Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Bachelorarbeit (Informatik)

Schlüsselmanagement und Verschlüsselung für Live-Videostreams

Autoren	Nicolas Da Mutten Daniela Simona Egli Andreas Meier
Hauptbetreuung	Dr. Gürkan Gür

Datum	11.06.2021

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)
Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	**Name Studierende:**
Winterthur, 11.06.2021	Nicolas Da Mutten
Zell, 11.06.2021	Daniela Simona Egli
Thayngen, 11.06.2021	Andreas Meier

Zusammenfassung

Obwohl Streaming-Dienste immer mehr an Bedeutung gewinnen, sind Liveübertragungen für Telekommunikationsanbieter weiterhin ein wichtiges Thema. Denn gerade bei Sport und Reality-TV ist die Lizenzierung von Streaming-Material essenziell, da mit diesen Übertragungen viel Geld erwirtschaftet wird. Es gibt aktuell viele kommerzielle Digital Rights Management (DRM)-Lösungen, aber keine modularen, quelloffenen Systeme und sehr wenig wissenschaftliche Literatur dazu. Deshalb war das Ziel dieser Arbeit, einen Architekturentwurf für ein modulares Key Management System zu entwickeln. Ausserdem sollte ein Prototyp dieser Architektur umgesetzt werden. Dazu wurden zuerst Literaturrecherchen durchgeführt. Anschliessend wurde nach dem Security Development Lifecycle vorgegangen. Dabei wurde eine Bedrohungsanalyse erstellt, welche beim Entwerfen der Architektur verwendet wurde. Der Entwurf sieht zur einfachen Skalierbarkeit des Systems den Einsatz von Microservices vor. Durch diese modulare Architektur können in Zukunft auch andere DRM-Systeme und Verschlüsselungsarten einfacher in die Lösung integriert werden. Auf dieser Basis wurde entschieden, dass der Prototyp mit FFmpeg als Transmuxer, Apple FairPlay Streaming (FPS) als DRM-System und AES-128 als Verschlüsselungsalgorithmus umgesetzt wird. Anhand des Architekturentwurfes wurde ein Key Generator entwickelt, welcher pro Kanal Schlüssel generiert und diese an die Key Server sendet. Die Key Server speichern die Schlüssel in einer zentralen Datenbank und stellen diese den DRM-Systemen zur Verfügung. Der FPS Server holt die Schlüssel beim Key Server ab und stellt diese dem Client per FPS-Schlüsselaustausch zur Verfügung. Die entworfene Architektur zeigte sich als solide und zweckmässige Lösung. Doch gerade in puncto Skalierbarkeit und Ausfallsicherheit hat er noch Potenzial. Beispielsweise ist die Schlüsseldatenbank nicht redundant und stört bei einem Ausfall den Betrieb des gesamten Systems. Weiter können bei der aktuellen Architektur die Schlüssel nicht bereits im Vorfeld an die Clients verteilt werden. Dies kann zu Überlastung und Ausfall des Systems führen. Der Prototyp wurde gemäss Architekturentwurf umgesetzt und getestet. Jedoch funktioniert die Wiedergabe der verschlüsselten Streams nicht, da FFmpeg kein Sample-AES unterstützt und das für FPS benötigt wird. Anhand des Datenstroms und des Schlüsselaustauschs wurde bestätigt, dass die restlichen Funktionen des Prototyps wie gewollt funktionieren.

Abstract

Even though streaming services are becoming increasingly more important, live TV remains an important topic for telecommunication providers. After all, especially with sports and reality TV, licensing is an important topic, as a considerable amount of money is made with such TV contents. Currently there are many commercial Digital Rights Management (DRM) solutions available but no modular, open source systems and only very limited literature on this topic can be found. Therefore, the goal of this thesis was to develop an architectural draft of a modular key management system. Furthermore, a prototype was to be implemented based on this architecture. To this end, a literature survey was carried out. Subsequently the Security Development Lifecycle approach was applied during development. As part of this, a comprehensive threat analysis was carried out, which then was used to design the architecture. To achieve fast and simple scalability, the use of microservices was promoted. Thanks to this modular architecture, other DRM systems and encryption types can easily be integrated into the solution in the future. Based this, the decision was made to implement the prototype with FFmpeg as transmuxer, Apple FairPlay Streaming (FPS) as DRM system, and AES-128 as encryption algorithm. Following the blueprint, a key generator that generates keys per channel and sends them to the key servers was developed. The key servers store the keys in a centralized database and provides them to the DRM systems. The FairPlay Streaming Server fetches the keys from the key server and provides them to the client via an FPS key exchange.
The designed architecture proved to be a solid and practical solution. However, it still shows some room for improvement, especially in terms of scalability and reliability. For example, the key database is not redundant and disrupts the entire system in the event of a failure. Additionally, using the current architecture it is impossible to distribute the keys to the clients in advance. This can lead to system overload or failure. The prototype was implemented according to the plan and tested. The playback of the encrypted streams turned out to not be working, because FFmpeg does not support Sample-AES which is required by FPS. Based on the data stream and key exchange, it was confirmed that the remaining functions of the prototype work as intended.

Vorwort

Diese Bachelorarbeit entstand im Rahmen unseres Teilzeitstudiums in Informatik an der ZHAW School of Engineering in Winterthur. Die Idee für diese Arbeit stammt von Nicolas Da Mutten, der Teilzeit bei Init7 (Schweiz) AG in der Entwicklung des TV-Services arbeitet. Als er das Thema vorstellte, war für Daniela Egli und Andreas Meier schnell klar, dass sie bei dieser Bachelorarbeit mitwirken möchten.
Die Arbeit wurde von Dr. Gürkan Gür betreut. Er arbeitet an der ZHAW im Forschungsschwerpunkt Information Security und als Dozent für IT-Sicherheit. Für die Unterstützung bei unserer Arbeit und seine Flexibilität, besonders zu Zeiten des Coronavirus, möchten wir ihm herzlich danken. Ebenfalls möchten wir uns bei Pascal Gloor von Init7 (Schweiz) AG bedanken. Er hat uns mit seiner Erfahrung und seinem Wissen bei den Meilenstein-Sitzungen wertvolle Inputs geliefert. Weiter danken wir Jonas Meier, Init7 (Schweiz) AG für die Unterstützung bei der Analyse der bisherigen Situation der Umgebung. Ebenfalls sprechen wir Reto Vit von Quickline AG, Bruno Haug und Jean-François Allaire von Swisscom AG für die Auskünfte, wie bei diesen beiden Providern die TV-Verschlüsselung umgesetzt ist, unseren Dank aus. Ein weiterer Dank geht an Dr. Igor Matic, ZHAW Angewandte Linguistik, der uns durch die Schreibberatung der ZHAW unterstützt hat. Danken möchten wir auch allen weiteren Personen, welche die vorliegende Arbeit Korrektur gelesen haben.

