IT-Sicherheit für Entwickler

Ein Drama in drei Akten

Sicherheit ist das vielleicht am meisten vernachlässigte Gebiet der Informationstechnologie. Sie ist umfangreich, divers und komplex. Sie ist politisch. Sie ist undankbar. Sie ist ein Drama. Der Versuch einer Bewältigung in drei Akten.

Version 0.4.2, Juni 2022

http://dck.one, txt@dck.one

Prolog

Die Bühne

Ein kleines Unternehmen, *kein* Konzern, als Kulisse.

Die Szenen

In drei Akten wird das Sicherheitskonzept der Open Source Applikation BlockKeeper erläutert: Der erste Akt beleuchtet zunächst allgemeine Grundlagen und Kontext. Im zweiten und dritten Akt folgen theoretische Konzeption sowie die praktische Umsetzung.

Der Protagonist

Ein erfahrener Entwickler, kein Sicherheitsexperte, im inneren Dialog.

Der Autor

Besitzt gut 25 Jahre Berufserfahrung in den Bereichen IT und Maschinenbau als Architekt, Entwickler, UNIX System-/Netzwerkadministrator und Unternehmer. Er war 2017 einer der leitenden BlockKeeper Programmierer.

Als Mitgründer von Micro-Colocation.com treibt er seit 2021 grünes Edge-Computing für Kleinrechner wie Raspberry Pi, Odroid und Jetson Nano voran. CC lizenzierte Texte wie dieses Drama entstehen innerhalb dieser Unternehmung, werden also durch Buchungen und Teilen von Micro-Colocation.com in sozialen Medien gefördert. Vielen Dank.

Inhaltsverzeichnis

Prolog....2
Erster Akt....4
Erste Szene – Die Dilemmata....4
Zweite Szene – Der Entwickler....6
Dritte Szene – Die Verbündeten....7
Vierte Szene – Die Konzeption....12
Zweiter Akt....15
Erste Szene – Die Applikation....15
Zweite Szene – Die Identifikation....19
Dritte Szene – Der Zugriff....22
Vierte Szene – Die Verschlüsselung....24
Dritter Akt....27
Letzte Szene: Die Umsetzung....27
Epilog....30
Bitte / Danke....30
Requisiten....31

Erster Akt

Erste Szene – Die Dilemmata

In der Informationstechnologie existieren mindestens drei grundlegende Dilemmata:

Das Ökonomie-Dilemma

Um den in kapitalistischen Unternehmungen angestrebten Gewinn zu erzielen, ist die Minimierung des Produktionsaufwand ein entscheidender Faktor. Nicht nur in der Informationstechnologie korrelieren die Kosten insbesondere mit der Produktionszeit. Deren maximale Größe ist deshalb im Normalfall hartkodiert und führt zu einem zentralen Problem: Wenn sie überschritten wird, die Entwicklungsgeschwindigkeit also zu gering ist, ist das Produkt nicht überlebensfähig. Entweder die Konkurrenz bedient den Markt zuvor oder die (Personal-) Kosten sprengen den akzeptablen Rahmen. Wird hingegen die definierte Produktionszeit eingehalten, steht nie genügend Zeit zur Umsetzung eines *perfekten* Produkts zur Verfügung.

Das Bedarfs-Dilemma

Der Bedarf nach innovativen, funktionalen, intuitiven und glitzernden Produkten ist hoch. Der Bedarf nach sicheren Produkten ist vergleichsweise gering. Sicherheit wird erst wichtig, wenn es zu spät ist.

Das Sicherheits-Dilemma

Die Designphilosophie des auf Sicherheit und Datenschutz fokussierten Email-Dienstes ProtonMail:

Our basic assumption is that all servers can be compromised, and that sooner or later, ProtonMail will also be compromised. Incidents like the

Yahoo breach are not isolated, and will gradually become the norm over the next decade.

ProtonMail Blog: Improved Authentication for Email Encryption and Security

Insbesondere ein Aspekt macht das Thema Sicherheit schwer beherrschbar: Schwachstellen haben häufig nicht lokale (Feature), sondern globale (Produkt) Auswirkungen. Bereits kleinste Fehler bei Konzeption oder Implementierung können die Sicherheit des gesamten Systems gefährden. Hoher Aufwand resultiert nicht zwingend in einem sicheren Produkt. Das Web, als *die* informationstechnologische Plattform schlechthin, hat zusätzlich mit dem Web-Dilemma zu kämpfen:

Cryptography is a systems problem, and the web is not a secure platform for application delivery. The web is a way to easily run untrusted code fetched from remote servers on-the-fly. Building security software inside of web browsers only makes the problem harder.

Tony Arcieri: What's wrong with in-browser cryptography?

Sicherheit ist komplex.

Die Konsequenz

Sicherheit ist nur einer von vielen Produktaspekten und verfügt aufgrund des Bedarfs-Dilemmas häufig über eine geringe Priorität.

Aufgrund der geringen Priorität und der durch das Ökonomie-Dilemma begrenzten Zeitressource muss der Umfang von Sicherheitsmechanismen sinnvoll gewählt werden: Zu viele Maßnahmen resultieren in einer Teilumsetzung und führen genau wie zu wenige Maßnahmen zu einem unbrauchbaren Produkt.

Ausgangspunkt einer sinnvollen Maßnahmenauswahl ist eine Risikobewertung (Risk Rating), die die Leitfrage beantwortet: Welche Sicherheitsvorkehrungen müssen getroffen werden, damit im Fehlerfall der (wirtschaftliche) Schaden für Anwender und Hersteller akzeptabel bleibt?

Auch nach einer fundierten Risikobewertung ist das Finden eines einheitlichen team-, unternehmens- oder gar internetweiten Konzeptkonsens aufgrund des Sicherheits-Dilemmas schwierig. Kritik an der Maßnahmenauswahl sollte deshalb als prozessimmanent verstanden, akzeptiert und für stetige Verbesserungen genutzt werden.

Ein sicherheitstechnisch optimiertes IT-Produkt bedingt fundierten Entwurf, vollständige Umsetzung und kontinuierliche Verbesserung eines *inhaltlich und wirtschaftlich passenden* Konzepts.

