Richtlinie zum „Software-Sprint“

Sequoia – High-Level Bibliothek für Sequoia PGP

Schlussbericht

Zuwendungsempfänger*in:

Sequoia PGP GbR

Neal H. Walfield und Justus Winter

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Forschung, Technologie und Raumfahrt unter dem Förderkennzeichen 16IS25S17 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei dem*der Autor*in.

Kurze Darstellung der Aufgabenstellung und Motivation in der Förderphase

OpenPGP ist ein weit verbreitetes dezentrales Internetprotokoll zur Verschlüsselung, Signieren und Verwaltung von Schlüsseln. Seit 1991 schützt es die Kommunikation von Aktivisten, Privatpersonen und Unternehmen und sichert die Software-Lieferkette.

OpenPGP-Software hat jedoch den Ruf, für Privatanwender und Entwickler schwer zu handhaben zu sein. Ziel dieses Projekts ist es deshalb, eine High-Level-Bibliothek für Sequoia PGP zu entwerfen und zu implementieren. Im Gegensatz zu einer Low-Level-API, bietet eine High-Level-API einige wenige Funktionen für allgemeine Arbeitsabläufe und verbirgt technische Details. Dies vereinfacht die Integration für Entwickler und führt zu einer konsistenteren User Experience (UX). Dies ist nicht nur für die Benutzerfreundlichkeit, sondern auch für die Sicherheit von Vorteil, da Benutzer weniger Fehler machen, wenn sie mit vertrauten Vorgängen arbeiten.

Die Entwicklung hatte einige Meilensteine. Zunächst haben wir das Fundament und die Infrastruktur der Bibliothek aufgebaut. Danach haben wir die einzelnen Workflows aus sq, zu der Bibliothek portiert. Auch wenn ein großer Teil der Implementierung schon vorhanden war, bestand die Herausforderung darin, eine geschickte API zu gestalten. Dabei haben wir sq angepasst, um die neue API der Bibliothek zu nutzen und dabei die API zu evaluieren. Zum Ursprunglichen Plan eine Demo zu erstellen, sind wir nicht gekommen.

Beitrag des Projektes zu den Zielen der Förderinitiative „Software Sprint“

Sequoia ist ein sicherheitsrelevantes Projekt, das von der Sovereign Tech Agency als kritische Internetinfrastruktur eingestuft wird. Sequoia wird beispielsweise von vielen Linux-Distributionen - darunter Debian, Ubuntu, Fedora und Red Hat Enterprise Linux - genutzt, um Software-Updates zu sichern.

Sequoia basiert auf OpenPGP. OpenPGP wird von Aktivisten, Journalisten sowie Anwälten und deren Klienten verwendet, um ihre Kommunikation zu schützen. Heute kommt Sequoia auch in diesem Bereich zum Einsatz. So wird es von der Whistleblowing-Plattform SecureDrop verwendet. Diese wird von The New York Times, The Guardian und Der Spiegel genutzt um Quellen zu schützen. Auch die Whistleblowing-Plattform HushLine nutzt Sequoia.

Das Projekt, eine High-Level-API zu entwickeln, erleichtert die Integration von Sequoia in viele Anwendungen. Dadurch wird Sequoia hoffentlich weiter verbreitet, und mehr Menschen werden von seinem Schutz profitieren.

Ausführliche Darstellung der Ergebnisse in der Förderphase

Das Ziel des Projekts war die Entwicklung einer High-Level-API auf Basis der Workflows in unserem Commando-Ziele-Tool, sq. Bevor wir mit der Entwicklung des High-Level-APIs beginnen konnten, mussten wir zunächst das Fundament legen und die Infrastruktur aufbauen.

Wir haben mit einem Objekt namens „Sequoia” begonnen. Dieses Objekt dient als eine Art Session. Es beinhaltet die Konfiguration, die kryptographische Richtlinie sowie Handles zum Zertifikats- und Schlüsselspeicher. Mithilfe des „Sequoia”-Objekts kann der Nutzer die Funktionalität nutzen, ohne viele Parameter angeben zu müssen.

Beim Instanziieren eines „Sequoia”-Objekts wird zunächst die Konfigurationsdatei eingelesen. Dieses Verhalten kann unterdrückt werden, um eine stateless Instanz zu erzeugen, oder geändert werden, um eine alternative

Konfiguration zu nutzen. Abhängig von der Konfiguration werden zusätzliche Initialisierungen durchgeführt. Das Einlesen sonstiger Dateien, wie etwa Keyrings, erfolgt bei Bedarf, um die Initialisierungszeit zu verkürzen.