Nicolas Da Mutten
Daniela Egli
Andreas Meier

Winterthur, 11. Juni 2021

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Streaming-Dienste wie Netflix, Disney+ oder Amazon Prime gewinnen stetig an Bedeutung [1]. Netflix vergrösserte seine Nutzerbasis von 2018 bis 2020 um mehr als das Doppelte. Für Telekommunikationsanbieter ist Live-TV aber nach wie vor ein wichtiges Tätigkeitsfeld. Inhalte wie Nachrichten, Sport und Reality-Shows findet man noch nicht auf Streaming-Diensten oder diese haben sich dort noch nicht durchgesetzt. Die grosse Beliebtheit von solchen Inhalten führt dazu, dass für diese grosse Beträge ausgegeben werden. So bezahlte Teleclub kürzlich 20 Millionen Schweizer Franken für ihre Lizenzen für Fussballübertragungen an die UEFA [2]. Wegen diesen grossen Summen ist es wichtig, dass diese Inhalte nicht illegal konsumiert werden können. Für Video on Demand (VoD)-Streamingdienste sind funktionierende Kopierschutzmassnahmen ebenfalls zentral, da sie keine weiteren Einnahmequellen wie Werbespots haben. Verschlüsselung und Kopierschutz sind schon für sich zwei anspruchsvolle Themen. Übertragungen in Echtzeit sind ebenfalls nicht trivial. Diese Themen dann auch noch in einem grossen, verteilten System sicher umzusetzen, ist noch anspruchsvoller, da mit wachsender Grösse des Systems auch die Anzahl möglicher Sicherheitslücken zunimmt.

Die Init7 (Schweiz) AG, nachfolgend Init7 genannt, ist ein kleiner, schweizweiter Internetanbieter. Sie bietet ihren Kunden neben reiner Internet-Anbindung auch die Möglichkeit, Fernsehen in Standard Definition (SD)- und High Definition (HD)-Qualität zu empfangen. Um ihr TV-Angebot zu verbessern, will sie nun die Sender der *ProSiebenSat.1 Media* in HD anbieten. Diese verlangt dafür jedoch aus lizenztechnischen Gründen, dass ihre HD-Streams kopiergeschützt verbreitet werden. Um eine möglichst realitätsnahe Analyse zu erhalten, dient die Init7 im Rahmen dieser Arbeit als konkretes Anwendungsbeispiel.

1.2 Ausgangslage

Das TV-Angebot der Init7 beinhaltet zum Zeitpunkt der Veröffentlichung dieser Arbeit 200 Sender (siehe Tabelle 7.2), 90 davon in HD [3]. Diese können mit einer Apple TV App [4], Android TV App [5], sowie beliebigen Multicast-Playern, wie z.B. dem VLC media player, wiedergegeben werden.

Die Init7 nutzt ein von ihr entwickeltes TV-System, das von unterschiedlichen Anbietern Videostreams empfängt. Diese Videostreams werden damit einerseits direkt den Endkunden zur Verfügung gestellt und andererseits in ein anderes Streamingformat (HTTP Live Streaming (HLS)) umgewandelt und aufgezeichnet. Die Init7 transportiert dazu die Streams in ihrem Internet Protocol (IP)-Netzwerk und betreibt mehrere physische Server an verschiedenen Standorten.

1.3 Leitfrage

Zurzeit gibt es keine modularen Streaming-Kopierschutz-Lösungen, welche Open Source zur Verfügung stehen und ein Anbieter wie die Init7 in ihrem System einbauen könnte.

Ziel dieser Arbeit ist es deshalb, einen **Architekturentwurf für eine modulares Key Management System (KMS)** zu entwickeln. Zudem sollen die **Entscheidungen und Überlegungen sauber und nachvollziehbar dokumentiert** werden, damit potentielle Anwender des Systems in der Lage sind, es für ihre Zwecke einzusetzen oder anzupassen.

Ferner soll auch ein **Prototyp** umgesetzt werden, welcher die vorgeschlagene Architektur in den bestehenden Systemen der Init7 demonstriert. Zu diesem Prototyp gehören alle Systeme, welche die Init7 nicht bereits einsetzt und spezifisch für den Kopierschutz benötigt. Anpassungen an bestehenden Systemen sind zwar nötig, aber nicht Teil des Prototyps.

Die vereinbarte Aufgabenstellung, welche dieser Leitfragen zugrunde liegt, befindet sich im Anhang (7.1.1 Aufgabenstellung).

1.4 Übersicht

Kapitel 1 Im ersten Kapitel wird die Relevanz von Live-TV Streaming, Verschlüsselung und Kopierschutz sowie das konkrete Fallbeispiel Init7 aufgezeigt. Zudem definiert dieses Kapitel die Leitfrage und grenzt sie ab.

Kapitel 2 Das Funktionsprinzip der Streamingprotokolle HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (DASH), Microsoft Smooth Streaming (MSS) und Adobe HTTP Dynamic Streaming (HDS) sowie der Digital Rights Management (DRM)-Systeme Apple FairPlay Streaming (FPS), Google Widevine und Microsoft PlayReady wird in diesem Kapitel aufgezeigt. Es erklärt grundsätzliche Prinzipien der Verschlüsselung und gibt einen Überblick über kommerziell eingesetzte Lösungen. Ebenfalls stellt es Methodiken zur Softwareentwicklung vor.

Kapitel 3 In diesem Abschnitt ist die Vorgehensweise dieser Arbeit beschrieben. Es zeigt auf, wie bei der Literaturrecherche vorgegangen wird. Danach wird auf die Methodik der Softwareentwicklung eingegangen. Diese ist nach dem Security Development Lifecycle (SDL) aufgebaut.

Kapitel 4 Die Requirements, die entworfene Architektur, die Threat Analyse, sowie die Implementation und Evaluierung sind in diesem Kapitel beschrieben.

