Richtlinie zum „Software-Sprint“

Rosenpass: post-quantum VPN mit minimalen Privilegien

Schlussbericht

Zuwendungsempfänger:

Schmidt Schütze Varner Zaeske GbR

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS23S30 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Was war Deine Motivation? Welches Problem wolltest Du mit Deinem Projekt lösen? Wie war die geplante Vorgehensweise zur Problemlösung (auch Angabe der wichtigsten Meilensteine)?

Die fortschreitende Entwicklung von Quantencomputern stellt eine erhebliche Gefahr für die Datensicherheit aller Internetnutzer*innen dar. Sobald ein ausreichend fortgeschrittener Quantencomputer entwickelt ist, wird es möglich sein mit diesem Gerät den Großteil der verschlüsselten Kommunikation im Internet zu entschlüsseln ohne die entsprechende Befugnis zu besitzen. Alternative Grundbausteine für Verschlüsselungsverfahren ohne diese Anfälligkeit stehen zur Verfügung und die Migration hin zu diesen neuen Verfahren sollte sofort durchgeführt werden, da Angreifer bereits jetzt von der Möglichkeit Gebrauch machen, wichtige private Daten und Geheimnisse zwischenzuspeichern bis diese entschlüsselt werden können.

Rosenpass adressiert einen Teilbereich dieser übergeordneten Infrastrukturaufgabe: Die Absicherung des weit verbreiteten WireGuard Protokolls gegen Quantenangriffe. Dieses Ziel wurde bereits vor dem Beginn dieses Förderprojektes mit dem initialen Release im Jahre 2023 erreicht, wodurch Rosenpass zur einzigen produktionsreifen Lösung für quantensichere VPNs wurde.

Obwohl Rosenpass bereits seit einem Jahr als Lösung bereit steht ist noch viel zu tun um die Software in jedem Umfeld nutzbar zu machen; einer der wichtigsten Bereiche ist dabei der Schutz vor Sicherheitslücken durch Softwarefehler: Rosenpass dient dazu, bestimmte Sicherheitslücken zu schließen. Allerdings ist die Installation neuer Software immer mit einem gewissen Risiko verbunden, da der neue Code seinerseits sicherheitsrelevante Fehler enthalten kann. Dieser Umstand ist kritisch da Rosenpass benötigt besondere Systemprivilegien benötigt um eine VPN-Verbindung etablieren zu können, sich aber gleichzeitig an besonders sicherheitsbewusste Nutzer*innen richtet.

Ziel unseres Förderprojektes war es vor allem, Technologien zu integrieren, die die Auswirkungen potenzieller Sicherheitslücken minimieren. Dabei setzten wir auf Broker; kleine Applikationen die, im

Auftrag der Hauptapplikationen, klar umgrenzte Operationen durchführen. Diese sind in der Lage das Risiko von Sicherheitslücken weit über den heutigen Stand der Technik hinaus zu adressieren.

Viel der hierfür benötigten technologischen Verbesserung dreht sich letztlich um die Entwicklung von Kommunikationsschnittstellen. Im Zuge des Projektes entschlossen wir uns daher, ebenfalls die Entwicklung von Schnittstellen für externe Anwendungen in den Blick zu nehmen, die es Drittentwickler*innen möglich machen, Rosenpass mitsamt aller Hochsicherheitseigenschaften innerhalb ihrer Applikationen zu nutzen.

Dies sollte für Rosenpass einen Schritt von einer allein nutzerorientieren Anwendung hin zu einer Basistechnologie für viele Softwareprojekte darstellen.

Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint“

Welche Bezüge gibt es zu den Themenfeldern „Civic Tech“ und „Data Literacy“ des Software Sprints oder zu weiteren gesellschaftlich relevanten Zielen bzw. Lösungsansätzen?

