Schlussbericht CypherLock—Nötigungsresistente Festplattenverschlüsselung

Frank Rieger, Gordon Gremme und Matthias Menk GbR

2019-09-30

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS18S43 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Kurze Darstellung der Aufgabenstellung und Motivation

Das Ziel des Projektes *Cypherlock* ist die Implementierung eines Softwaretools und der dazugehörigen Backend-Infrastruktur um Geheimnisse (im Konkreten: Passphrases zur Festplattenverschlüsselung) durch zeitbasierende Regeln vor Zugriff zu schützen und damit nötigungsresistente Festplattenverschlüsselung zu ermöglichen. Dieses Ziel ist durch die Tatsache motiviert, dass es weltweit einen Anstieg an forensischen Angriffen und Zwangsmitteln gegen Journalisten bei Grenzübertritt oder auch im Rahmen von Checkpoints in Konfliktregionen gibt. Diese Angriffe gefährden das Wohl der Journalisten als auch ihrer Quellen.

Das Projekt ist in vier Felder unterteilt worden:

1. Implementierung notwendiger Algorithmen, Datenformate und Protokolle für verteilte zeitbasierende Regeln.
2. Implementierung von Backend-Software zur Beantwortung von Entschlüsselungsanfragen und zur Verwaltung serverseitiger geheimer Schlüssel.
3. Implementierung von Client-Software zur Verwaltung von geheimen Schlüsseln und verschlüsselten Nachrichten.
4. Implementierung von Benutzeroberflächen.

Als Milestones wurden definiert:

1. Datenformate und Serialisierungsbibliothek für schützenswerten Daten in geschütztem Arbeitsspeicher ohne Allokation.

2. In-Memory Verwaltung von Geheimnissen, mit Schutz gegen Angriffe durch priviligierte User wie z.B. durch Memory-Scanning.
3. Implementierung der notwendigen kryptographischen Verfahren für den Einsatz in geschütztem Arbeitsspeicher ohne Allokation.
4. Implementierung des verteilten Wiederherstellungs-Protokolls für Geheimnisse.
5. Protokoll-Implementation für ein byzantines Synchronisierungs-Protokoll, sowie dafür notwendige Basisalgorithmen (speziell verifizierbare Merkle-Trees).
6. Implementierung eines nachträglich offenlegbaren Random-Number-Generators für Schlüssel- und Nonce-Erzeugung.
7. Implementation der Client-seitigen Speicherung persistenter Daten unter Berücksichtigung von Informations-Leaks durch Zugriffscharakteristika.
8. Anbindung von Bluetooth-HID Hardware für die Kommunikation zwischen Benutzersoftware (auf Smartphone) und zu schützenden Gerät (Laptop).
9. Implementierung von Benutzersoftware: CLI (primär Linux), iOS und Android.
10. Erstellung relevanter Dokumente zur technischen Beschreibung des Systems.
11. Erstellung von Schulungsmaterial für Benutzer und Dritte.

Die Aufgaben wurden über mehrere Projekt-Teilnehmer verteilt und iterativ bearbeitet und in wöchentlichen Koordinierungs-Treffen besprochen.

Neben dem Ziel ein Produkt zu schaffen, dass die beschriebenen Angriffe erfolgreich abwehren wurde zusätzlich ein Fokus auf die Sicherheit der Implementation gegen fortgeschrittene Angriffe, die sichere und einsatz-relevante Benutzung, sowie die langfristige Pflege der Software gelegt.

Beitrag des Projektes zu den Zielen der Förderinitative "Software-Sprint"

Die primäre Zielgruppe unserer Lösung sind Journalisten, die auf Recherchereisen ihre lokale Daten gegen den Zugriff Dritter schützen müssen.

Bei Benutzung unserer Software können sie die Freigabe von kryptographischen Schlüsseln erfolgreich verhindern in dem sie eine Herausgabe kurzfristig verzögern. Danach sind sie in der Lage zu beweisen, dass kein Zugriff auf die notwendigen Schlüssel mehr möglich ist.

Dazu steht ihnen ein Software-Tool sowie die dafür nötige Infrastruktur zur Verfügung die eine Benutzung auch in Stress-Situationen sicher ermöglicht.

Das Projekt wurde im Rahmen der 5. Runde des Prototype-Funds unter dem Teil-Schwerpunkt “Datensicherheit” gefördert. Cypherlock erhöht die Datensicherheit für besonders gefährdete Berufsgruppen wie Journalisten signifikant. Der gewählte Ansatz ist innovativ, bisher noch nie realisiert worden und hochgradig praxisrelevant.