Kapitel 5 Dieses Kapitel fasst die Ergebnisse zusammen, reflektiert die Resultate, bettet diese in den Gesamtkontext ein und gibt Hinweise zu Verbesserungsmöglichkeiten oder zur Weiterverfolgung des Themas.

Kapitel 6 und Kapitel 7 In den letzten beiden Kapiteln finden sich die Verzeichnisse sowie weitere Anhänge.

2 Theoretische Grundlagen

2.1 Verschlüsselung

2.1.1 Verschlüsselungsverfahren

Verschlüsselungsverfahren werden in symmetrische und asymmetrische Verfahren aufgeteilt [6]. Symmetrische Verfahren verwenden sowohl für die Ver- und Entschlüsselung denselben Schlüssel. Dazu muss für den Austausch von verschlüsselten Daten zuerst der Schlüssel geheim übermittelt werden. Zu dieser Art zählen zum Beispiel Data Encryption Standard (DES) oder Advanced Encryption Standard (AES). Bei der asymmetrischen Verschlüsselung haben beide Parteien jeweils ein Schlüsselpaar. Das besteht aus einem sogenannten Private und einem Public Key. Damit die Kommunikation gemeinsam verschlüsselt stattfinden kann, wird zuerst der Public Key den anderen Parteien bekannt gegeben. Der Private Key muss geheim bleiben. Mit dem Public Key des Empfängers verschlüsselt der Sender die Nachricht, danach kann der Empfänger mit seinem Private Key die Nachricht entschlüsseln. Das Verfahren stellt sicher, dass nur der Empfänger die Nachricht entschlüsseln kann, nicht mal der Sender kann sie nachträglich dechiffrieren. In diese Kategorie gehört unter anderem RSA.
Für HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (DASH), Microsoft Smooth Streaming (MSS) und Adobe HTTP Dynamic Streaming (HDS) werden symmetrische Verschlüsselungsverfahren angewendet. Advanced Encryption Standard (AES) ist der Nachfolger von Data Encryption Standard (DES). Es ist der weltweite Standard und eine der beliebtesten Technologien, die für Security-Zwecke eingesetzt werden. AES hat eine Blocklänge von 128 Bit. Die Zahl 128 bei AES-128 steht nicht für die Blocklänge, sondern für die Schlüssellänge. Als Schlüssellänge kann entweder 128, 192 oder 256 Bit gewählt werden. AES-128 wird oft für Video-Streaming benutzt.

2.1.2 Video-Verschlüsselung

Nachfolgend werden die verschiedenen Video-Verschlüsselungsarten erläutert [7, 8, 9]. Eine Methode ist die Fully Layered Encryption, da wird der gesamte Inhalt verschlüsselt. Eine weitere Methode ist die «Permutation Based Encryption», dabei verwendet man verschiedene Verschlüsselungstechniken und -algorithmen, die bestimmte Bytes vertauschen. Bei der «Selective Encryption» verschlüsselt man nur bestimmte Bytes eines Video-Frames und bei der «Perceptual Encryption» wird die Audio- und Videoqualität verschlechtert. Auf diese Methoden wird nachfolgend eingegangen.

2.1.2.1 Fully Layered Encryption (Vollständige Verschlüsselung)

Nach der Komprimierung werden alle Daten mit den traditionellen Standardalgorithmen wie Data Encryption Standard (DES), Rivest-Shamir-Adleman (RSA), International Data Encryption Algorithm (IDEA) oder Advanced Encryption Standard (AES) verschlüsselt [7, 8, 9]. Bei Videodaten geschieht dies Bild für Bild. Das bietet zwar hohe Sicherheit, ist aber in dieser Form für Echtzeit-Videostreaming nicht geeignet, da es sehr rechenintensiv und dadurch langsam ist [10]. Die vollständige Verschlüsselung von Video-Streams wird auch «naive Technik» genannt und ist die einfachste Methode der Video-Verschlüsselung. Jedes einzelne Byte des Moving Picture Experts Group (MPEG)-Streams wird mit den oben genannten Standardalgorithmen verschlüsselt. Durch die Verwendung von Algorithmen wie AES ist die Übermittlung sehr sicher, da bis jetzt kein Algorithmus existiert, der diese brechen kann.

2.1.2.2 Permutation Based Encryption (Permutationsbasierte Verschlüsselung)

Bei der permutationsbasierten Verschlüsselung werden verschiedene Permutationstechniken und -algorithmen verwendet, um die Videoinhalte zu verschlüsseln und zu schützen [7, 8, 9]. Beispiele für die verwendeten Algorithmen sind die Huffman-Kodierung oder die Zig-Zag Permutation. Dabei werden bestimmte Bytes vertauscht. Eine Permutationsliste dient als geheimer Schlüssel, um die Videodaten zu verschlüsseln. Diese Methode ist in der Regel schneller als Fully Layered Encryption, dafür bietet sie auch weniger Sicherheit.

2.1.2.3 Selective Encryption (Selektive Verschlüsselung)

Bei selektiven Verschlüsselungsverfahren werden Algorithmen verwendet, die nur bestimmte Bytes innerhalb eines Video-Frames verschlüsseln [7, 9]. Es werden dabei nur sensible und wichtige Bytes verschlüsselt. Dadurch reduziert sich der Rechenaufwand für die Verschlüsselung und die Entschlüsselung. Der Aufwand und die Sicherheit ist abhängig davon, welche und wie viele Bytes verschlüsselt werden.

2.1.2.4 Perceptual Encryption (Wahrnehmungsbasierte Verschlüsselung)

Mit der wahrnehmungsbasierten Verschlüsselung wird die Audio- und Videoqualität verschlechtert, sodass nach der Verschlüsselung die Inhalte nur noch teilweise wahrnehmbar sind [7, 8, 9]. Der Grad der Verschlechterung kann dabei gewählt werden. Die Verschlüsselungsart ist nicht geeignet für Anwendungen, die eine hohe Sicherheit benötigen. Empfohlen ist die Perceptual Encryption für Pay-per-View (PPV) TV und Video on Demand. Dort kann dem Kunden das Video vor dem Kauf in schlechterer Qualität gezeigt werden. Erst wenn es gekauft wurde, kann es in voller Qualität angeschaut werden.