Zweite Szene – Der Entwickler

Protagonist des Dramas ist ein erfahrener Entwickler. Kein IT-Sicherheitsexperte. Kein Kryptograph. Dieser Unterschied ist wesentlich: Entwickler sind es gewohnt, sich in abstrakte Problemstellungen einzuarbeiten, das Thema IT-Sicherheit nimmt aber aus zwei Gründen eine Sonderstellung ein:

1. Das Sicherheits-Dilemma existiert: Systemsicherheit ist nicht nur umfangreich, sondern auch komplex und das Ökonomie-Dilemma gewährt nur wenig Einarbeitungszeit. Es bedarf einer Vielzahl grundlegender Kenntnisse, um sich dem Thema brauchbar zu nähern.
2. Sicherheit ist schwer verifizierbar: Die Vorhersage, ob eine gewählte Implementierung jedem zukünftigen Angriff standhalten wird, ist nicht möglich.

While the developer will say "but the modified data will come back as garbage after decryption", a good security engineer will find the probability that the garbage causes adverse behaviour in the software, and then he will turn that analysis into a real attack.

Stack Overflow: Forenbeitrag von Benutzer TheGreatContini

Kleinere Unternehmen verfügen normalerweise nicht über Sicherheitsexperten in den eigenen Reihen. Deshalb das Sicherheits-

Dilemma zu ignorieren, ist keine Option. Stattdessen fällt die Bewältigung in den Aufgabenbereich der hauseigenen Entwickler. Und die brauchen: Verbündete.

Dritte Szene – Die Verbündeten

Literatur und Standards

Fachpublikationen und Standards sind die engsten Verbündeten bei der Einarbeitung in ein Thema. Bei ihrem Studium besteht die Herausforderung in der effektiven Filterung der Informationen: Sowohl bei der Auswahl als auch beim Querlesen der Literatur muss sich auf den Punkt *Implementierung* konzentriert werden, denn ein tiefes Studium der (mathematischen Kryptographie-) Theorie ist aufgrund des Ökonomie-Dilemmas nicht möglich. Es ist aber auch nicht erforderlich, da der Software-Ingenieur keine Grundlagenforschung betreibt, sondern die Anwendung beherrschen muss.

Standardisierte Prozesse helfen ihm dabei und sind insbesondere in der IT-Sicherheit von großer Bedeutung. Offizielle Richtlinien wie die von NIST (SP 800, Computer security) und RFC publizierten sind zwar als Informationsquelle für Forscher insgesamt von größerer Bedeutung, aber trotzdem auch für Anwender zur Beantwortung von Detailfragen interessant. Denn nicht jede Code Bibliothek bringt umfangreiche Dokumentation mit sich, häufig ist unklar, was einzelne Parameter bedeuten. Ein Blick in die zugehörige NIST SP 800 Algorithmusbeschreibung kann in einem solchen Fall hilfreich sein.

Bundesbehörden

Aufgrund der großen Bedeutung der Informationstechnologie besitzen viele Staaten darauf fokussierte Behörden. In deren Aufgabenbereich fallen u.a. die Spezifizierung von kryptografischen Normen, die Dokumentation und Warnung vor Sicherheitslücken sowie die sicherheitstechnische Überprüfung und Lizenzierung von IT-Systemen.

Bekannt ist insbesondere das amerikanische National Institute of Standards and Technology (NIST), in Deutschland das Bundesamt für Sicherheit in der Inforamtionstechnik (BSI). Dessen Technische Richtlinien sowie das umfangreiche IT Grundschutz Kompendium bieten nicht nur eine allgemeine Einführung, sondern auch zielgerichtete Empfehlungen zum Thema IT-Sicherheit. Für Entwickler ist es mehr als hilfreich, beispielsweise konkrete Angaben zu kryptografischen Schlüssellängen und einen Katalog mit sinnvollen Sicherheitsmaßnahmen für Kleinunternehmen zu erhalten.

In Expertenkeisen wie dem Chaos Computer Club wird allerdings in vielen Vorträgen rege thematisiert, ob bei staatlichen Institutionen die Absicherung von (kritischen) Systemen oder die Belange von Geheimdiensten, Strafverfolgungsbehörden oder gar militärischen Hackback Interessen im Vordergrund stehen.

Diese offene Diskussion weder ignorierend noch überbewertend, spielen Behördenrichtlinien im Folgenden keine weitere Rolle. Gründe sind eine dem Textumfang geschuldete Quellenfokussierung sowie der diesem Drama zugrundeliegende Open Content Gedanke, der sich besser über die Nutzung internationaler, öffentlicher Community-Inhalte transportiert.

Praxisleitfäden

Dank der inzwischen weit verbreiteten Open Source Kultur finden sich zu jedem IT-Thema umfangreiche Dokumentationen, Konzepte und Best Practices wie beispielsweise Lessons learned and misconceptions regarding encryption and cryptology. Bereits in einer frühen Phase gilt es, die zur eigenen Produktentwicklung passenden Methoden zu identifizieren. Wenn das eigene Produkt einen extremen Spagat zwischen Bedienbarkeit und Sicherheit (Bedarfs-Dilemma) absolvieren muss, kann beispielsweise ein Blick auf die Vorgehensweise des Signal Messenger Teams lohnen, da es mit den gleichen Anforderungen zu kämpfen hat.

I am regularly impressed with the thought and care put into both the security and the usability of this app. It's my first choice for an encrypted conversation.

Signal Hompage: Bruce Schneier über den Messenger Signal

Auch der Passwort-Manager LastPass und viele andere Projekte geben Einblick in ihre Technologien. Eine gründliche Recherche und sorgfältige Auswahl in diesem Umfeld, kann den Konzeptentwurf extrem verkürzen und dadurch mehr Zeit für die Implementierung schaffen.

Und der gefühlte König der Prinzipien darf natürlich nicht unerwähnt bleiben. Insbesondere im Umfeld von Sicherheit und Kryptographie kann nur sein Regiment den Untergang des Reichs verhindern:

Keep it simple, stupid: The KISS principle states that most systems work best if they are kept simple rather than made complicated; therefore simplicity should be a key goal in design and unnecessary complexity should be avoided.

Wikipedia: Das KISS Prinzip

Code Bibliotheken

Müssen verwendet werden! Das Don't Roll Your Own Crypto Mantra steht für jeden Entwickler an vorderster Stelle: Software-Entwickler sind *Anwender* von Sicherheits- und Kryptographielösungen.