Der Configuration Parser wurde von sq übernommen, aber verfeinert. Die Schnittstelle wurde überarbeitet, sodass sie die Anforderungen einer Bibliothek erfüllt. So gibt es beispielsweise statt der direkten Verwendung von Feldern jetzt Getter-Funktionen. Darüber hinaus wurde die Konfiguration von der CLI-Argumentparsing entkoppelt.

Was die Infrastruktur betrifft, so wurden viele Typen und Funktionen aus sq in die Bibliothek portiert. Dazu gehören Funktionen, um ein Zertifikat anhand seines Fingerprints oder seiner User-ID zu finden, eine repräsentative User-ID für ein Zertifikat unter Berücksichtigung der vorhandenen Authentifizierungsinformationen zu ermitteln, das Handling von Secret Keys, und die Zertifikatsprovenienz zu verwalten. Zu den Typen zählen die Spezifikation von Algorithmen und Profilen, die Spezifikation der aktuellen Zeit und eines Ablaufdatums, sowie die Spezifikation von Queries. Wie beim Konfigurationsobjekt musste auch die API oft angepasst werden.

Als Nächstes musste ein neues Prompt-System entwickelt werden. Benutzer werden bei Bedarf zur Eingabe von Passwörtern aufgefordert. Dies ist beispielsweise beim Entschlüsseln einer Nachricht oder beim Generieren von Schlüsseln der Fall. Wenn eine Smartcard benötigt wird, werden sie ebenfalls aufgefordert, diese mit dem System zu verbinden. Da ein sq-Benutzer eine Person ist, die am Terminal arbeitet, erfolgte dies entsprechend über das Terminal. Eine Anwendung, die unsere neue Bibliothek nutzt, muss jedoch keine Terminal-Anwendung sein und wird nicht zwingend von einer Person ausgeführt.

Um diesen neuen Anforderungen gerecht zu werden, haben wir ein Prompt-System entwickelt, bei dem die Darstellung des Prompts und das Abfragen der Nutzereingabe vollständig vom Beschreiben des Prompts entkoppelt sind. Dies funktioniert wie folgt: Wenn eine Funktion möglicherweise eine Eingabe vom Benutzer benötigt, nimmt die Funktion einen Prompt-Parameter. Dieser zusätzliche Parameter ist ein Objekt, das unter anderem, über eine Methode verfügt, die anhand einer Beschreibung dem Benutzer einen Prompt anzeigt.

Wenn die Bibliothek eine Eingabe vom Benutzer benötigt, z. B. wenn bei Signieren eine Datei ein Passwort benötigt wird, um das Schlüsselmaterial zu entschlüsseln, beschreibt die Bibliothek, was sie braucht, und ruft die Methode auf. Die Methode, die normalerweise von der Anwendung implementiert wird,

zeigt dem Benutzer das Prompt an und gibt die Eingabe an die Bibliothek zurück. Die Anwendung hat die komplette Kontrolle darüber, wie das Prompt aussieht. So kann beispielsweise festgelegt werden, ob es mit einem GUI-Widget oder in einem Terminal dargestellt wird oder ob es rein programmiert beantwortet wird. Die Beschreibung ist mächtig genug, um alle möglichen Szenarien abzudecken, bleibt dabei aber trotzdem einfach zu nutzen. Ferner kann die Schnittstelle bei Bedarf erweitert werden.

Nachdem das Fundament und die Infrastruktur fertiggestellt waren, war alles für die High-Level-API vorbereitet.

Wir erwarten, dass die High-Level-API häufig im interaktiven Rahmen genutzt wird. Da manche Operationen einige Zeit in Anspruch nehmen können, muss der Caller die Möglichkeit haben, die Ergebnisse bereits während der Operation zurückzugeben. So kann beispielsweise das Importieren eines großen Keyrings einige Sekunden dauern. Die Funktion kann nach jedem einzelnen Zertifikat den Caller informieren, dass ein Zertifikat importiert worden ist, und der Caller kann eine Fortschrittsanzeige aktualisieren.