2.2 Streaming-Protokolle

2.2.1 Multicast

Beim Internet Protocol Television (IPTV) über Multicast werden Audio und Videodaten an mehrere Zuschauer gleichzeitig gesendet. Dadurch sind keine Eins-zu-Eins-Verbindungen erforderlich und die benötigte Datenrate im Quell-Netzwerk wächst nicht mit der Anzahl Benutzer, sondern pro Quellen-Stream. Das impliziert aber auch, dass alle immer dasselbe Signal erhalten, es kann also nur mit Live-TV und nicht Video on Demand (VoD) genutzt werden. Ein weiterer Nachteil ist die Kompatibilität. Multicast benötigt auf den Routern Unterstützung und ist auch häufig von Firewalls gesperrt.

Um einen Multicast-Stream zu erhalten, nutzt der Client das Internet Group Management Protocol (IGMP). Er signalisiert so dem nächsten Router, dass er gerne einen spezifischen Stream erhalten möchte. Dies wird dann von Router zu Router weiter gemeldet, bis die Quelle erreicht wird. In grösseren Netzwerken mit mehreren möglichen Wegen zum Ziel werden dafür die Protokolle Protocol Independent Multicast (PIM) und Distance Vector Multicast Routing Protocol (DVMRP) verwendet [11].

Technisch werden Multicast-Streams über das User Datagram Protocol (UDP) gesendet. Die Datensegmente der UDP-Pakete sind dabei nach dem MPEG Transport Stream (MPEG-TS)-Standard aufgebaut. Dieser Standard wurde von der Moving Picture Experts Group zur kontinuierlichen Übertragung von Audio- und Videodaten entwickelt und wird unter anderen auch bei Digital Video Broadcasting (DVB)-Übertragungen via Satellit, Funk oder Kabelnetz eingesetzt. Alternativ zu MPEG-TS gibt es auch die Möglichkeit, die Streams per Real-Time Transport Protocol (RTP) zu übertragen. Dieses Protokoll ist ein reines IP-Protokoll und hat deshalb einige Vorteile wie eine bessere Fehlertoleranz, optimierter Übertragung in IP-Netzwerken oder effizienterer Nutzung der Bandbreite [12].

2.2.2 HTTP Live Streaming (HLS)

Das Streaming-Protokoll HTTP Live Streaming (HLS) ist eine Entwicklung von Apple und kann auf fast allen Plattformen genutzt werden. Es unterstützt eine dynamische Anpassung der Stream-Qualität an ändernde Netzwerkbedingungen und sowohl Live-Übertragungen als auch Video on Demand.

In Abbildung 2.1 wird prototypisch die Funktionsweise von HLS aufgezeigt. Dabei wird in einem ersten Schritt das Quellsignal von einem Media Encoder in die gewünschten Codecs transcodiert. Grundsätzlich kann man beliebige Codecs verwenden, allerdings gibt es von den Wiedergabegeräten häufig Einschränkungen. Es ist üblich, die von Apple verlangten Codecs Advanced Video Codec (AVC), High Efficiency Video Codec (HEVC), Advanced Audio Coding (AAC) oder Dolby Digital (AC-3) zu verwenden [13]. Der Media Segmenter unterteilt dann im zweiten Schritt die encodierten Datenströme zeitlich. Diese einige Sekunden langen Teildateien heissen Segmente oder Chunks. Zudem erstellt er eine Indexdatei im MP3-URL (M3U)-Format, die sogenannte Playlist, welche alle Segmente auflistet [14]. Der Videoplayer lädt mit Hypertext Transfer Protocol (HTTP) diese Index-Playlist und die darin aufgelisteten Segmente und gibt diese als kontinuierlichen Videostream wieder. Eine Playlist kann ebenfalls auf weitere Playlisten verweisen. Eine solche Master-Playlist listet dann für verschiedenen Datenströme (Video-, Audio- und Untertitelspuren) und unterschiedliche Varianten (Bitrate, Codec) sogenannte Varianten-Playlists auf, welche dann auf die eigentlichen Chunks zeigen. Dadurch kann sich der Client die passende Variante aufgrund der Netzwerkbedingungen und Wiedergabecharakteristika selbst auszusuchen. Er muss auch nur die benötigten Daten laden und nicht jeweils alle Sprach-, Codec- und Bitratenvarianten. Durch die Verwendung von HTTP als Transportprotokoll ist HLS selbst ein Protokoll in der Applikationsschicht des OSI-Schichtenmodells. Dies bietet diverse Vorteile. Dazu zählen die Kompatibilität mit Firewalls und Proxy Servern oder die Nutzung bestehender Content Delivery Network (CDN) Systeme.

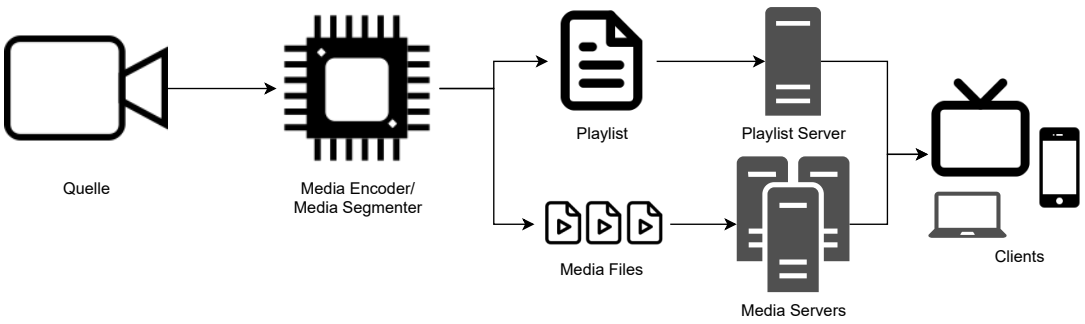


Abbildung 2.1: Funktionsweise von HTTP Live Streaming (HLS)

HLS definiert zwei Verschlüsselungsvarianten: AES-128 und SAMPLE-AES. Beide benutzen zur Verschlüsselung den Advanced Encryption Standard (AES) mit einem Schlüssel mit Schlüsselgrösse 128 Bit im Cipher Block Chaining (CBC)-Modus. AES-128 benutzt AES Public-Key Cryptography Standards #7 (PKCS7)-Padding und verschlüsselt jeweils ein komplettes Segment, es ist also eine vollständige Verschlüsselung (Unterunterabschnitt 2.1.2.1). SAMPLE-AES hingegen verwendet kein Padding und verschlüsselt nur die Medien-Daten (Selektive Verschlüsselung, Unterunterabschnitt 2.1.2.3), bevor diese in die Segmente verpackt werden. Die Metadaten der Segment-Dateien sind also unverschlüsselt. HLS definiert selbst keine Schlüsselaustauschmechanismen oder DRM-Systeme, kann aber pro Segment mehrere solche angeben. In der Spezifikation der Kopierschutzlösung kann auch eine Schlüsselreferenz mitgegeben werden, sodass das DRM-System weiss, welchen Schlüssel es liefern soll.

