BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

PORTING TANG TO OPENWRT

PORTOVANIE TANG NA OPENWRT

TERM PROJECT
SEMESTRÁLNÍ PROJEKT

AUTHOR **TIBOR DUDLÁK**
AUTOR PRÁCE

SUPERVISOR **Ing. ONDREJ LICHTNER**
VEDOUCÍ PRÁCE

BRNO 2017

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Reference

DUDLÁK, Tibor. *Porting Tang to OpenWRT*. Brno, 2017. Term project. Brno University of Technology, Faculty of Information Technology. Supervisor Lichtner Ondrej.

Porting Tang to OpenWRT

Declaration

Hereby I declare that this term project was prepared as an original author's work under the supervision of Ing. Ondrej Lichtner. The supplementary information was provided by Jan Pazdziora, Ph. D.

. . . .
Tibor Dudlák
January 26, 2017

Contents

Chapter 1

Introduction

Nowadays, the whole world uses information technologies to communicate and simply to spread knowledge in form of bits to the other people. But there are personal information such as photos from a family vacation, videos of our children as they grow, contracts, testaments and so on which we would like to protect.

Encryption protects our data even when we do not know that. It protects our conversation privacy, personal information stored all over governmental authorities, bank accounts, in general data transmitted around the internet and stored on our hard drives. Encryption provides process of transforming our information in such way that only trusted person can decrypt data and access it. An unauthorized person might be able to access secured data but will not be able to read the information from them without the proper key. The most important thing is keeping the encryption key (the password) a secret.

The goal of this bachelor thesis will be to port *Tang* server [4], its dependencies, and *Clevis* framework [5] to *OpenWRT* system [6]. With accomplishing, this we will be able to automatize process of unlocking encrypted drives on our private respectively home network. There will be no need for any decryption server but only *OpenWRT* device running the *Tang* server itself.

My primary goal is to focus on the packaging system of the *OpenWRT* and build processes on it.

Treba to doplnit...

Chapter 2

Encryption

For most of us is common to have a password protected system. But the encrypted disk requires another password (key) to decrypt. Imagine that you came home in mood to enjoy your time and your system ask for password not once but twice. I think this is reason why most of us do not use encryption even when we know it will protect our data. This is what Tang 4 is for and to whom is for because we want automatize things.

Lets look how the encryption is typically done. As you can see in the image below it all starts with desire to keep our data to ourselves and as a secret to the other people. At the most these secrets are stored on our hard drives. Usually we encrypt this secret by using an encryption key. But our secret data might grow in size, and it is time and resource consuming to decrypt and encrypt secret every time encryption key changes or it is compromised. Because of that we wrap encrypted data in the key encryption key and this is what system prompts from us on boot. So changing key encryption key does not affect encrypted data. We can change it whenever we desire to and redistribute new key to all users who are suposed to access this data.

To automatize this we could generate something cryptographicaly stronger than user provided password. Then we store this cryptographicaly stronger random key on some remote system which is can then get from it. This is basically how the Escrow 3 model works.

Chapter 3

Escrow

Before Tang, automated decryption was handled usually with key escrow. Client using key escrow usually generates a key, encrypt data with it and then stores the key in a remote server. Key escrow server (also known as a “fair” cryptosystem) stores decryption keys. This should signal potential security risks when storing and also transmitting keys over network. For security purposes key escrow uses SSL/TLS to provide privacy and data integrity between server and client. Even with SSL/TLS key escrow requires authentication. To sum up an authorized third party may gain access to those keys under certain circumstances only.

Treba to doplnit...

Chapter 4

Tang

Tang server is open source project implemented in C programming language, and it bind data to network presence. It rely on few other software libraries:

- http-parser 4.1
- systemd / xinetd 4.2
- jose 4.3
  - jansson 4.3.1
  - openssl 4.3.2
  - zlib 4.3.3

Now when we know what Tang rely on and what is for we should be more specific about it and find out how Tang provides its service. What binding data to network presence really means? Essentially it allows us to make some data to be available only when the system containing the data is on a certain, usually secure, network.

Tang server advertises asymmetric keys and client is able to get list off these signing keys 4.3 by HTTP GET request. Now follows the provisioning step. With list of these public keys may process of encrypting data start. Client chooses one of asymmetric keys to generate a unique encryption key with. After this client encrypt data using created key. Once the data is encrypted, the key is discarded. Some small metadata has to be produced as part of this operation. The client should store these metadata in a convenient location. Finally when client want to access encrypted data it must be able to recover encryption key. This step starts with loading stored metadata and ends with simply performing a HTTP POST to Tang server.