Rosenpass als Basistechnologie strebt an, allen Bürger*innen die Möglichkeit zu geben, ihre Daten langfristig zu schützen. Rosenpass kann bereits jetzt genutzt werden um Datenverbindungen zu realisieren, die Sicherheit gegen Angriffe von Quantencomputern bieten. Damit stellt die Applikation eine Basis dar, mit der weitere Projekte im Bereich "Civic Tech" mit hohen Anforderungen an Datenschutz umgesetzt werden können.

Als unabhängige Gruppierung, bestehend aus Experten aus Industrie, Open-Source, Hackerszene, und akademischer Kryptografieforschung, stellen wir ein Forum dar, in dem die genannten Bereiche gemeinsam die beste Technologie für Nutzer*innen entwickeln können.

Besonderes Augenmerk legen wir dabei auf die populärwissenschaftliche Kommunikation unserer Ergebnisse. Im Auge der meisten Bürger*innen gilt Kryptografie weiterhin als Geheimwissenschaft, was unter anderem auch an der Tendenz Forschender liegt, sich bei der Präsentation ihrer kryptografischen Konstruktionen mehr auf die Funktionsweise zu konzentrieren und weniger auf die Use-Cases, in denen ihre Technologie angewandt wird. Um diese Problematik zu adressieren, setzen wir auf multimodale Lernkonzepte um Nutzungsszenarien klar zu kommunizieren: Welche Personen könnten motiviert sein Rosenpass einzusetzen und was ist die Geschichte hinter ihrer Motivation?

Ausführliche Darstellung der Ergebnisse

Welche konkreten Ergebnisse hast Du erzielt? Konnten alle Meilensteine erreicht werden? Welche zusätzlichen Erkenntnisse hast Du aus der Projektarbeit gewonnen, auch im Hinblick auf die Begleitung durch die Open Knowledge Foundation?

Erstellen einer Rust-Softwarebibliothek für das Deserialisieren von Netzwerkpaketen in kryptographischen Anwendungen

Während der Entwicklung von Rosenpass wurde eine Bibliothek benötigt um Daten mit kryptografisch sensitiven Inhalten effektiv serialisieren zu können. Übliche Serialisierungsframeworks für Rust, wie z.B. Serde, stellten sich als unbrauchbar für diesen Zweck heraus, da sie nicht für die Verarbeitung von kryptografischen Schlüsseln konzipiert waren. Unter anderem erstellen solche

Bibliotheken häufig Kopien der Daten, die eingelesen oder ausgelesen werden. Der hierfür benutzte Speicher wird zwar hinterher wieder freigegeben, jedoch ohne die Kopien der Daten vorher sicher zu löschen. Für kryptografisches Material bedeutet dies, dass Schlüsselmaterial im schlimmsten Fall in die Hände von Angreifern gelangen könnte.

Um dieses Problem zu beheben, setzten wir ursprünglich auf die "Transmute" Crate, die es erlaubt Speicherblöcke als Rust-Strukturen neuzudeuten. Diese Umdeutung jedoch ist risikoreich, denn die Transmute Crate garantiert nicht, dass die umgedeuteten Datenblöcke den korrekten Inhalt enthalten. Die Daten müssen vorher validiert werden was leicht zu Fehlern führen kann. Wenn die betroffenen Daten aus dem Netzwerk gelesen werden, können mögliche, bei der Validierung entstandene, Fehler von Angreifern genutzt werden und dadurch eine Sicherheitslücke darstellen.

Um die Sicherheit unserer Implementierung zu erhöhen, beschlossen wir stattdessen eine eigene Lösung zum Serialisieren und Deserialisieren zu entwickeln. Die Lenses Library wurde für diesen Zweck entwickelt und setzt auf einen Mechanismus zur automatischen Codegenerierung mit dem der direkte Zugriff auf strukturierte Daten, ohne diese vorher zu Rust-Strukturen umzuwandeln, sich deutlich ergonomischer handhaben lies. Damit wurde ein erheblicher Zugewinn an Sicherheit im Rosenpass-Projekt erreicht.