2.2.3 Dynamic Adaptive Streaming over HTTP (DASH)

Dynamic Adaptive Streaming over HTTP (DASH), oder auch MPEG-DASH, wurde von der Moving Picture Experts Group (MPEG) mit dem Ziel entwickelt, einen Standard für adaptives Bitraten-Streaming zu schaffen und die vielen verschiedenen Standards zum Videostreaming abzulösen [15]. Die Funktionsweise von MPEG-DASH gleicht der von HLS. Anstatt einer M3U-Playlist wird bei DASH eine sogenannte Media Presentation Description (MPD) eingesetzt. Deren Übertragung an den Client ist nicht spezifiziert und kann beliebig erfolgen. Die Daten sind aber ebenfalls in Segmente unterteilt und werden von der MPD referenziert.

Eine MPD ist ein XML-Dokument, welches als DASH-Manifest bezeichnet wird und alle nötigen Informationen zur Wiedergabe des Inhalts enthält [16]. Innerhalb einer MPD gibt es sogenannte «Adaptation Sets» (Verschiedene Inhalte wie Video-, Audio- oder Untertitelspuren), welche «Representations» (verschiedene Varianten desselben Inhalts) enthalten. Diese bestehen wiederum aus «Segment Lists» (einzelne Mediendateien). Da MPDs für längere Streams entsprechend grösser werden, besteht die Möglichkeit, anstelle von Segment Lists sogenannte «Segment Templates» zu verwenden. Mit diesen lassen sich die Form der URLs der Segmente beschreiben, wodurch die tatsächliche URL für jedes Segment hergeleitet werden kann. Wie auch HLS gibt DASH kein DRM vor und kann agnostisch mit verschiedenen Lösungen umgehen.

2.2.4 Microsoft Smooth Streaming (MSS)

Microsoft Smooth Streaming (MSS) wurde von Microsoft unter dem Namen Microsoft Silverlight Smooth Streaming entwickelt und später zu Microsoft Smooth Streaming umbenannt [17]. Das Funktionsprinzip basiert ebenfalls auf HTTP und Segmenten, allerdings unterscheidet sich MSS in der Art, die Videodaten abzuholen. Im Gegensatz zu den bisher vorgestellten Protokollen werden die Segmente in einem speziellen Dateiformat ausgeliefert, welches alle Metainformationen zum Stream und die angeforderte Variante der Daten enthält [18]. Auch hier wird ein Manifest verwendet, welches ebenfalls im XML-Format daherkommt. In dem Manifest wird eine parametrisierte «StreamingIndex URL» und verschiedene mögliche Werte für die Parameter angegeben. Der Player kann sich also aus diesen Parametern seine Segmente selbst auswählen.

2.2.5 Adobe HTTP Dynamic Streaming (HDS)

Adobe HTTP Dynamic Streaming (HDS) ist ein von Adobe entwickeltes Streamingprotokoll, welches vor allem für den Einsatz mit dem Adobe Flash Player und Adobe AIR gedacht war [19]. Seine Funktionsweise basiert ebenfalls auf dem Einsatz von Segmenten. HDS setzt dabei auf fragmented MP4 (FMP4) als Containerformat und XML für die Manifest-Datei [20]. HDS ist nicht sehr verbreitet, da Apple-Geräte keine Unterstützung bieten und der Adobe Flash Player als bevorzugter Player keine Unterstützung mehr erhält.

2.3 Digital Rights Management (DRM)

Für den Schutz von digitalen Medien werden Digital Rights Management (DRM)-Systeme eingesetzt [21]. Diese Systeme verhindern ein unerlaubtes Kopieren von Inhalten. Um dies zu bewerkstelligen, greifen DRM-Systeme auf Verschlüsselungsverfahren zurück. Im Grundsatz funktionieren bei allen DRM-Systemen die Schutzmechanismen gleich und bestehen aus den nachfolgend genannten Komponenten, welche zusätzlich in Abbildung 2.2 aufgezeigt werden. Der **Packager** holt sich die Medieninhalte und strukturiert diese. Die Inhalte werden anschliessend von einem **Encryptor** verschlüsselt und abgelegt. Die Schlüssel werden dem **DRM Server** mitgeteilt und in einer **Schlüsseldatenbank** hinterlegt. Bei den Client Systemen kann es sich entweder um eine native Applikation oder um den HTML5 Webplayer eines Browsers handeln. Die **Client Systeme** holen sich die verschlüsselten Dateien vom Server und fragen den Schlüssel beim DRM Server ab. Dieser teilt den Schlüssel den Geräten mit. Für die Anfrage der Schlüssel wurde von den DRM-Herstellern ein sogenanntes Content Decryption Modul (CDM) auf den Clientsystemen implementiert. Damit können sie garantieren, dass die Anfragen korrekt sind und die Schlüssel von den Benutzern nicht ausgelesen werden können. Die übermittelten Schlüssel werden durch das DRM-System in einen abgesicherten Speicher abgelegt. Das Grundkonstrukt kann noch mit Authentifizierungssystemen erweitert werden.