I learned how easy it is to fall into a false sense of security when devising an encryption algorithm. Most people don't realize how fiendishly difficult it is to devise an encryption algorithm that can withstand a prolonged and determined attack by a resourceful opponent.

Phil Zimmermann: An Introduction to Cryptography

Es gilt die Faustregel: Minimiere die Fertigungstiefe der eigenen Sicherheitsfunktionen. D.h. in der Praxis: Von Experten geprüfte, im Idealfall standardisierte Code Bibliotheken bei der Implementierung nutzen und nicht entwickeln.

Security is only as strong as the weakest link, and the mathematics of cryptography is almost never the weakest link. The fundamentals of cryptography are important, but far more important is how those fundamentals are implemented and used.

Bruce Schneier: Practical Cryptography Preface

Experten

Auch für erfahrene Software-Entwickler ist es anfangs nicht einfach, den richtigen Weg im Dschungel der IT-Sicherheit zu finden. Schon die Auswahl der Literatur gestaltet sich schwierig, da die eigenen Informationsfilter noch nicht kalibriert sind. Wie in jeder Szene gibt es aber auch in der IT-Sicherheitsbranche viele Experten, die aufgrund ihrer geleisteten Arbeit einen besonderen Vertrauensstatus genießen. Ihre Publikationen und Vorträge können ein Startpunkt für weitere Recherchen und das Einsteigen in die Materie sein. Z.B. stößt man häufig auf Bruce Schneier, Matthew Green, Phil Zimmermann und D. J. Bernstein. Auch national finden sich immer Größen, z.B. hat sich nicht nur im deutschsprachigen Raum Prof. Rüdiger Weis einen Namen gemacht und wird als Kryptograph nicht müde, in Chaos Computer Club Vorträgen die Wichtigkeit der korrekten Nutzung von Sicherheitsfunktionen zu betonen. Und alle genannten sind nur die prominente Spitze des Eisbergs, viele weitere Fachleute finden sich in Foren und leisten dort fantastische Supportarbeit.

Yet it may be roundly asserted that human ingenuity cannot concoct a cipher which human ingenuity cannot resolve.

Edgar Allan Poe

Die generelle Weisheit der Meinung Einzelner nicht blind zu vertrauen, gilt auch hier und wird durch das Sicherheits-Dilemma forciert. Unklare Konzeptpunkte sollten deshalb nicht nur mit dem eigenen Team (und zugehörigen Einzelexperten), sondern im Zweifelsfall auch mit einer größeren Anzahl Sachkundiger diskutiert werden…

Foren

Zum Thema Sicherheit finden sich im Internet zahlreiche Informationsquellen und Foren (Communities). Im Folgenden werden exemplarisch zwei von ihnen genannt, die bei der im zweiten und dritten Akt beschriebenen Umsetzung eine zentrale Rolle spielen.

OWASP: Die OWASP Foundation ist eine im IT-Sicherheitsumfeld etablierte Community, die Handlungsempfehlungen veröffentlicht. Publikationen wie z.B. die Top 10 Web Application Security Risks sind insbesondere für die anfängliche Risikobewertung hilfreich.

OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security.

Die OWASP Foundation Mission

Stack Exchange: Sowohl in der Konzeptphase als auch bei der späteren Implementierung stellen sich regelmäßig konkrete Fragen, die sich auch durch tiefe Recherchen nicht beantworten lassen. *Das* Q&A Community-Netzwerk für solche Fälle ist jedem Entwickler bekannt: Stack Exchange. Hilfe im Bereich der IT-Sicherheit findet sich insbesondere in den folgenden Foren:

Information Security Stack Exchange is a question and answer site for information security professionals.

Die Information Security Stack Exchange Mission

Cryptography Stack Exchange is a question and answer site for software developers, mathematicians and others interested in cryptography.

Die Cryptography Stack Exchange Mission

In beiden Foren erhält der Fragende meist innerhalb weniger Stunden professionelle Antwort und kann bei Unklarheiten weitere Details über

den Kommentardialog erfahren. Voraussetzung ist allerdings die Beachtung des Stack Exchange Prinzips:

Focus on questions about an actual problem you have faced. […] Avoid questions that are primarily opinion-based, or that are likely to generate discussion rather than answers.

Stack Exchange Credo

Wie soll mein Sicherheitskonzept aussehen? wird deshalb zu weniger oder keiner Resonanz führen. Stattdessen gilt es, konkrete Fragen wie *Was bedeutet Parameter x bei Verschlüsselungsalgorithmus y im Kontext von z?* zu stellen. Grundlage hierfür ist ein mit Hilfe der anderen Verbündeten entwickeltes Konzept.

Vierte Szene – Die Konzeption

Obwohl zu Beginn eines Projekts viele (Produkt-) Details nicht bekannt sind, ist die Erstellung eines ersten Grobkonzepts als Ausgangbasis unerlässlich.

Der Entwurf

Teil dieser Skizzen ist eine Abschätzung des angestrebten Sicherheitsstandards, der sich aus einer ersten Gewichtung der Dilemmata ergibt. In Kombination mit anderen Produktparametern gilt es anschließend, die IT immanenten Technologiefragen zumindest rudimentär zu beantworten. Z.B. ist die Auswahl der Plattform (Web-, Mobile-, Desktop-Anwendung) von entscheidender Bedeutung. Sie gibt den Rahmen der Programmiersprachen, Code Frameworks und Bibliotheken vor.

Die Risikobewertung

Entlang des Entwurfs gilt es, die einzelnen Aspekte der Dilemmata in Hinsicht auf das geplante Produkt detaillierter zu klassifizieren. Einschätzungen dieser Art sind ein winziger Bestandteil des Risk- bzw.

Information Security Managements. Dessen vollständige Lehrbuchumsetzung und ggf. Zertifizierung nimmt gigantische Ausmaße an und kann von kleinen Unternehmen nur schwer geleistet werden. Der Entwickler sollte sich stattdessen zunächst auf eine extrem vereinfachte Risikobewertung (Risk Rating) konzentrieren: 1.) In welchem Maß ist das neue Produkt von einem Dilemma oder Aspekt betroffen? 2.) Welche Auswirkungen hat es auf den Entwicklungs- und Geschäftsprozess? Ist das Produkt technisch und wirtschaftlich umsetzbar? 3.) Wie sehen die Prioritäten der resultierenden Maßnahmen aus?