Um den komplexeren Anforderungen der Broker- und API-Features die als Teil des PTF-14-Projektes geplant waren gerecht zu werden, sollte als Teil dieses Projektes auch die Lense-Library erhebliche Verbesserungen erhalten.

Zunächst migrierten wir die Library in eine eigene Softwarebibliothek und passten sie an, um sie auch für andere Entwickler*innen nutzbar zu machen. Nachdem wir die Schnittstellenspezifikationen für die Broker und API auf Papier erstellt hatten, begannen wir auszuarbeiten, welche Änderungen an der Lensesbibliothek nötig wären um diese Schnittstellen einzubauen:

Ein System sollte gebaut werden, mit dem Rust-Datenstrukturen analysiert und in ihre Einzelteile zerlegt werden können. Aus dieser Beschreibung sollte die Bibliothek eine weitere Datenstruktur erzeugen, die jedem Eintrag einen Speicherplatz innerhalb eines übergeordneten Speicherblocks zuweist. Die gesamte Struktur kann als Ganzes in einem vorher bereits reservierten Speicherbereich hineindecodiert werden oder in Teilen einzeln encodiert/decodiert werden; letzterer Modus bietet mehr Effizienz, da ausschließlich die relevanten Daten verarbeitet werden.

Leider überstieg die Implementierung dieses Mechanismus das Zeitbudget, das uns für das PTF-14 Projekt zur Verfügung steht bei Weitem.

Zunächst versuchten wir die API- und Broker-Features mit den existierenden Mitteln der nun ordentlich paketierten Lenses-Bibliothek umzusetzten. Leider stellte sich dieser Ansatz als unzureichend heraus. Glücklicherweise wies uns eine Kollege vom Max-Planck-Institut für Sicherheit und Privatsphäre auf die "Zerocopy" Crate hin, die ähnliche wie "Transmute", Speicherblöcke in begrenztem Rahmen zu Rust-Strukturen konvertieren kann. Anders als Transmute setzt Zerocopy auf computergestützte Korrektheitsbeweise, die den Arbeitsaufwand zur Verifikation der Daten erheblich verringern. Der Kollege begann auch direkt selbst damit, die Zerocopy Crate in Rosenpass zu integrieren. Die "Zerocopy"-Bibliothek stellte sich als ausreichend heraus, um die Anforderungen der Broker- und API-Features in einer ersten Version zu implementieren.

Die Weiterentwicklung der Lenses-Bibliothek sollte dennoch zu einem späteren Zeitpunkt weitergeführt werden. Insbesondere im Umgang mit Datenstrukturen dynamischer Größe bietet die hier ausgearbeitete Architektur erhebliche Vorteile. Die Zerocopy-Bibliothek kann bei der Weiterentwicklung genutzt werden, um die Entwicklungsarbeit innerhalb der Lenses-Bibliothek zu verringern.

Fazit: Das Projektziel wurde mit der Migration der Lenses-Bibliothek zu einer eigenen Crate erreicht. Darüber hinaus wurde eine Softwarearchitektur für eine erhebliche Weiterentwicklung der Bibliothek entwickelt, die zukünftig angegangen werden kann. Die Zerocopy-Crate wurde integriert um die anderen Projektziele des PTF-14-Projektes ohne erheblichen Mehraufwand möglich zu machen.

Ansteuerung von Rosenpass durch einen Netzwerksocket ermöglichen

Ziel dieses Projektes war es, das Sicherheitsrisiko durch Softwarefehler in Rosenpass zu minimieren. Dazu sollte Rosenpass in mehrere Komponenten aufgespalten werden, die jeweils mit den für ihre Funktionalität minimalen Privilegien arbeiten.

Während zuvor ein Fehler in jedem Teil der Software zu erheblichen Sicherheitsrisiken führen konnte, sollten Fehler hinterher lediglich Risiken für den konkreten Aufgabenbereich eines Moduls bedeuten.