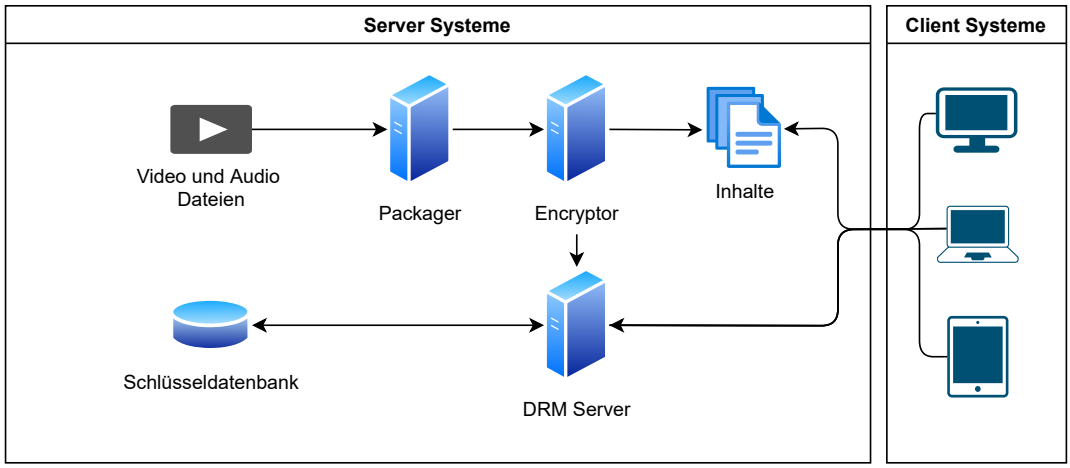


Abbildung 2.2: Funktionalität eines DRM-Systems, adaptiert von [21]

Es existieren für das Streaming von Audio- und Videodaten DRM-Systeme von verschiedenen Herstellern. Die grössten und auch meistverwendeten DRM-Systeme sind Apple FairPlay Streaming (FPS), Microsoft PlayReady und Google Widevine. Die Hersteller haben ihre Systeme hauptsächlich für die eigenen Geräte entwickelt, dadurch muss für eine gute Abdeckung der verschiedenen Geräte eine Kombination von mehreren DRM-Systemen verwendet werden. Die Spezifikationen und Anforderungen der einzelnen Streamingprotokolle gestalten den Aufbau eines Multi-DRM-System sehr komplex. HLS beispielsweise speichert die Chunks standardmässig im Transport Stream (TS) Format und verschlüsselt die Daten im AES-128 Cipher Block Chaining (CBC) Modus. DASH hingegen speichert die Daten normalerweise im MPEG-4 (MP4) Format und verschlüsselt die Daten im AES-128 Counter mode (CTR). Die Chunk Dateien müssten in diesem Szenario redundant in zwei verschiedenen Formaten abgespeichert werden. Sofern noch mehr DRM-Systeme oder Streaming Protokolle hinzukommen würden, müssten die Daten in weiteren Formaten abgespeichert werden [21].

2.3.1 Apple FairPlay Streaming

Bei Apple FairPlay Streaming (FPS) handelt es sich um das von Apple entwickelte DRM-System [22]. Früher wurde es für die Sicherung der iTunes Musikdateien verwendet. Mittlerweile wurde es von Apple soweit weiterentwickelt, dass jetzt auch Drittanbieter dieses System nutzen können [23]. Die Technologie wird ausschliesslich auf Apple Produkten in Kombination mit HLS verwendet. HTTP Live Streaming in Kombination mit FPS unterstützt die Verschlüsselung AES-128 CBC. Die Dokumentationen von Apple FairPlay Streaming zeigen nicht auf welche HLS-Spezifikationen unterstützt werden, gemäss einem Onlineartikel [23] unterstützt es jedoch AES-128 und SAMPLE-AES.

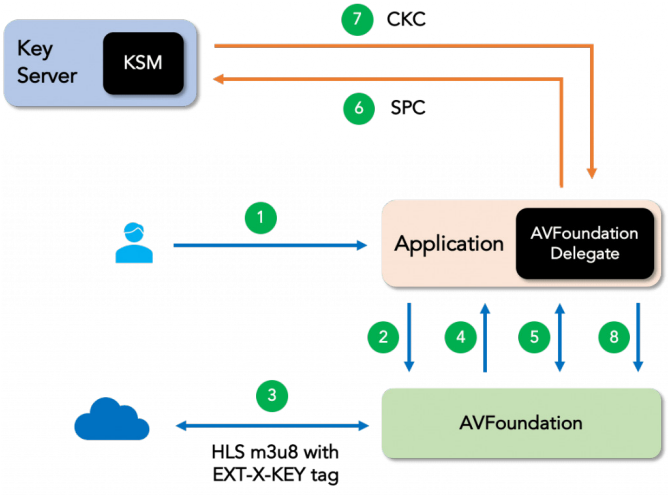


Abbildung 2.3: Kommunikation Apple FairPlay Streaming [24]

In Abbildung 2.3 wird gezeigt, wie die Kommunikation mit Apple FairPlay Streaming funktioniert. Die Kommunikation entspricht dem schon erwähnten DRM-Grundaufbau. Das von Apple bereitgestellte Modul AVFoundation wird als CDM eingesetzt und sorgt dafür, dass die Schlüssel vom Benutzer nicht mitgelesen werden können. Apple stellt für die Entwicklung zwei Frameworks zur Verfügung. Wichtige Funktionen für die Kommunikation mit dem Key Server und dem AVFondation Modul sind im AVFondation Delegate Framework enthalten. Zur Entwicklung eines Key Servers stellt das Key Security Module (KSM) alle nötigen Funktionen für die FPS Kommunikation zur Verfügung. Die zu übermittelnden Nachrichten zwischen dem Server und dem Client sind detailliert spezifiziert. Doch wie die Nachrichten zum Server gelangen, kann der Entwickler selbst festlegen. Das AVFoundation Modul lädt die Playliste herunter und sucht nach einem EXT-X-Key Attribut. Mit diesen Informationen fragt das Modul beim AVFoundation Delegate nach dem Schlüssel. Die App muss nun einen Server Playback Context (SPC) beim AVFoundation Modul anfragen und diesen an den Key Server schicken. Der Key Server öffnet diese verschlüsselte Anfrage und sucht mit den Informationen aus dem SPC nach dem gesuchten Schlüssel. In einem verschlüsselten Content Key Context (CKC) encodiert der Server unter anderem anschliessend den Schlüssel und schickt diesen an den Client zurück. Durch die Verschlüsselung ist nur das AVFoundation Modul in der Lage, den Schlüssel zu lesen. Auf Apple Geräten werden die Schlüssel in der «Secure Enclave» gespeichert. Dies ist ein vom Hauptprozessor abgetrennter Chip, welcher für die Speicherung und Verschlüsselung von sensiblen Daten genutzt wird.