Allerdings bleibt die Beantwortung auch dieser wenigen Leitfragen schwierig. Hilfestellung findet sich z.B. bei der OWASP Foundation, die die Risk Rating Methodology publiziert:

Discovering vulnerabilities is important, but being able to estimate the associated risk to the business is just as important. [...] By following the approach here, it is possible to estimate the severity of all of these risks to the business and make an informed decision about what to do about those risks. Having a system in place for rating risks will save time and eliminate arguing about priorities. This system will help to ensure that the business doesn't get distracted by minor risks while ignoring more serious risks that are less well understood. Ideally there would be a universal risk rating system that would accurately estimate all risks for all organizations. But a vulnerability that is critical to one organization may not be very important to another. So a basic framework is presented here that should be customized for the particular organization.

OWASP: Risk Rating Methodology

Konkret veröffentlich OWASP regelmäßig die Top 10 Web Application Security Risks, die auch als praktischer Leitfaden bei der Risikobewertung dienen können.

Die Planung

Die in der Risikobewertung identifizierten Anforderungen werden im Zuge der Planung mit verfügbaren Technologien verknüpft und zum Schluss in konkrete Implementierungsaufgaben überführt.

To produce a secure web application, you must define what secure means for that application.

OWASP: Top 10 2017 Web Application Security Risks

Die Verbündeten helfen dabei, die richtigen Entscheidungen zu treffen: Während anfangs die Einarbeitung mittels Literatur und Handlungsempfehlungen dominiert, nimmt im Laufe der Zeit die Bedeutung von konkreten Code Beispielen sowie die Hilfe von Q&A Communities zu. Insbesondere bei agiler Softwareentwicklung vermischen sich die Arbeitsschritte Konzeptionierung, Prototyping und Implementierung permanent, da die Auswahl geeigneter Sicherheitslösungen vieler Tests bedarf. Auch die Frage der Verfügbarkeit geeigneter Code Bibliotheken ist ein entscheidender Faktor. Er nimmt bereits in einer frühen Konzeptphase Einfluss und ist von der Produktplattform sowie der verwendeten Programmiersprache geprägt. Z.B. verfügen aktuelle Browser über eine standardisierte Kryptographie-Schnittstelle, die das Web-Dilemma zumindest etwas entschärft. Sie ermöglicht die Überlegung, ob ein ursprünglich als Mobile-App angedachtes Produkt vielleicht auch als Web-Service realisiert werden kann.

Vorhang.

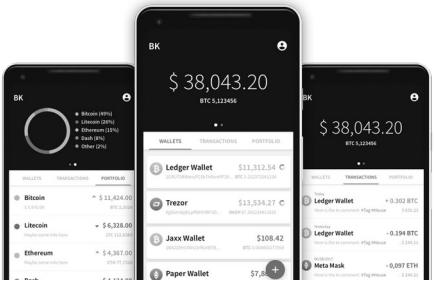
Zweiter Akt

Der vorausgegangenen Exposition folgt nun im zweiten Akt die Schilderung des für BlockKeeper entworfenen Sicherheitskonzepts. Besonderes Augenmerk gilt hierbei der Rolle der vorgestellten Verbündeten und den Kompromissen, die aufgrund der definierten Dilemmata bei der Planung gefunden werden mussten. Um den Umfang der Aufführung nicht zu strapazieren, konzentrieren sich die folgenden Szenen auf die Hauptthemen Authentifikation, Autorisierung und Verschlüsselung. Bei den Verbündeten liegt der Fokus auf den von der OWASP Foundation publizierten Top 10 Web Application Security Risks sowie den Lessons learned and misconceptions regarding encryption and cryptology des Information Security Stack Exchange Blogs.

Anmerkungen: Der erste kurze Rohentwurf dieser Aufführung entstand Ende 2017. Der Autor war damals einer der leitenden BlockKeeper Entwickler. BlockKeeper selbst wird seit 2019 nicht mehr weiterentwickelt. Seine Ansätze und Technologien waren aber Ideengeber für spätere Projekte.

Erste Szene – Die Applikation

Um getroffene Konzeptentscheidungen einordnen zu können, ist es zunächst wichtig, Eigenschaften, Ziele und Umgebung einer Anwendung genau zu definieren.



Track your crypto-portfolio, secure and anonymous.
The world is going to run on Blockchains and the number of digital coins and assets are exploding. BlockKeeper is your book keeping App for Blockchain assets that helps you to track all your digital wealth, income and expenses.

BlockKeeper Homepage 2017

Die Eigenschaften

BlockKeeper stellt seinen Benutzern Informationen über ihre Kryptowährungsvermögen zu Verfügung, verwaltet also Finanzdaten, die grundsätzlich als sensitiv einzustufen sind. Allerdings besitzt es nur lesenden Zugriff darauf. Konkret verwendet und speichert BlockKeeper ausschließlich öffentliche Blockchain-Adressen (Public Keys), um Salden (Balances) und Transaktionswerte von Kryptowährungen (Coins) darzustellen.

Einzelne öffentliche Adressen sind für sich genommen kein Geheimnis: Ihre Salden sind jederzeit über sogenannte Block-Explorer wie z.B. Blockchain.info einsehbar. Um die Coins ausgeben zu können, ist immer der zugehörige private Schlüssel (Private Key) erforderlich. Da BlockKeeper diesen Schlüssel nicht besitzt, kann selbst das größte Datenleck nicht zu einem Vermögensverlust beim Benutzer führen.

Es ist allerdings nicht öffentlich einsehbar, welche Adresse welchem Benutzer gehört: Blockchains kennen nur Adressen, Transaktionen und zugehörige Salden, sie speichern nicht, dass der Public Key x der Person y zuzuordnen ist. Gängige Kryptowährungen besitzen deshalb einen gewissen Grad an Pseudoanonymität.