Wo vorher nur eine Sicherheitsdomäne existierte, die sich intern mit Standard-Programmiertechniken koordinieren konnte, existieren nun etliche Domänen, die miteinander kommunizieren müssen. Um dies zu ermöglichen musste eine Architektur geschaffen werden, die die Kommunikation an sich möglich macht.

Wir beschlossen, Softwarekomponenten maximal einfach zu halten um die Sicherheitsrisiken im Falle eines Softwarefehlers zu minimalisieren. Um dennoch ein hohes Maß an Flexibilität zu ermöglichen, beschlossen wir, zwischen den Komponenten Übersetzungsmodule einzubauen. Diese Module ermöglichen es uns, mehrere Verbindungen gleichzeitig zu verarbeiten und weitere Serialisierungsformate anzubieten, damit Drittnutzer*innen leicht mit den Komponenten interagieren können.

Ein Mechanismus zu Aufbau von Kontrollverbindungen wurde in Rosenpass integriert und ein Broker zum Umgang mit symmetrischen Schlüsseln wurde eingebaut.

Fazit: Das Projektziel wurde teilweise erreicht. Die grundlegende Kommunikationsinfrastruktur wurde eingebaut. Schnittstellen-Endpunkte, die über das Verarbeiten von symmetrischen Schlüsseln hinausgehen wurden aus Zeitgründen nicht integriert.

Config-Broker erstellen, der eine Konfigurationsdatei über das Fernsteuerungsfeature in Rosenpass einspielt

Das Projektziel wurde aufgrund von Zeit- und Ressourcenmangel nicht adressiert. Die Grundlage hierfür wurde jedoch mit dem Aufbau der hierfür nötigen Kommunikationsinfrastruktur geschaffen.

Key-Broker erstellen, der Schlüssel von Rosenpass erhält und diese in WireGuard einspielt

Rosenpass benutzt erhöhte Privilegien ausschließlich um die generierten Schlüssel an WireGuard senden zu können. Das wichtigste Ziel unseres Projektes war daher die Entwicklung eines Vermittlers (Brokers), der die Weiterleitung von symmetrischen Schlüsseln an WireGuard übernimmt.

Dieser Vermittler wurde entwickelt um die im Bertireb entstehenden Sicherheitsrisiken zu minimieren indem er die Implementierung auf wenige hundert Zeilen einfachen Programmcodes beschränkt. Dies war möglich durch den Einsatz eines Zwischenvermittlers, der Aufgaben wie die Annahme mehrerer Kommunikationsverbindungen zum Prozess mit erhöhten Privilegien übernimmt.

Darüber hinaus investierten wir Zeit in die Recherche, wie die Privilegien des Vermittlerprozesses selbst reduziert werden könnten. Als Resultat dieser Nachforschungen beschlossen wir, das Linuxprivileg NET_ADMIN anstelle von vollen Administratorprivilegien zu nutzen.

Fazit: Das Projektziel wurde erreicht. Über die Erstellung eines Brokers zum Weiterleiten der symmetrischen Schlüssel hinaus wurden weitere Maßnahmen ergriffen um die genutzten privilegien zu reduzieren.

Docker-basiertes Isolationswerkzeug für die Komponenten (Aufteilung in mehrere Container)

Ein dockerbasiertes Isolationswerkzeug mit extrem wenig Privilegien konnte erstellt werden. Zur Umsetzung nutzen wir Podman statt Docker, da Podman über zusätzliche Sicherheitsfeatures verfügt.

Fazit: Das Projektziel wurde erreicht. Weitergehende Maßnahmen zur Absicherung des Isolationswerkzeuges wurden getroffen.