Ausführliche Darstellung der Ergebnisse

Die Milestones 1-7 wurden vollständig erreicht, Milestone 9 nur teilweise, Milestones 8, 10 und 11 sind nicht erreicht:

Datenformate und Serialisierungsbibliothek für schützenswerten Daten in geschütztem Arbeitsspeicher ohne Allokation

Wir haben eine allgemein nutzbare Implementierung für das serialisieren von Daten geschaffen die ausschliesslich pre-alloziierten Speicher benutzt und in Verbindung mit bestehenden Schutzmechanismen die einfache, effiziente und sichere Konvertierung von Strukturen aus Standard-Typen in Byte-Messages erlaubt. Damit wird die Umsetzung von sicherer Software in der gewählten Programmiersprache *Go* unterstützt und wesentlich vereinfacht.

In-Memory Verwaltung von Geheimnissen, mit Schutz gegen Angriffe durch priviligierte User wie z.B. durch Memory-Scanning.

Als fundamentales Modul unserer Lösung haben wir eine Bibliothek für die Erzeugung und Verwaltung von geheimen Schlüsseln (primär Curve25519 und symmetrische Schlüssel) in geschütztem Speicher geschaffen. Diese Bibliothek erlaubt das Erzeugen, Nutzen und persistente Speichern von Schlüsseln auf Systemen die nicht garantiert gegen den Zugriff durch priviligierte Nutzer gesichert sind.

Weiterhin beinhaltet diese Bibliothek die Erzeugung und Verwaltung von zeitgesteuerten Schlüsseln für die Umsetzung von Richtlinien für zeitlich begrenzten Zugriff auf Schlüsselmaterial. Damit ist eine Implementierung von Systemen mit Zeitrelevanz einfach möglich.

Implementierung der notwendigen kryptographischen Verfahren für den Einsatz in geschütztem Arbeitsspeicher ohne Allokation.

Ein wesentlicher Bestandteil unseres Systems ist eine Bibliothek für die Komposition von geteilten Geheimnissen (shared secrets) auf Basis von Curve25519. Wir haben ein allgemeines Framework für die Definition solcher Komposite geschaffen mit der z.B. auch multi-party Datei-Verschlüsselung oder sichere Messenger

einfacher umsetzbar sind. Im Kontext unseres Projektes erlaubt uns diese Bibliothek die Abbildung aller kryptographischer Envelopes mit öffentlichen Schlüsseln auf eine einzige Konstruktion.

Weiterhin beinhaltet diese Bibliothek eine Implementierung aller von uns genutzten kryptographischen Algorithmen für die Verwendung von geschütztem Speicher.

Implementierung des verteilten Wiederherstellungs-Protokolls für Geheimnisse.

Kernelement unseres Projektes ist das Protokoll für die Wiederherstellung von verteilten Geheimnissen. Hierzu wurden die notwendigen Protokoll-Nachrichten, die persistente Speicherung der States, sowie der Protokollfluss implementiert.

Protokoll-Implementation für ein byzantines Synchronisierungs-Protokoll, sowie dafür notwendige Basisalgorithmen (speziell verifizierbare Merkle-Trees).

Für die byzantine Version unseres Systems, das auch mit der Kontrolle eines Angreifers auf Teile der Backend-Infrastruktur sicher umgehen kann, wurden die Nachrichten-Typen, Speicherung der States sowie der Protokollfluss implementiert. Als weiteren wiederverwendbaren Programmcode wurde in diesem Kontext eine Bibliothek für das Erzeugen und Verifizieren von Merkle-Trees geschaffen die auf Basis von validierten und bewiesenen Konstruktoren basiert.

Implementierung eines nachträglich offenlegbaren Random-Number-Generators für Schlüssel- und Nonce-Erzeugung.

Um das Verhalten und die Funktion unserer Software durch Dritte beweisbar zu machen wurde ein Modul entwickelt, dass die Erzeugung von Zufallszahlen im Nachhinein verifizierbar macht. Damit kann sichergestellt werden, dass alle Daten, die von der Applikation gesendet oder gespeichert wurden, frei von Seitenkanälen sind und das Zufallszahlen ohne Manipulation erzeugt wurden. Der erzeugte Beweis ist für die Applikation nicht veränderbar, vorhersehbar oder unterdrückbar.

Implementation der Client-seitigen Speicherung persistenter Daten unter Berücksichtigung von Informations-Leaks durch Zugriffscharakteristika.

Die Speicherung von schützenswerten Daten für unsere Applikation wurde in einem Modul realisiert, dass den randomisierten Zugriff auf Dateien implementiert

die verschlüsselt gespeichert sind. Die Verschlüsselung benutzt unvorhersehbare „Decoy-Schlüssel“ um das Erpressen von Passwörtern unwirksam zu machen. Ein Angreifer hat keine Möglichkeit zwischen validen und gefälschten Daten zu unterscheiden. Diese Methode ermöglicht dem Benutzer die Wahrung von „Plausible Deniability“ und verhindert forensische Analysen.

