Securing WireGuard private keys with a hardware token

Peter Van Eenoo

A Master's Capstone
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Cyber Security Engineering

University of Washington

2022

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Securing WireGuard private keys with a hardware token

Peter Van Eenoo

Chair of the Supervisory Committee:
Dr. Brent Lagesse
Computing & Software Systems

The WireGuard VPN has had a successful mainstream adoption, as evidenced by its recent inclusion in many open-source operating systems as well as a recent native Windows kernel module[1]. However key management is intentionally left up to the users and WireGuard does not use x509 certificates or the traditional PKI infrastructure. This causes a problem because the client key is not easily associated with a user or system and cannot be revoked, except by removing the client's public key from all servers. The client and server's public key consist of a single 256-bit public key based on Curve25519 which must be pre-shared with the server and client respectively before a successful handshake can be made. On a Linux laptop for example, the private key is held in a plaintext file along with the server's public key and IP address. Securing this file is left up to the user and it would be an easy target for malware or a malicious party to steal. The WireGuard protocol implements perfect forward secrecy, meaning loss of a private key still won't allow past conversations to be decrypted. However, secure key management is still very important because loss of the client's private key would result in two major vulnerabilities. First, it would allow an attacker to connect the target VPN undetected, masquerading as a legitimate client for any system where the private key is trusted. Secondly, it would enable the attacker to perform a persistent DoS attack against the client's connection to the server. If malware were programmed to look for WireGuard configuration files, collecting the keys and connecting to

remote systems would be trivial for attackers and hard to detect for system administrators.

TABLE OF CONTENTS

Page

LIST OF FIGURES

GLOSSARY

SECURITY KEY: A device that securely stores cryptographic keys and restrict access through a limited interface. Nitrokey Start[3] or the more common YubiKey[4], [5], are a class of security keys that implement a user-controlled cryptographic authenticator which are resistant to malware and phishing attacks and are essentially a hardware security module (HSM). These devices typically connect to a computer or smartphone via USB or NFC

WIREGUARD: a VPN technology proposed in 2017 that operates at the network layer, which aims to replace popular TLS-based VPNs like OpenVPN and IPsec. WireGuard has been described as "crypto-opinionated", meaning the WireGuard protocol supports only one cryptographic primitive for each cryptographic requirement. There is no support for negotiation of cryptographic parameters. For example, WireGuard only supports ChaCha20 for symmetric encryption[7] of data and Curve25519 key pairs for client authentication.

CURVE25519: digital signature operations are referred to as Ed25519. Key-signing operations are referred to as x25519. This distinction is important because WireGuard uses only x25519 operations for Elliptic-curve Diffie–Hellman (ECDH) key exchange, when generating the symmetric cipher key.

Chapter 1 INTRODUCTION

The main reason we write is to communicate

The other reason is for street cred

1.1 Background

Background is important to have for big papers Provide background here

1.2 Contributions

The Curve25519 algorithm is being used in an increasing number of open-source and commercial products[11] from SSH to Signal, however hardware support in popular security keys is currently almost non-existent. YubiKey is currently the most popular security key manufacturer however all of their current products lack support for x255191. I hope my project will encourage more manufactures to add support for x25519 in their security keys.

1.3 Methodology

In order to evaluate what I have done, I will do XYZ and present my findings

Chapter 2

BIG NEW SECTION

2.1 Introduction

Here is section 1

2.1.1 my subsection

Minor points

Chapter 3

DATA1

Chapter 4
DISCUSSION OF RESULTS

4.1 Introduction

I will discuss the limitations of my project as well as the technical hurdles that I faced while implementing and evaluation my project.

4.2 Limitations

I had initially hoped that my project could include mobile operating systems such as a smart phone. Users can easily use security keys with mobile operationg systems but due to technical requirements of the WireGuard protocal, a user would have to leave their security key connected to their smartphone perminitly, which could only be accomplishing using duct-tape for NFC devices or leaving a usb device inserted into their USB port, which seems like a hazard for the device. This always-connected requirement exists because the WireGuard protocol performs a rekey operation between peers at a default 120 seconds or after 2^{60} messages are exchanged. I could require users to modify their WireGuard server and client to default to a much longer timeout for rekey but this seems like a burdensome approch and it's not possible in all situations for the client to modify server settings.

More issues discussed here

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Future Research

What I think future research would look like