Hardwarebasiertes Isolationswerkzeug für die Komponenten (Aufteilung zwischen mehreren Computern)

Mithilfe der Netzwerkanschlussweiterleitungsfunktion von OpenSSH und Socat kann eine Verbindung zwischen WireGuard Schlüsselvermittler und dem Rosenpasshauptprozess hergestellt werden. Hierdurch kann eine Konfiguration mit hardwarebasierter Isolation hergestellt werden. Im Entwicklungszweig für die vermittlerbasierte Architektur befindet sich eine Anleitung zum Reproduzieren dieser Konfiguration mitsamt einer Analyse der hierbei erreichten Isolationseigenschaften und Ansätzen zum weiteren Absichern der Verbindung zwischen den zwei Geräten.

Fazit: Das Projektziel wurde erreicht. Eine weitergehende Sicherheitsanalyse wurde ebenfalls erstellt.

Micro-Virtual Machine - basierte Isolation der Komponenten mit RustyHermit

Standardmäßig erfolgt die Isolation von Sicherheitsdomänen in Rosenpass auf dem selben System mit Hilfe von Betriebssystem-Prozessen. Virtuelle Maschinen bieten eine besser geschützte Isolationsschicht, doch um diese Werkzeuge zu nutzen musste früher ein zusätzliches Serverbetriebssystem ausgeführt werden, was einen erheblichen Overhead erzeugte.

Micro-Virtuelle-Maschienen nutzen speziell angepasste Betriebssysteme, die für die Ausführung einer einzelnen Applikation ausgelegt sind und damit einen wesentlich geringeren Ressourcenverbrauch aufweisen.

Ziel dieser Projektkomponente war es, einzelne Sicherheitsdomänen in Rosenpass in eigenen virtuellen Maschinen mit Hilfe des Hermit-Betriebssystems für Micro-VMs zu betreiben.

Während der Integration stellte sich heraus, dass Hermit derzeit asynchrone Lese- und Schreiboperationen nicht unterstützt. Dies ist ein Feature, das von Rosenpass genutzt wird um die vielen Verbindungen, mit denen die Applikation gleichzeitig umgehen muss, effizient zu verarbeiten. Um diese Hürde zu überwinden wurde Kontakt mit dem Entwicklerteam von Hermit an der RWTH Aachen aufgenommen. Eine Einbettung der fehlenden Features wurde bereits begonnen. Die Integration von Hermit wird weitergeführt, sobald die notwendigen Fähigkeiten in Hermit eingebaut wurden.

Fazit: Dieses Projektziel konnte aufgrund von fehlenden technischen Voraussetzungen nicht fertiggestellt werden. Kontakt mit den Entwickler*innen von Hermit wurde aufgenommen. Die Integration wird fertiggestellt sobald die Vorraussetzungen hierfür geschaffen wurden.

Dokumentation und Tutorials

Wir haben unserer Website rosenpass.eu den neuen Bereich "Documentation" hinzugefügt. Dort finden sich, neben dem zuvor schon vorhandenen Whitepaper, nun eine Reihe von neuen Inhalten: eine Anleitung um Rosenpass selbst zu kompilieren, Release Notes, Aufzeichnungen und Folien der unserer Talks auf Konferenzen und weitere Anleitungen.

Die "Quick Start"-Anleitung wurde noch benutzerfreundlicher gestaltet und aktualisiert. Die in "What is Rosenpass?" enthaltenen Erklärungen sind jetzt noch verständlicher strukturiert und bebildert.

Außerdem haben wir eine Reihe von Blogeinträgen in deutscher und englischer Sprache geschrieben, die unter anderem unsere Arbeit am PTF-14-Projekt dokumentieren. Diese haben wir auch auf den sozialen Medien verbreitet. Die Blogeinträge enthalten unter anderem Infografiken, die die Zielsetzung unseres PTF-Vorhabens und das daraus entstehende System aus Einzelprozessen visualisieren.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Welcher Nutzen ergibt sich für die Zielgruppe aus den Ergebnissen Deines Projekts? Welche weitergehenden Effekte ergeben sich aus der Open-Source-Stellung der Ergebnisse? Gibt es Ideen für die Weiterentwicklung Deiner Lösung und Pläne zu deren Umsetzung? Hat die Arbeit in dem Projekt Dich in Deiner persönlichen, fachlichen Weiterentwicklung unterstützt?

