Anysphere: Private Communication in Practice

Security Whitepaper

Arvid Lunnemark Shengtong Zhang Sualeh Asif
{arvid,stzh1555,sualeh}@anysphere.co

June 30, 2022

Last updated: June 16, 2022

ABSTRACT

We describe Anysphere, a metadata-hiding communication system in practice. We discuss the theoretical protocol as well as security-critical implementation details. The goal of this whitepaper is to describe the threat model, and show how Anysphere guarantees privacy against all threats within the model.

CONTENTS

1 INTRODUCTION

When the internet was first established, everything sent over it was public. If A sent a message to B, anyone on their path through the internet could see that such a message was sent, and read the actual message. As of today, many messaging services are end-to-end encrypted, meaning that no one can read the contents of messages. However, for sufficiently powerful adversaries, it is still possible to find out who is talking to whom. The goal of private communication is to hide this metadata: we want to create a system where A and B can send messages to each other over an untrusted network, without anyone knowing that they talk to each other. Such a private communication system would be critically important to protect and expand freedom in the world [Lun21].

Anysphere is a provably secure metadata-hiding communication platform, built for the real world. In this document, we outline how we guarantee theoretical security, and how we make sure that the theoretical security translates to good practical security.

2 THREAT MODEL

Touch on: server, friends, client-side computer, etc.

3 CLIENT-SIDE SECURITY IN PRACTICE

Our theoretical threat model assumes that the user's local computer is completely trusted, and that it is running a correct implementation of our protocol. If your computer is compromised, or you are running a buggy or intentionally incorrect version of our code, none of our previously outlined theoretical guarantees will apply. This is inherent and unavoidable — no matter the fancy encryption schemes you come up with, nothing will help you if your computer comes with a preinstalled backdoor. While **we fundamentally cannot eliminate the client-side risk**, we can *reduce* it, which we will do in this section.

3.1 Reducing the attack surface

The first step in mitigating security is to reduce the attack surface. To do so, we architected our client to consist of two parts: a UI frontend and a daemon backend, where the daemon backend contains all security-critical code. We sandbox the UI frontend in such a way that it is not allowed to talk to the internet, and let all message sending go through the daemon, which handles the cryptography. That way, even if there are bugs in the UI frontend, or potentially malicious code, there is not much it can do.

[**TODO**: Figure of client architecture with UI and daemon, showing how we cut off internet access.]

We also reduce the attack surface of the daemon itself. In particular, we use C++ instead of other popular languages (Rust, Go, Python), because all other practical languages are significantly more susceptible to supply chain attacks. Our daemon has 4 direct dependencies (Abseil, gRPC, SQLite, Libsodium) and 0 transitive dependencies. A comparable implementation in a language with a package manager would easily use 100s of transitive dependencies. We elaborate more on our choice of C++ in this blog post [**TODO**: Link.].

3.2 Code distribution

We sign everything.

[**TODO**: Either understand whether standard OS signing is good enough, or whether we should sign things ourselves.]

3.3 Updates

[**TODO**: Write how we verify the signature every time here.]

Arvid Lunnemark Shengtong Zhang Sualeh Asif
{arvid,stzh1555,sualeh}@anysphere.co

June 30, 2022

Last updated: June 16, 2022

3.4 Protecting against non-privileged local malware

If you've granted administrator access to a malicious program on your computer, there is, unfortunately, nothing to be done. We can, nevertheless, reduce the risk of non-privileged malware.

[**TODO**: Actually implement: allow to encrypt the database, in which case the both the GUI and the CLI need to require passwords (and the GUI may cache the password for some amount of time).]

Again, we do not aim to eliminate the risk here. Non-privileged malware may still gather information from side-channel attacks, and potentially other avenues. Once an attacker has access to your computer, it is very, very hard to shield yourself from them.

REFERENCES

[Lun21] Arvid Lunnemark. *It Is Time to Move Beyond End-to-End Encryption.* 2021. URL: https://arvid.xyz/posts/it-is-time-to-move-beyond-end-to-end-encryption/ (visited on 11/27/2021).