Außerdem möchten viele Caller nicht nur ein einfaches Ergebnis, sondern auch Informationen darüber, was genau passiert ist. Beim Importieren eines Keyrings möchte der Caller möglicherweise nicht nur erfahren, ob alle Zertifikate importiert wurden, sondern auch, welche Zertifikate importiert wurden, ob ein Zertifikat bereits vorhanden war und ob es aktualisiert wurde. Diese Informationen hat die Funktion ohnehin parat und kann sie an den Caller weitergeben.

Um dieser Anforderung nachzukommen, nehmen die High-Level-Funktionen einen zusätzlichen Parameter entgegen. Dieser ist ein Objekt, das eine sogenannte Ereignis-Stream-Schnittstelle implementiert. Bei jedem wichtigen Schritt einer Funktion, wird ein Ereignis über dieses Objekt emittiert. Die Funktion, die Zertifikate importiert, emittiert vier unterschiedliche Ereignisse: Ein Zertifikat wurde erfolgreich importiert, es gab einen Fehler beim Importieren eines Zertifikats, eine Autocrypt-Nachricht wurde entschlüsselt und gescannt, und eine Zusammenfassung des Vorgangs.

Die Ereignisse sind so konzipiert, dass sie auf vorwärts- und rückwärtskompatible Weise erweitert werden können. Beispielsweise sind die Fehlermeldungen hierarchisch strukturiert. So kann beim Importieren eines Zertifikats leicht unterschieden werden, ob es sich um einen Systemfehler oder einen Fehler mit dem Zertifikat selbst handelt. Systemfehler und Fehler mit

einem Zertifikat sind dann weiter unterteilt. Soll eine neue Zertifikatfehlermeldung benötigt werden, kann diese der zweiten Kategorie zugeordnet werden. Anwendungen, die noch nicht aktualisiert wurden, können dennoch erkennen, dass es sich um einen Zertifikatfehler handelt, auch wenn sie die Details nicht verstehen.

Die Implementierung eines Stream-Objekts ist relativ unkompliziert. Uninteressante Ereignisse können ohne Aufwand einfach ignoriert werden. Wenn eine Anwendung die Ereignisse gar nicht benötigt, kann sie eine vom Bibliothek implementierte Stream Objekts benutzen, die alle Ereignisse ignoriert. Falls die Anwendung die Ereignisse erst im Nachhinein auswerten möchte, gibt es auch eine Implementierung die alle Ereignisse in einem Vektor sammelt.

sq verfügt über ungefähr 50 Workflows, die wir in der neuen Bibliothek implementieren wollten. Letztlich haben wir die Workflows für die folgenden Kommandos vollständig implementiert: `sq sign`, `sq verify`, `sq encrypt`, `sq decrypt`, `sq pki authenticate`, `sq pki lookup`, `sq pki identify`, `sq pki path`, `sq cert import`, `sq cert export`, und `sq key import`. Aus Mangel an Zeit konnten wir die folgenden Workflows nur teilweise implementieren: `sq pki link add`, `sq pki link authorize`, `sq pki vouch add`, `sq pki vouch authorize`, `sq key generate`, `sq inspect`, und `sq packet dump`.

Der Grund, warum wir das Projekt nicht vollständig abschließen konnten, ist, dass einer von uns krank geworden ist und für mehr als zwei Drittel des Projekts arbeitsunfähig war.

Auch wenn uns zu Beginn des Projekts bereits klar war, was wir machen wollten, kristallisierten sich die Details erst im Laufe der Implementierung heraus. Eine wichtige Erkenntnis ergab sich beispielsweise aus dem Design der Streaming-API. Ursprünglich beinhaltet die Stream-Typen viele Felder mit anonymen Typen. Das macht die Beschreibung des Typs zwar kompakter, erschwert aber die Erstellung von Funktionen, die nur einen Teil der Struktur nutzen. Das liegt daran, dass Funktionen keine Parameter mit anonymen Typen haben können. Entweder musste also zu viel Information weitergereicht werden oder die Information musste neu gemarshallt werden, was Boilerplate-Code bedeutete.