Als Bestandteil dieses Modules wurde die sichere Löschung von Daten auf iOS eingebunden, die auf das in Apple-Geräten vorhandene Secure Element zur Schlüsselerzeugung und Verwaltung aufbaut.

Implementierung von Benutzersoftware: CLI (primär Linux), iOS, Android.

Aufgrund technischer (sichere Löschung) und zeitlicher Gründe wurde dieser Milestone nur teilweise erreicht.

Es besteht ein Prototyp der iOS App, die User Interface und Benutzerführung implementiert. Die Einbindung von Bluetooth und Netzwerk-Protokollen fehlt und wird im Nachlauf der Förderphase implementiert.

Neben den Ergebnissen der Implementierungsarbeiten ist insbesondere die Arbeit im Bereich UX/UI entscheidend gewesen. Hierbei hat insbesondere die Arbeit mit einer vom Prototype Fund zur Verfügung gestellte externe Beraterin dazu beigetragen unsere Annahmen zum Benutzer-Verhalten und Ansprüchen zu überprüfen und entscheidend weiterzuentwickeln. Dabei wurde ein vermutlich kritischer Fehler in unseren Annahmen frühzeitig erkannt und konnte somit rechtzeitig in der Entwicklung korrigiert werden.

Weiterhin wurden im Rahmen des Projektes Fähigkeiten zur Frontend-Entwicklung (Flutter) und der Integration von Cross-Platform-Code auf mobilen Endgeräten aufgebaut (go-mobile Integration mit Flutter/Dart).

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Unsere Zielgruppe erhält die Möglichkeit schützenswerte Daten sicher zu verwalten und gegen den Zugriff Dritter in Zwangssituationen zu schützen sowie selber den Umfang einer solchen Situation zu beeinflussen. Damit sind unsere User befähigt Recherchen mit besser einschätzbaren Risiken durchzuführen.

Die erstellten Software-Bibliotheken ermöglichen weiterhin die Implementierung von Produkten mit Hinblick auf sichere Speicher-Nutzung und Schlüsselverwaltung, sowie die einfache Umsetzung von Protokollen die kompositive Verschlüsselung über ECDH beinhalten (Messenging, Forward-Security, Multi-Party Kommuniktion). Zusätzlich sind die erarbeiteten Module zur Serialisierung in sicherem Speicher für den Einsatz in Produkten geeignet, die kryptographische Software auf Geräten mit geringer Sicherheit implementieren. Zudem ist unsere

Implementierung für eine nachträglich verifizierbare Quelle für Zufallszahlen ein allgemein nutzbares Modul für die Beweisbarkeit von kryptographischen Protokollen und kann z.B. dafür genutzt werden Seitenkanäle im Nachrichtenaustausch aufzudecken.

Die Software wird auch in Zukunft von uns weiterentwickelt und fehlende Elemente implementiert. Dieses ist bis Ende 2019 geplant und wird in Teilzeit umgesetzt. Es sollen keine Funktionen hinzugefügt werden, die nicht in der ursprünglichen Leistungsbeschreibung vorgesehen waren.

Weiterhin wird das Endprodukt als App im Apple AppStore angeboten werden. Die notwendige Infrastruktur für den Betrieb des Systems wird durch Erlöse aus dem App-Verkauf erzielt. Weiterhin werden wir Dritte beim Aufbau von dedizierten Backends unterstützen. Eine Weiterentwicklung der Android-Version ist nicht geplant, da sie innerhalb der definierten Sicherheitsparameter im Verlaufe des Projekts als nicht umsetzbar erwiesen hat.

Es gibt weiterhin Versuche die Client-Software auf mobile Peripherie-Geräte/single-board computer (wie z.B. USB-Armory Mk II) zu portieren um eine höhere Sicherheit in der Anwendung zu erreichen. Dieses hätte den weiteren Vorteil, dass Abhängigkeiten zu einer einzigen Platform gemindert werden könnten.

Für unser Team ist das Projekt mit einer starken Lernkurve verbunden gewesen. Neben der Einarbeitung in Frontend-Technologieen (Flutter, UX/UI Prozesse) ist insbesondere die Möglichkeit sich umgehend mit den forensisch relevanten Aspekten von mobiler Hardware auseinanderzusetzen eine Herausforderung sowie auch eine Bereicherung gewesen.