Im Sinne der oben dargestellten Zielsetzung sprechen wir mit dem hier geschilderten Entwicklungsschritt Softwaredistributoren und Anwendungsentwickler*innen an. Mit der hier durchgeführten Arbeit wurde die Barriere zum Einsatz unserer Software verringert.

Rosenpass wird im Wege der agilen Softwareentwicklung konstruiert. Die im Projekt addressierten Features sind für Rosenpass als Projekt unabdingbar und werden Schritt für Schritt weiterverfolgt. Eine Vielzahl an weitreichenden technischen Infrastrukturentwicklungen wurden konzeptioniert.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Gab es Arbeiten bzw. Lösungsansätze, die nicht weiter verfolgt wurden? Was waren die Hintergründe, und wie bist Du alternativ vorgegangen?

Die wesentlichen Projektziele wurden erreicht. Insbesondere die Machbarkeit von vermittlerbasierter Isolation wurde bewiesen und die Eignung der Technik mit verschiedenen besonders sicheren Isolationsschichten wurde erprobt.

Die Arbeiten wurden nur teilweise in den produktionsreifen Zweig der Entwicklersoftware integriert, da zusätzliche Verbesserungen in Benutzbarkeit und plattformübergreifender Kompatibilität vor der Produktionsreife nötig sind.

Das Einspielen von Konfigurationsdateien mittels einer Programmierschnittstelle wurde aus Zeitgründen nicht integriert.

Anstatt primär auf eine Verbesserung der Lense-Library zu setzen, um die Serialisierungs- und Deserialisierungsanforderungen zu erfüllen, beschlossen wir, die Zerocop-Bibliothek einzusetzen. Dies war zeiteffizienter.

Die Integration des Hermit Microkernels konnte aufgrund von fehlenden technischen Voraussetzungen nicht fertiggestellt werden. Kontakt mit den Entwickler*innen von Hermit wurde aufgenommen.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Wo können sich Interessenten detailliert über Deine Projektergebnisse informieren (z.B. Webseite, GHitHub, Veröffentlichungen)?

Nutzer*innen finden eine Vielzahl von Informationsmaterialien auf https://rosenpass.eu und https://github.com/rosenpass/rosenpass

Der Resultate dieses Förderprojekts die noch nicht in den Hauptentwicklungszweig integriert wurden finden sich vor allem im Entwicklungszweig "dev/broker-architecture"

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Gab es im Projektverlauf Ereignisse, die eine Anpassung der Planung erforderlich machten – z.B. Mehr- oder Minderaufwand bei der Bearbeitung von Teilaufgaben?

Die Projektmitglieder Stephan Schütze und Wanja Zaeske konnten aus privaten Gründen wesentlich weniger Zeit im Förderprojekt aufwenden als anfänglich geplant. Karolin Varner und Lisa Schmidt konnten ihren Ausfall nur teilweise kompensieren, zumal der Projektplan von Anfang an ambitioniert war.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Gab es Entwicklungen anderer Personen oder Institutionen, die Einfluss auf Deine Arbeiten und die Zielsetzung hatten? Wenn ja, worin bestand dieser und wie bist Du damit umgegangen?

Die Ergebnisse dieses Förderprojektes werden von einigen Seiten stark antizipiert. Obwohl die Projektresultate noch nicht vorliegen, befinden sich mehrere externe Integrationsprojekte in Arbeit. Eine nutzerorientierte VPN-Lösung hat Rosenpass bereits integriert. Eine Arbeitsgruppe bestehend aus mehreren Forschenden wurde gestartet, um die Integration von vermittlerbasierten Isolationswerkzeugen und die Entwicklung einer Entwicklerschnittstelle für Rosenpass voranzubringen.