	Escrow	Tang
Stateless	No	Yes
X.509	Required	Optional
SSL/TLS	Required	Optional
Authentication	Required	Optional
Anonymous	No	Yes

Table 4.1: Comparing Escrow and Tang

popisat...

Tang is originally packaged for Fedora OS version 24 and later but we can build it from source of course.

doplnic...

To build it from source download source from poject's GitHub or clone it.

Make sure you have all needed dependencies installed and then run:

```
$ autoreconf -if
$ ./configure -prefix=/usr
$ make
$ sudo make install
```

Optionally to run tests:

```
$ make check
```

4.1 http-parser

Tang uses this is parser for both parsing HTTP requests and HTTP responses. You can find this parser on its own GitHub [9].

4.2 systemd / xinetd

systemd [11] is a suite of basic building blocks for a Linux system. It provides a system and service manager that runs as PID 1 and starts the rest of the system. systemd provides aggressive parallelization capabilities, uses socket and D-Bus activation for starting services, offers on-demand starting of daemons, keeps track of processes using Linux control groups, maintains mount and automount points, and implements an elaborate transactional dependency-based service control logic.

4.3 José

José [10] is a C-language implementation of the Javascript Object Signing and Encryption standards. Specifically, José aims towards implementing the following standards:

- RFC 7515 - JSON Web Signature (JWS) [5]
- RFC 7516 - JSON Web Encryption (JWE) [3]
- RFC 7517 - JSON Web Key (JWK) [4]
- RFC 7518 - JSON Web Algorithms (JWA) [7]
- RFC 7519 - JSON Web Token (JWT) [6]
- RFC 7520 - Examples of ... JOSE
- RFC 7638 - JSON Web Key (JWK) Thumbprint [4]

JOSE (Javascript Object Signing and Encryption) is a framework intended to provide a method to securely transfer claims (such as authorization information) between parties.

Tang uses JWKs in comunication between client and server. Both POST request and reply bodies are JWK objects.

4.3.1 jansson

Jansson [8](licenced under MIT licence)is a C library for encoding, decoding and manipulating JSON data. It features:

- Simple and intuitive API and data model
- Comprehensive documentation
- No dependencies on other libraries
- Full Unicode support (UTF-8)
- Extensive test suite

4.3.2 OpenSSL

OpenSSL [2] contains an open-source implementation of the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is used by network applications to secure communication between two parties over network.

4.3.3 zlib

Library zlib [1] is used for data compression.

zmazat ako podkapitoly?

Chapter 5

Clevis

Chapter 6

OpenWRT

Chapter 7

Conclusion

Bibliography

[1] Adler, M.: *zlib*. [Online].
Retrieved from: https://github.com/madler/zlib

[2] Eric A. Young, T. J. H.: *OpenSSL*. [Online].
Retrieved from: https://github.com/openssl/openssl

[3] Jones, e. a.: *JSON Web Encryption*. [Online].
Retrieved from:
http://tools.ietf.org/html/draft-ietf-jose-json-web-encryption

[4] Jones, e. a.: *JSON Web Keys*. [Online].
Retrieved from: http://tools.ietf.org/html/draft-ietf-jose-json-web-key

[5] Jones, e. a.: *JSON Web Signing*. [Online].
Retrieved from:
http://tools.ietf.org/html/draft-ietf-jose-json-web-signature

[6] Jones, e. a.: *JSON Web Tokens*. [Online].
Retrieved from:
https://tools.ietf.org/html/draft-ietf-oauth-json-web-token

[7] Jones, e. a.: *JWT Authorization Grants*. [Online].
Retrieved from: http://tools.ietf.org/html/draft-ietf-oauth-jwt-bearer

[8] Lehtinen, P.: *jansson*. [Online].
Retrieved from: https://github.com/akheron/jansson

[9] McCallum, N.: *HTTP-parser*. [Online].
Retrieved from: https://github.com/nodejs/http-parser

[10] McCallum, N.: *jose*. [Online].
Retrieved from: https://github.com/latchset/jose

[11] Poettering, L.; Sievers, K.; Hoyer, H.; et al.: *systemd*. [Online].
Retrieved from: https://en.wikipedia.org/wiki/Systemd