Wir haben es damit erreicht, zwei wesentliche Fähigkeiten für zukünftige Projekte aufzubauen:

1. Benutzerzentriertes Vorgehen in den Bereichen User Experience und insbesondere auch User Interfaces. Insbesondere die Erweiterung unseres Horizonts im Hinblick auf nicht-technische Benutzer ist berreichernd und wird uns für weitere Projekte befähigen.
2. Die Auseinandersetzung mit der Sicherheit mobiler Endgeräte und möglicher sicherer Peripherie hat einen grossen Einfluss auf weitere Pläne unsererseits. Als Konsequenz dieser Lektionen werden wir uns in Zukunft stark auf die Entwicklung von sicheren Basistechnologieen konzentrieren.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Während der initialen UI/UX Phase ist unsere Annahme, dass Benutzer ständig Zugriff auf das Internet haben verworfen worden. Daraus hat sich ergeben, dass wir schutzwürdige Daten sicher von Endgeräten löschen können müssen.

Nach langwieriger Recherche und mehreren Implementationsversuchen mussten wir allerdings feststellen, dass eine forensisch sichere Löschung auf aktuellen Android-Geräten nicht sichergestellt werden kann. Wir haben dazu keine Lösung gefunden und haben daher die Entwicklung des Android Clients eingestellt.

Für iOS Geräte konnte eine akzeptable Lösung gefunden und umgesetzt werden.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Der aktuelle Stand der Entwicklung (Source Code, Projekt-Dokumente) sind über die Cypherlock Webseite[1] beziehbar. Die Website wird auch in Zukunft fortlaufend mit neuen Ergebnissen aktualisiert.

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Zwei Ereignisse haben einen Mehraufwand von insgesamt ca. 160 Stunden erzeugt:

1. Während der initialen UI/UX Phase wurde unsere Annahme, dass Benutzer ständig Zugriff auf das Internet haben, verworfen. Dieses hatte zur Folge, dass wir unsere Protokoll-Spezifikation verändern mussten. Ohne Online-Fähigkeit ist es notwendig geheime Daten vom Endgerät so sicher löschen zu können, dass sie auch durch forensische Analyse nicht wieder hergestellt werden können. Weiterhin macht es erforderlich, dass Zeit-Regeln nicht nur einzeln sondern auch in grösseren Mengen durch einzelne Nachrichten devalidiert werden können.

   Aus Konsequenz daraus mussten wir Teile des Protokolls grundlegend verändern sowie die Backend-Implementation für das Speichern von User-Daten neu programmieren. Weiterhin haben wir signifikaten Aufwand für die Recherche zum sicheren Löschen von Daten auf mobilen Endgeräten vorgenommen und mehrere Methoden dazu implementiert und validiert.

2. Zum letzten Drittel des Projektes wurde ein fundamentaler Angriff gegen die Verschlüsselung des Bluetooth-Protokolls bekannt (die sog. KNOB Attack[2]). Dieser zielt auf die Implementierung des Bluetooth-Protokolles und ist breit wirksam. Der Angriff erlaubt es einem Angreifer ohne physikalischen Zugriff auf die verwendeten Geräte die Verschlüsselung der Nachrichtenübertragung abzufangen und im Klartext zu lesen. Da unser Projekt eine sichere Uebertragung von Daten über Bluetooth zwingend vorraussetzt mussten wir die beschriebene Schwachstelle in der von uns vorgesehenen

[1] https://cypherlock.org
[2] https://knobattack.com

Hardware verifizieren. Daraus ergab sich eine Kooperation mit dem Hersteller um die Auswirkungen auf die Sicherheit unseres Produktes zu mindern. Dieser Prozess ist noch nicht abgeschlossen und ein erfolgreicher Ausgang ist momentan nicht sicher.

Daher haben wir weiterhin alternative Hardware-Platformen recherchiert und validiert. Sollten eine Absicherung der bestehenden Hardware durch den Hersteller nicht erfolgen so besteht für uns die Möglichkeit Anpassungen an alternativen Hardware-Produkten vorzunehmen um die Sicherheit und Nutzbarkeit unserer Software zu gewährleisten.

Grundlegend haben wir in der initialen Projektplanung den Aufwand für die Umsetzung im Kontext von unsicherem Speicher unterschätzt. Zu Anfang sind wir davon ausgegangen, dass wir alle notwendigen kryptographischen Algorithmen als Implementierungen vorfinden würden die einfach für die Verwendung mit unsicherem Speicher nutzbar sind. Im Projektverlauf mussten wir allerdings erheblichen Aufwand in die Implementierung von Speicher-Schutz-Mechanismen stecken und waren weiterhin gezwungen zusätzlichen Programmcode für grundlegende Operationen wie Serialisierung und Speicherung zu schaffen. Als Resultat daraus haben wir ca. 160 Stunden zusätzlich in die Implementierung von Bibliotheken investiert, die den Parametern des Projektes gerecht werden.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Bis auf den bereits oben erwähnten fundamentalen Angriff auf das Bluetooth-Protokoll gab keine Entwicklungen oder Veröffentlichungen Dritter die einen Einluss auf Zielsetzung oder Arbeitsprozesse hatten.