Die Begleitung durch die Open Knowledge Foundation war sehr hilfreich. Neben der nicht zu vernachlässigenden administrativen Unterstützung bot die Open Knowledge Foundation Coaching und Seminare an. Wir haben beide Angebote genutzt. Auch wenn diese wenig mit den technischen Aspekten unseres Projekts zu tun hatten, waren sie dennoch sehr informativ. Beim Coaching haben wir

beispielsweise über Projektmanagement gesprochen. Diese Hilfen hätte ich mir alleine nicht geholt, im Nachhinein waren sie jedoch sehr wertvoll.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Mit diesem Projekt sprechen wir Entwickler an, die Kryptografie in ihre Anwendungen integrieren möchten, ohne sich viele Gedanken über die Funktionsweise der Kryptografie oder des OpenPGP-Protokolls machen zu müssen.

Die von uns entwickelte High-Level-API verbirgt viele Details, indem sie dem Entwickler zahlreiche Entscheidungen abnimmt. Das bedeutet einerseits, dass die API weniger ausdrucksstark ist und dementsprechend nicht für alle Anwendungsfälle geeignet ist. In den vielen Fällen, in denen sie doch geeignet ist, bedeutet es andererseits, dass sich Kryptografie mit Sequoia relativ unkompliziert benutzen lässt. Das hat einige Folgen. Erstens sind mehr Entwickler in der Lage, Kryptografie bzw. Sequoia in ihre Anwendungen zu integrieren, da weniger Wissen erforderlich ist. Das ist gut für den Datenschutz. Zweitens weil Entwickler weniger Entscheidungsfreiheit haben, gibt es auch weniger Möglichkeiten, etwas falsch zu machen. Dadurch werden Bugs vermieden. Zuletzt da die Workflows vorgegeben sind, werden unterschiedliche Anwendungen ähnlich funktionieren. Vertraute Workflows sind nicht nur gut für die Usability, sondern auch für die Sicherheit, da Nutzer weniger Fehler machen.

Wir planen, die Bibliothek fertigzustellen, indem wir alle vorgesehenen Workflows implementieren. Nichtsdestotrotz können Entwickler bereits jetzt von dem Projekt profitieren. Die Workflows, die wir während der Förderzeit umgesetzt haben, sind die wichtigsten Workflows und reichen schon für manche Anwendungen.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Wir freuen uns, berichten zu können, dass wir die erledigte Arbeit überwiegend wie geplant umsetzen konnten. Bei der Implementierungsarbeit gibt es immer kleine Experimente und Versuche, die nicht weiter verfolgt werden. Sie sind aber nicht der Rede wert.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer*innen

Interessierte können sich in unserem GitLab-Projekt detailliert über die Projektergebnisse informieren. Die neue Bibliothek ist Teil des Sequoia sq

Projektes und befindet sich im Top-Level-Verzeichnis „lib”.

https://gitlab.com/sequoia-pgp/sequoia-sq

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Aufgrund der Krankheit eines Teammitglieds mussten wir den Arbeitsumfang verkleinern. Dies betraf drei Bereiche.

Erstens, hatten wir geplant, mehr High-Level-Workflows zu implementieren. Aufgrund der mangelnden Ressourcen, haben wir den Umfang reduzieren müssen. Um die Nützlichkeit des Projekts zu gewährleisten, konzentrierten wir uns auf die wichtigsten Workflows. So denken wir, ist eine Integration der Bibliothek in ihrem jetzigen Umfang in viele Projekte trotzdem möglich.

Zweitens hatten wir geplant, eine kleine grafische Demo-Anwendung zu entwickeln, um die Bibliothek zu präsentieren. Wir haben jedoch entschieden, diese Anwendung nicht weiterzuverfolgen, um uns auf die Implementierung der Bibliothek zu konzentrieren.

Schließlich wollten wir mehr Community-Arbeit leisten. Vor dem Projekt hatten wir mit einigen Projekten Kontakt gehabt, die die Bibliothek benutzen wollen. Wir hatten geplant, mindestens eines davon in den letzten Monaten bei der Integration der Bibliothek zu begleiten. Leider war dies nicht möglich, da die Arbeit noch nicht weit genug fortgeschritten war.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Wir haben Sequoia vor acht Jahren ins Leben gerufen. In dieser Zeit wurde Sequoia in einige Softwareprojekte integriert. Durch die Zusammenarbeit und die Beobachtung der Arbeit anderer haben wir gelernt, welche Workflows benötigt werden und was gut bzw. weniger gut funktioniert. Insofern wurde unser Prototype-Fund-Projekt in den letzten Jahren erheblich durch die Community beeinflusst.