Eine Applikation, die eine Vermögensübersicht bereitstellt, muss zwingend innerhalb ihrer Prozesse die Beziehung zwischen einem Benutzer und seinen öffentlichen Adressen herstellen. Zwar speichert BlockKeeper keinerlei persönliche Daten wie reale Namen oder Email-Adressen, so dass dieser Part der Pseudoanonymität erhalten bleibt. Trotzdem sind die gespeicherten Beziehungsinformationen (im Folgenden

als *Coin-Adressen* bezeichnet) sensitiv und müssen bestmöglich vor den Blicken Dritter geschützt werden.

Die Plattform

Das Web ist ein hervorragender Ort, um Informationen darzustellen. Einfache Bedienbarkeit, hoher Bekanntheitsgrad und weitestgehende Betriebssystemunabhängigkeit sind gute Argumente, wenn sich bei einem Produkt die Plattformfrage stellt. Die Vorzüge des Webs helfen bei der Bewältigung des Bedarfs-Dilemmas, gleichzeitig verschärft die Browser-Schnittstelle aber auch das Web-Dilemma:

> *Whether you are new to web application security or are already very familiar with these risks, the task of producing a secure web application or fixing an existing one can be difficult. If you have to manage a large application portfolio, this task can be daunting.*
>
> OWASP: Top 10 2017 Web Application Security Risks

Trotzdem entschloss sich das BlockKeeper Team in der Entwurfsphase 2017 für das Web als Plattform, da es *für dieses Produkt* die beste Ausbalancierung zwischen Anforderungen und Dilemmata bot. Jahre zuvor wäre diese Entscheidung anders ausgefallen: Browser besaßen erst seit Kurzem die WebCrypto-API, die einen standardisierten Zugriff auf kryptographische Javascript Funktionen ermöglicht. Obwohl diese Schnittstelle von vielen Experten kritisiert wird und schon gar nicht das grundsätzliche Web-Dilemma löst, war dieser Verbündete trotzdem ausschlaggebend zur Erreichung des in der Entwurfsphase definierten Sicherheitsstandards. Die Konsequenz der Nutzung ist aber auch, dass BlockKeeper ausschließlich Browser neuerer Generation unterstützt und dadurch das Bedarfs-Dilemma verschärft.

Die User Experience

Ein wichtiger Faktor und Teil des Bedarfs-Dilemmas ist die User-Experience *(UX)* einer Applikation. Viele Web-Anwendungen sind in dieser Hinsicht suboptimal, auch weil häufige Ladezeiten die Nutzung

stören. Mit Hilfe moderner Browser-Funktionen und Javascript Frameworks ist es aber möglich, eine ähnlich flüssige Navigation zu erzielen, wie sie der Anwender von Desktop- und Mobile-Applikationen kennt. Eine in diesem Zusammenhang intensiv von BlockKeeper genutzte Funktion ist der Local-Storage des Browsers: Seine Verwendung als Primärspeicher für alle Benutzerdaten reduziert die Kommunikation mit dem serverseitigen Backend und damit die Ladezeiten erheblich, hat aber Auswirkungen auf das Sicherheits-Dilemma.

Die Dritten

In Sicherheits- und Datenschutzdefinitionen ist schnell von sogenannten Dritten die Rede. In der Konzeptionsphase stellt sich die zentrale Frage:

Wer sind die Dritten (Angreifer, Feinde, ...), vor denen die Anwenderdaten geschützt werden müssen?

Im Falle von BlockKeeper lautet sie konkreter: Welche Person darf auf die sensiblen Coin-Adressen zugreifen? Die Antwort ist: Nur der Anwender. Schwieriger wird die Beantwortung derselben Frage, wenn sie aus dem Blickwinkel der Applikation gestellt wird:

Welche Systemkomponenten dürfen auf die Anwenderdaten zugreifen?

Bei gefühlten 90 Prozent der derzeit auf dem IT-Markt verfügbaren Applikationen (unabhängig von Branche, Funktion oder anderen Kriterien) lautet die Antwort: Alle. Das bedeutet, dass Daten nie nur auf dem Gerät des Anwenders verarbeitet, sondern immer auch zum Backend des Betreibers übertragen und dort gespeichert werden. Daraus resultiert, dass nicht nur der Benutzer, sondern auch der Dienstbetreiber Zugriff auf die Daten besitzt. Bei vielen Services ist diese Maßnahme nachvollziehbar, sinnvoll oder technisch zwingend notwendig. Bei einem Produkt wie BlockKeeper hingegen nicht. Das BlockKeeper Team war deshalb der Auffassung, dass weder böswillige Angreifer, noch das Team als Betreiber Zugriff auf die Anwenderdaten haben sollten. Entsprechend wurde das Sicherheitskonzept gestaltet.

Zweite Szene – Die Identifikation

Der Sicherheitsgrad von Authentifikation und Autorisierung ist insbesondere von einem Faktor abhängig: Den sogenannten Credentials (Benutzername/Passwort). Wodurch sich der Entwickler sofort im Bedarfs-Dilemma befindet:

People are notoriously poor at achieving sufficient entropy to produce satisfactory passwords.

Wikpedia: Human-generated passwords

Der User Identifier

BlockKeeper rückt den Anmeldekomfort zugunsten der Sicherheit in den Hintergrund: Der Anwender kann nicht wie gewohnt bei der Registrierung einen Benutzernamen wählen, sondern bekommt vom System einen sogenannten User Identifier zugewiesen. Der Nachteil ist offensichtlich: Für Menschen ist es unmöglich, sich an eine Zeichenfolge wie 74ad7a9a-3c0b-4dad-8aa9-c80437506605 zu erinnern. Im Falle von BlockKeeper verliert dieser Faktor aber durch mehrere Aspekte an Gewicht:

- Der Benutzer wird direkt bei der Registrierung eingeloggt und muss sich im Normalfall (an diesem Gerät) kein weiteres Mal anmelden: Die persistente Speicherung der notwendigen Daten übersteht im Local-Storage auch Neustarts von Browser und Betriebssystem.
- Die Kryptowährungsgemeinde ist es gewohnt, mit kryptographischen Schlüsseln in Berührung zu kommen. Die sichere Aufbewahrung auch komplexer Credentials ist ihr deshalb mehr vertraut als fremd.
- Dem Benutzer wird die Akzeptanz der Credentials durch den Support von Passwort-Managern, einer passenden Gestaltung der

Registrierungsmaske, einer einfachen Backup-Funktion sowie zugehöriger FAQ Beiträge erleichtert.

