 26th of April 2019, 13:00* in *ZL0.13, Lagerstrasse 45, Zürich*. Raphael Emberger was resonsible for the minutes.

3 Points of discussion

3.1 Login and shell usage

The remote login, starting and usage of a user shell works now. It still does not behave like intended, as there are warnings printed on the screen and every line written gets echoed back, but overall, it works.

3.2 Pty echoes stdin back to stdout on client

As described in the point above, when entering shell commands on the shell after remote login, the written lines gets echoed back after hitting enter.

Solution The reason this was so, is because the terminal on the client was still in the *cooked* mode rather than the *raw* mode, which behaves differently from the default *cooked* mode, which read line by line and catches and interprets signals like [Ctrl]+[C] or [Ctrl]+[D]. Setting the terminal into *raw* mode should solve the issue as explained in ”The Linux Programming Interface”.

3.3 Transfer of SIGWINCH/ioctl

As described in the first point of discussion, when dropping into the remote shell, there are warnings displayed about a problem with ioctl.

Solution In ”The Linux Programming Interface” it is also mentioned that making the child the session leader would solve this issue.

3.4 Login with test user fails

Trying to login with the test user results in an error when starting the shell because of missing files.

Solution This issue was easily solved as the script for setting up the test user was faulty: It didn’t properly create the home directory of the test user and since the process which dropped privilege after login didn’t have root rights anymore, it couldn’t enter the home directory. Therefore, the directory owner and permissions were amended.

3.5 Forking abandoned

3.4 of the last meeting suggested using the C-style sockets to pass the file descriptors to the child process, which should solve the issue with forking and still using the net.Conn object. This has been implemented and after some adjustment worked out well.

4 Old Business

- **Login attempts in Linux fail** This problem was deemed lower priority, as Login works on the WSL and can still be dealt with in later stages of the project.

Next Meeting

Friday, 3th of March 2019, 12:30 in *ZL0.13, Lagerstrasse 45, Zürich*

A.2. Others