2.3.2 Google Widevine

Das von Google entwickelte DRM-System wird für den Kopierschutz auf chromium-basierten Webbrowsern und Android eingesetzt [25]. Seit 2016 ist die DRM-Technologie auch im Firefox Webbrowser integriert. Für die Verschlüsselung wird der Common Encryption (CENC) Standard und für die Übertragung der Inhalte DASH verwendet. Im Übersichtsdatenblatt von Google ist ebenfalls ersichtlich, dass aufgrund der Inkompatibilität mit den iOS Geräten die CDM Dynamic

Library entwickelt wurde. Die Library kann bei der Entwicklung von iOS Apps genutzt werden und konvertiert die Inhalte in von Apple unterstütze Formate. Hauptbestandteil davon ist der Universal DASH Transmuxer, welcher DASH in HLS konvertiert. Für die Implementation des DRM-Systems stellt Google den Shaka Player und Packager zur Verfügung. Auf den Geräten wurde auch hier ein CDM für die Geheimhaltung der Schlüssel implementiert. Die Kommunikation entspricht ebenfalls dem DRM Grundaufbau in Abbildung 2.2.

2.3.3 Microsoft PlayReady

Gemäss der Produktübersicht auf der Microsoft Seite wird Microsoft PlayReady von den meisten grossen Onlinevideodiensten eingesetzt und ist somit sehr verbreitet [17]. Wie auch bei FPS ist Microsoft PlayReady kein fertiges Produkt, welches man installieren und anwenden kann. Die Server und Programme müssen selbständig entwickelt und implementiert werden. Microsoft stellt mehrere Software Development Kit (SDK) für einzelne Komponenten zur Verfügung. Ausserdem sind der Weg und das Format der Lizenz, welche mit dem Schlüssel übertragen werden, vorgegeben. Der Schlüssel kann anschliessend von der Microsoft PlayReady Komponente, welche auf der Clientseite von Microsoft zur Verfügung gestellt wird, verarbeitet werden. Diese Komponente schränkt den Client auch soweit ein, sodass er den Inhalt nur anschauen, jedoch nicht kopieren kann. Des Weiteren kann dieses DRM-System auch die Abspiellänge und die Wiederholungsmöglichkeiten einschränken. Bei der Verschlüsselung muss der Schlüssel mit einer KeyID im Key Management System (KMS) gespeichert werden. Anders als bei den anderen beiden DRM Systemen wird diese KeyID nicht über die Playliste weitergegeben. Die ID wird hier in einem Header den verschlüsselten Dateien angehängt. Wenn der Client mit der Microsoft PlayReady Komponente die verschlüsselte Datei abspielen will, muss er zuerst beim Lizenzserver mit der KeyID und einer allfälligen Authentifikation den Entschlüsselungsschlüssel abholen. Durch den Einsatz des Microsoft PlayReady Headers ist das System nicht auf ein Streamingprotokoll limitiert. Es kann mit jedem Audio- und Videoformat durchgeführt werden, welches vom Client unterstützt wird.

2.4 Kompatibilität von Streaming- und DRM-Protokollen

Wie in der Tabelle 2.1 ersichtlich ist, sind verschiedene Kombinationen von Streaming- und DRM-Protokollen möglich. Allerdings kann nicht jede Plattform jede Kombination nutzen. Damit möglichst viele Kunden das verschlüsselte Signal nutzen können, muss man also mehrere Protokollkombinationen unterstützen. Ungeschütztes HLS wird von allen Plattformen unterstützt, ungeschütztes DASH von allen ausser den Plattformen Apples. HLS in Kombination mit einem Kopierschutz wird hingegen viel weniger unterstützt, dafür ist hier DASH wesentlich breiter anwendbar.
Verwendet man die beiden Kombinationen HLS + Apple FairPlay Streaming und DASH + Google Widevine, kann man die meisten grossen Plattformen abdecken. Unterstützt man zusätzlich die Kreuzkombination HLS + Widevine, kann man auch noch LG webOS anbieten. Man benötigt dann lediglich zwei Streaming- und zwei DRM-Protokolle, um den Grossteile der Systeme abzudecken. Der Einsatz mehrerer Protokolle ist aber trotzdem gut abzuwägen, da dies mehr Aufwand in Betrieb und Wartung und eine grössere Komplexität für das ganze System bedeutet.

Dies haben auch die Hersteller der Systeme erkannt und ab dem Jahr 2016 mehrere Änderungen implementiert, um Ressourcen einsparen zu können [27]. Dazu wurde der neue Standard Common Media Application Format (CMAF) festgelegt [28]. Der Standard definiert ein neues gemeinsames Dateiformat ISO base media file format (ISOBMFF), welches für die Streamingprotokolle HLS, DASH, MSS und HDS verwendet werden kann. Die Chunk Dateien können im fragmented MP4 (FMP4) Format gespeichert werden und müssen nicht mehr in mehreren Formaten abgelegt werden [23]. Damit die Verschlüsselung auch vereinheitlicht wird, wurde zusätzlich die Common Encryption (CENC) Spezifikation geschaffen. Diese schreibt vor, dass die Videos entweder im AES-128 CTR oder AES-128 CBC Modus verschlüsselt werden müssen. Nutzt man als Dateiformat CMAF und für die Verschlüsselung AES-128 CBC, ist es möglich, ein Multi-DRM-System zu bauen, das für alle Protokollkombinationen dieselben Mediendateien nutzt.

Tabelle 2.1: Kombinationsmatrix von Streamingplattformen, DRM-Lösungen und Platformen, adaptiert von [26]

Geräte-kategorie	Player	HLS	DASH	HLS + FairPlay	HLS + Widevine	DASH + Widevine	DASH + PlayReady	MSS + PlayReady
PCs/ Browsers	Chrome	Ja	Ja	Nein	Ja	Ja	Nein	Nein
	Firefox	Ja	Ja	Nein	Ja	Ja	Nein	Nein
	IE/Edge	Ja	Ja	Nein	Nein	Nein	Ja	Ja
	Safari	Ja	Nein	Ja	Nein	Nein	Nein	Nein
Handys	Android	Ja	Ja	Nein	Nein	Ja	Ja	Ja
	iOS	Ja	Nein	Ja	Nein	Nein	Nein	Nein
Set-top Boxen	Chrome-cast	Ja	Ja	Nein	Nein	Ja	Ja	Ja
	Android TV	Ja	Ja	Nein	Nein	Ja	Ja	Ja
	Roku	Ja	Ja	Nein	Nein	Ja	Ja	Ja
	Apple TV	Ja	Nein	Ja	Nein	Nein	Nein	Nein
	Amazon Fire TV	Ja	Ja	Nein	Nein	Ja	Ja	Ja
Smart TVs	Samsung/Tenzen	Ja	Ja	Nein	Nein	Ja	Ja	Ja
	LG webOS	Ja	Ja	Nein	Ja	Nein	Nein	Nein
	SmartTV Alliance	Ja	Ja	Nein	Nein	Nein	Ja	Ja
	Android TV	Ja	Ja	Nein	Nein	Ja	Ja	Ja