Dem Komfortverlust stehen eine Reihe von Vorteilen gegenüber:

- Wenn der User Identifier mit Hilfe eines kryptographisch sicheren Zufallszahlengenerators (CSPRNG) erzeugt wird, enthält er ausreichend Entropie (Make sure you seed random number generators with enough entropy), um als sicheres Authentifikations-Token zu dienen. Zusätzliche Key-Stretching Verfahren sind nicht erforderlich.
- In diesem Fall bedarf es auch keiner Benutzername/Passwort Kombination, der CSPRNG-generierte User Identifier vereint sie in *einem* Token: Er sorgt aufgrund seiner Zufälligkeit einerseits für die eindeutige Identifizierung des Anwenders (Benutzername), gleichzeitig aber auch für eine sichere Zugriffskontrolle (Passwort). Als dementsprechend sensitiv ist er zu bewerten und muss unbedingt vor den Augen Dritter geschützt werden.
- Da viele Anwender sich einen frei erfundenen Benutzernamen nicht merken können, tendieren sie zur Nutzung ihrer Email-Adresse, was aber dem grundlegenden BlockKeeper Konzept widerspricht: Es werden keine Personendaten gespeichert. Durch das Setzen eines User Identifiers wird diese Bredouille von vorneherein vermieden.

Der Crypto Key

Die von BlockKeeper verwalteten Informationen sind per Definition nur für den Anwender zugänglich, müssen aber trotzdem zuverlässig (d.h. nicht nur auf dem Benutzergerät) gespeichert werden. BlockKeeper löst dieses Bedarfs-Dilemma durch die Kombination von Server-Side Speicherung mit Client-Side Verschlüsselung:

Client-side encryption is the cryptographic technique of encrypting data on the sender's side, before it is transmitted to a server [...] Client-side

encryption features an encryption key that is not available to the service provider, making it difficult or impossible for service providers to decrypt hosted data.

Wikipedia: Client-side encryption

Der für die Chiffrierung benötigte Schlüssel besitzt hinsichtlich seiner Generierung die gleichen Entropieanforderungen wie der User Identifier. Da Letzterer dem Backend aber zwingend bekannt sein muss, kann er nicht zur Verschlüsselung der Daten verwendet werden. Es gilt die Grundregel:

Don't use the same key for both encryption and authentication.

Stack Overflow: Blogartikel von Benutzer roryalsop

Zusätzlich zum User Identifier benötigt der Anwender also ein zweites, in diesem Fall nur ihm bekanntes Credential. Bei BlockKeeper trägt es intern den Namen Crypto Key und wird bei der Registrierung auf dieselbe Weise wie der User Identifier erzeugt. Das Bedarfs-Dilemma wird dadurch nicht verschärft: Ob der Local-Storage, das Credential-Backup oder der Passwort-Manager ein oder zwei komplizierte Schlüssel enthalten ist irrelevant.

Die Speicherung

Die Credentials werden innerhalb der Prozesse permanent zur Authentifikation, Autorisierung, Ver- und Entschlüsselung benötigt. Sie müssen also nach dem Login im Browser vorgehalten und im Falle des User Identifiers regelmäßig zum Backend gesendet werden. Das Schließen von Browser-Tabs bzw. des ganzen Browsers ist allerdings ein häufiger Vorgang, der ohne weitere Maßnahmen jedesmal eine erneute Credentials-Abfrage zur Folge hätte. Um dieses Bedarfs-Dilemma zu lösen und eine einfache Übertragung des User Identifiers zu gewährleisten, wurde für BlockKeeper folgende Grundsatzentscheidung getroffen:

Passwort-Manager (betrifft nur den User Identifier) und Local-Storage des Browsers sowie der HTTPS Header sind *ausreichend unzugängliche*

Bereiche, um darin unverschlüsselte Informationen zu speichern bzw. zu übertragen. Das bedeutet, dass BlockKeeper alle Daten (Credentials, Coin-Adressen etc.) persistent und unverschlüsselt im Local-Storage ablegen sowie in HTTPS Header-Feldern an das Backend übermitteln darf.

Der Kompromiss

Diese Entscheidung ist eine typische Kompromissfindung, wie sie aufgrund der Dilemmata immer wieder notwendig wird, mit Blick auf die Applikationssicherheit ist diese Art der Speicherung in keinster Weise perfekt. Insbesondere wird dabei vorausgesetzt, dass es einem Angreifer nicht möglich ist, sich physikalischen Zugang zum Gerät des Anwenders zu verschaffen. Denn er könnte dann über die Web-Developer-Tools des Browsers die sensitiven Werte des Local-Storage sowie der HTTPS Header auslesen. Die Kompromissentscheidung bewirkt also eine für BlockKeeper geeignete Absicherung, die sich mit den definierten Sicherheitsanforderungen deckt und dabei den KISS Verbündeten nicht ignoriert. Bei einer anderen Applikation mit anderen Sicherheitsanforderungen, wird dieser Kompromiss vollkommen unzureichend sein:

To produce a secure web application, you must define what secure means for that application.

OWASP: Top 10 2017 Web Application Security Risks

Sicherheit ist ein Dilemma. Meistens sogar ein Drama.

Dritte Szene – Der Zugriff

Ein Blick auf die harte Realität, in der unbegründeten Hoffnung, es vielleicht etwas besser zu machen:

Broken Authentications: Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other

implementation flaws to assume other users' identities temporarily or permanently.

Broken Access Control: Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

OWASP: Top 10 2017 Web Application Security Risks

BlockKeepers Zugriffsrechte sind denkbar einfach: Jeder Anwender darf auf seine, aber keine sonstigen Ressourcen zugreifen. Zur Adressierung der Ressourcen werden UUIDs verwendet, deren Generierung bei BlockKeeper ausschließlich durch Einsatz einer CSPRNG-fähigen Code Bibliothek erfolgt. Durch diese Maßnahme ergibt sich der pragmatische Ansatz, den User Identifier als UUID realisieren zu können und ihn eine Doppelrolle übernehmen zu lassen:

1. Er fungiert als normale ID, die unter Verwendung von UUID-basierten Standardverfahren zur Adressierung von Anwender-Ressourcen genutzt wird.
2. Er repräsentiert den Eintrittspunkt zu den Ressourcen des zugehörigen Benutzers. Das bedeutet der Anwender darf auf alle Ressourcen zugreifen, die der User Identifier Ressource hierarchisch untergeordnet sind.

Authentifikation und Autorisierung

Dadurch ist es möglich Authentifikation und Autorisierung in einem simplen Verfahren abzubilden:

- Bei der Registrierung wird für den Benutzer im Frontend automatisch ein User Identifier in Form einer UUID generiert. Zwar könnte diese Generierung von einem Angreifer durch den direkten Zugriff auf die Backend-API umgangen und stattdessen eine beliebige Zeichenkette übergeben werden. Diese "Attacke" wird allerdings durch eine backendseitige UUID-Validierung

unterbunden und würde auch nur zur Schwächung des angreifereigenen, nicht aber anderer User Identifier führen.

- Beim Anmeldevorgang überprüft das Backend, ob die User Identifier Ressource existiert. Wenn das der Fall ist, ist der Anwender dem System bekannt (Identifikation/Authentifikation).
- Analog zur Anmeldung wird auch bei jeder weiteren Backend-Abfrage der User Identifier (im HTTPS Header) mitgesendet. Durch seinen Abgleich mit der Speicherhierarchie kann das System jederzeit entscheiden, ob der Zugriff auf die angeforderte Ressource erlaubt ist (Autorisierung).

Der CSPRNG Faktor

Entscheidend für diesen Konzeptansatz ist die Nutzung eines kryptographisch sicheren Zufallszahlengenerators, denn generell gilt:

Do not assume that UUIDs are hard to guess; they should not be used as security capabilities (identifiers whose mere possession grants access), for example.

RFC 4122: Kapitel 6

Die Dokumentation der verwendeten Code Bibliothek muss also unbedingt dahingehend gesichtet werden. Nur wenn die Entropie der UUIDs hoch genug ist, ist der User Identifier ausreichend vor Kollisionen und einer automatisierten Herleitung geschützt.

Vierte Szene – Die Verschlüsselung

Sensitive Data Exposures: Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such

as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

OWASP: Top 10 2017 Web Application Security Risks

BlockKeeper speichert alle Daten unverschlüsselt im Local-Storage, chiffriert sie aber vollständig vor der Übertragung zum Backend. Für Anwendungsfälle dieser Art hat sich die symmetrische Verschlüsselung und konkret die Anwendung des Advanced Encryption Standard (AES) als de facto Standard etabliert. Bei AES handelt es sich um ein populäres, blockbasiertes Chiffrierungsverfahren. Allerdings ist Verschlüsselung nur die eine Seite der Medaille…

Die Integrität

Grundlegend wichtig ist der Unterschied zwischen Datenintegrität und Datenverschlüsselung, die einander bedingen:

[Symmetric encryption] schemes are not insecure because they leak plaintext information to someone who just intercepts a ciphertext. In fact, most modern schemes hold up amazingly well under that scenario, assuming you choose your keys properly […] The problem occurs when you use encryption in online applications, where an adversary can intercept, tamper with, and submit ciphertexts to the receiver. If the attacker can launch such attacks, many implementations can fail catastrophically, allowing the attacker to completely decrypt messages.

Matthew D. Green: How to choose an AE mode

When we think of encryption, the first thing that generally comes to mind is confidentiality, or keeping data secret. However, there's a second property that's equally important that is integrity, or ensuring that the data we decrypt is the same as the data we encrypted. The way that we do this is through the use of authentication, in the form of a message authentication code (MAC).

Xanderland: That Crypto Code Sample Is Probably Wrong

Der Operation Mode

Die fundamentale Regel Don't use encryption without message authentication birgt in der Praxis aber die Schwierigkeit der Komplexität:

Getting this right is hard all by itself, and if you do it incorrectly, you can leave your implementation exposed to whole classes of attacks, such as timing attacks on the MAC verification process.

Xanderland: That Crypto Code Sample Is Probably Wrong

Zum Glück gibt es einen Ausweg aus diesem Teil des Sicherheits-Dilemmas: Dank fleißiger Experten existieren sogenannte Operation Modes, die Verschlüsselungs- und MAC-Verfahren zuverlässig kombinieren und vergleichsweise leicht anwendbar machen. Bei der Einarbeitung in das Thema Chiffrierung und den AES Algorithmus sollte der Entwickler deshalb nicht das Ökonomie-Dilemma durch das tiefe Studium der detaillierten (mathematischen) Funktionsweise forcieren, sondern sich stattdessen auf die Auswahl eines geeigneten Operation Modes konzentrieren.

In general, the decision of which cipher mode to use is not something most people make every day, but when you do make that decision, you need to make the right one.

Matthew D. Green: How to choose an AE mode

Vorhang.

Dritter Akt

Nach der Definition der Voraussetzungen und der anschließenden Konzeptionierung bleibt für den dritten Akt: Die Implementierung. Sie ist weniger wortreich, ihr Gewicht gering, denn das Publikum wird müde, dem Autor sitzt das Ökonomie-Dilemma im Nacken und dieses Drama handelt von IT-Sicherheit und nicht der Applikation BlockKeeper. Ein schneller Blick auf das Thema Verschlüsselung muss deshalb genügen…

Letzte Szene: Die Umsetzung

Die BlockKeeper Architektur im Schnellüberblick:

- Das Frontend (React) ist als Single Page Application (SPA) realisiert, in der sich der Großteil der Applikationslogik befindet.
- Das Backend (Node.js, heute würde der Autor Python benutzen) dient der Speicherung der verschlüsselten Benutzerdaten.
- Die Kommunikation zur JSON-API des Backends und anderen externen Diensten erfolgt mittels Ajax Anfragen.
- Benutzerdaten werden ausschließlich in verschlüsselter Form zwischen Front- und Backend übertragen, aber unverschlüsselt im Local-Storage des Browsers zwischengespeichert.

Verschlüsselung und Integrität

Wie geschildert sind Datenintegrität und Datenverschlüsselung zwei Seiten derselben Medaille, entsprechende MAC- und Verschlüsselungsverfahren sind deshalb in Operation Modes miteinander verknüpft. Entwickler greifen mittels verbündeter Code Bibliotheken darauf zu und vermeiden so rudimentäre Fehler bei der Kombination.

Authenticated encryption (AE) provides confidentiality and data authenticity simultaneously. An AEAD scheme is usually more complicated

than confidentiality-only or authenticity-only schemes. However, it is easier to use, because it usually needs only a single key, and is more robust, because there is less freedom for the user to do something wrong.

Stack Exchange: Forenbeitrag von Benutzer Dmitry Khovratovich

AEAD Modes stellen nicht nur die benötigte Integrität und Verschlüsselung zur Verfügung, sondern verbergen auch deren Komplexität unter einer einheitlichen Schnittstelle.

Der AES Galois Counter Mode

Bei der Auswahl des AEAD Modes stellt sich zunächst die pragmatische Frage: Welche Modes unterstützt die angestrebte Code Bibliothek?

BlockKeeper nutzt die WebCrypto-API, die im Dezember 2017 nur einen AEAD Mode enthielt, der in allen aktuellen Browsern verfügbar war: AES-GCM.

Galois Counter Mode has quietly become the most popular AE(AD) mode in the field today, [...] if you have someone else's implementation — say OpenSSL's — it's a perfectly lovely mode. [...] Having read back through the post, I'm pretty sure that the 'right' answer for most people is to use GCM mode and rely on a free implementation.

Matthew D. Green: How to choose an AE mode

Die Wahl fiel im Falle von BlockKeeper also leicht. Sollte AES-GCM nicht passend sein oder nicht zur Verfügung stehen, gibt NIST SP 800-38D eine Übersicht über weitere AEAD Verfahren.

Abhängig vom Anwendungsfall erwartet AES-GCM eine unterschiedliche Anzahl an Parametern, die im Dezember 2017 überschaubar gut in den WebCrypto-API Funktionen dokumentiert sind. Verbündete Standards bringen aber Licht ins Dunkel, konkret RFC 5084:

AES-GCM has four inputs: an AES key, an initialization vector (IV [or nonce]), a plaintext content, and optional additional authenticated data (AAD).

AES-GCM generates two outputs: a ciphertext and message authentication code (also called an authentication tag).

The nonce [IV] is generated by the party performing the authenticated encryption operation. Within the scope of any authenticated-encryption key, the nonce value MUST be unique. That is, the set of nonce values used with any given key MUST NOT contain any duplicate values. Using the same nonce for two different messages encrypted with the same key destroys the security properties.

AAD is authenticated but not encrypted. Thus, the AAD is not included in the AES-GCM output. It can be used to authenticate plaintext packet headers [for example].

RFC 5084: Kapitel 1.5

Ein typisches Beispiel dafür, dass die primär für Sicherheitsbibliotheken wichtigen Normen auch wertvolle Informationen für ihre Anwendung liefern.

Vorhang.

Epilog

In der zur Verfügung stehen Produktionszeit ist es nie möglich, ein perfektes Produkt zu entwickeln.

Das Ökonomie-Dilemma

Zu einem perfekten Produkt gehört auch eine vollständige Dokumentation, die zwar der offene Blockkeeper Quellcode technisch, aber der Autor aus Zeitgründen an dieser Stelle prosaisch nicht liefern kann.

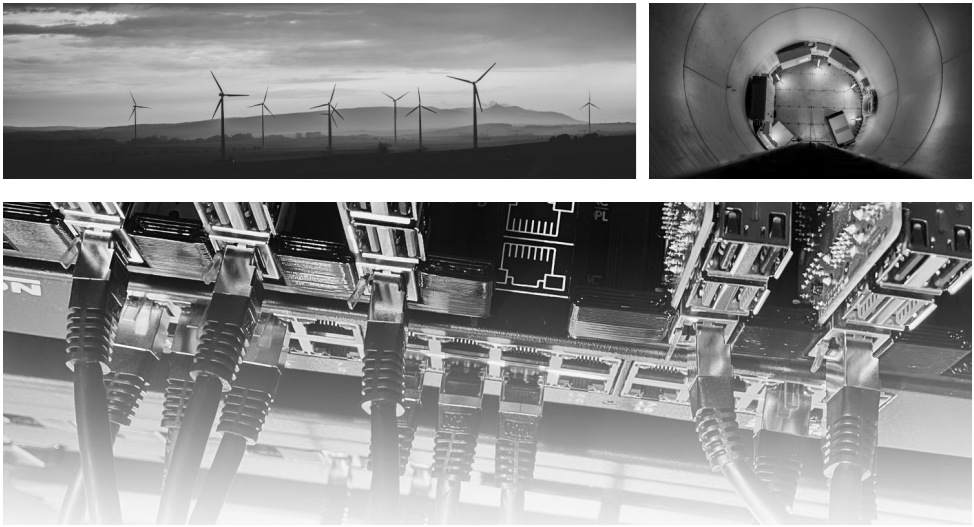
So endet dieses Drama.
Vielleicht nicht als Tragödie. Aber auch nicht als Komödie.

Bitte / Danke

Dieses Drama unterstützen:

- Seit 2021 beschäftigt sich der Autor intensiv mit grünem Edge-Computing auf Basis von Mikrocomputern (SBCs). Zusammen mit seinen Mitgründern hat er vor Kurzem Micro-Colocation.com veröffentlicht, das sich über jeden Raspberry Pi, Odroid, Jetson Nano und Co. Enthusiasten freut. CC lizenzierte Texte wie dieses Drama entstehen innerhalb dieser Unternehmung, werden also durch Buchungen und Teilen von Micro-Colocation.com in sozialen Medien gefördert.
- Technischen Dokumenten immanent sind Fehler beliebiger Art. Obwohl in einem Drama eher verzeihbar, sind Anmerkungen, Erweiterungen, Korrekturen und sonstige konstruktive Kritik an diesem Dokument immer erwünscht.
- Willkommen ist insbesondere Hilfe bei der Übersetzung des deutschen Originals in lesenswertes Englisch und gerne weitere Sprachen. In diesem Zusammenhang vielen Dank an den

kostenlosen Online-Dienst deepl.com, der bei der englischen Übersetzung eine große Hilfe war.



Micro-Colocation.com
Dein Kleinrechner (mit Peripherie) in einer Windkraftanlage/Solarpark

Vielen Dank.

dck

http://dck.one, txt@dck.one

Requisiten

Schriftart

Masken