2.5 Umsetzung bei Schweizer Providern

Für die Konkurrenzanalyse wurde online recherchiert, es liess sich aber nicht herausfinden, ob und wie einzelne Anbieter in der Schweiz ihren Content verschlüsseln. Deshalb wurden die Anbieter Genossenschaft GGA Maur, iWay AG, Quickline AG, UPC Schweiz GmbH und Swisscom AG angefragt, wie sie dabei vorgehen. Von iWay ist die Antwort zurückgekommen, dass sie Swisscom Broadcast für ihre IPTV-Lösung verwenden. Mit Quickline AG und Swisscom AG, nachfolgend Quickline und Swisscom genannt, konnte ein Gespräch geführt werden, wie sie ihre TV-Verschlüsselung aufgebaut haben. Im Folgenden ist das Swisscom Broadcast Angebot gemäss den online verfügbaren Informationen von Swisscom analysiert. Danach sind die Auskünfte der beiden Internet Service Provider (ISP) zusammengefasst. Anschliessend wird auf die verfügbaren Lösungen auf dem Markt eingegangen.

2.5.1 Swisscom Broadcast

Swisscom Broadcast bietet mit ihrem IPTV-Angebot Kunden fertige IPTV-Lösungen an [29]. Sie stellen ihren Kunden drei verschiedene Möglichkeiten zur Verfügung: TV-as-a-Service, Headend Services und die Verbreitung von TV-Content über ein kundeneigenes TV-Programm.

TV-as-a-Service TV-as-a-Service bietet eine auf die Kunden zugeschnittene und ausbaufähige IP-basierte TV-Lösung [29, 30]. Es kann dabei zwischen dem Angebot über das IP-basierte Netz, das Kabelnetz und einer hybriden Lösung mit DVB-C und IPTV gewählt werden. Mit TV-as-a-Service können über 600 Sender angeboten werden. Ausserdem kümmert sich Swisscom um die Aufbereitung der Signale und hostet die Plattform.
Jeder Kunde kann sein individuelles Firmendesign und sein Logo für eine individuelle Benutzeroberfläche verwenden. Beim Angebot handelt es sich um „eine schlüsselfertige Gesamtlösung aus einer Hand ohne hohe Eigeninvestitionen“ [31]. Beim IPTV für IP-basierte Netze bietet Swisscom im TV-as-a-Service neben Live-TV auch Live-Pause, Replay, ein Electronic Program Guide und Multi Device an. Des Weiteren werden diverse Zusatzfunktionen wie unlimitierte parallele Aufnahmen oder 4K Ultra High Definition (UHD) angeboten, die der Kunde auswählen kann.

Headend Services Mit diesem Service kann der Kunde seinen Endkunden eine eigene Programmpalette zur Verfügung stellen [31]. Diese kann aus Hunderten von TV-Programmen weltweit zusammengestellt werden. Die Kunden können aus folgenden Übertragungsmöglichkeiten wählen: Streaming für IPTV, Over the Top Content (OTT) oder DVB-C. Das Signal wird dann entsprechend verschlüsselt.

Verbreitung von TV-Content Dieses Angebot richtet sich an Kunden, welche ihre Inhalte auf einem eigenen TV-Sender zur Verfügung stellen wollen [31]. Angeboten wird das durch virtualisierte Playout Services aus der Cloud oder via Satellitenverbindung.

Swisscom kommuniziert bei keinem ihrer Broadcast-Angebote, wie sie genau streamen oder welche Verschlüsselungslösungen sie verwenden [32]. Da jedoch Privatsender in HD angeboten werden, müssen sie eine Verschlüsselungstechnologie einsetzen, da dies von den Sendern so gefordert wird.

2.5.2 Umsetzung bei Quickline und Swisscom

Wie Quickline und Swisscom ihre TV-Verschlüsselung aufgebaut haben, durfte je in einem inoffiziellen Gespräch in Erfahrung gebracht werden. Beide Schweizer Provider kaufen die Verschlüsslung für das DRM-System und Over the Top Content (OTT) ein. Anbieter auf dem Markt dafür sind beispielsweise Verimatrix [33], Viaccess-Orca [34], Google Widevine [35], Synamedia [36] oder Marlin [37]. Auf einige dieser Anbieter wird nachfolgend noch eingegangen. Auf das Content Delivery Network (CDN) wird mit Over the Top Content (OTT)-Protokollen wie HLS oder DASH zugegriffen. Die Cache-Server sind auf mehrere Standorte verteilt, so können Inhalte für den Kunden schneller bereitgestellt werden. Die Streams für OTT werden dabei unverschlüsselt abgelegt und während dem Streaming verschlüsselt. Auch die Schlüssel dazu werden im laufenden Betrieb generiert.
Von den Interviewpartnern wurden noch die folgenden Tipps und Überlegungen für die Arbeit mit auf den Weg gegeben. Ein Multi-DRM sei wichtig, damit möglichst viele Geräte unterstützt werden. Ausserdem soll nicht nur ein nativer Client angeboten werden, sondern auch die Möglichkeit, dass man sich im Browser ebenfalls die Inhalte anschauen kann. Eine wichtige Überlegung sei ebenfalls, wie man Lastspitzen der Schlüsselauslieferungen verhindern kann. Diese könne zum Beispiel nach einem Ausfall des Key Management Systems auftreten, da dann alle Clients gleichzeitig wieder verbinden. Wenn das zur Prime-Time passiert, könnte das zu einer Überlastung des Systems führen. Eine Möglichkeit, dies zu lösen, ist die Schlüssel bereits Tage im Voraus zu erstellen und jedem Kunden für seine häufig genutzten Sender auszuliefern.

2.6 Kommerzielle Lösungen

Auf dem Markt gibt es ein breites Angebot an kommerziellen Lösungen. Keine davon ist quelloffen und deshalb kommen sie für einen Anbieter, der auf Open Source setzt, nicht in Frage. Auf einige dieser kommerziellen Lösungen wird in diesem Kapitel eingegangen.

2.6.1 Verimatrix

Verimatrix bietet eine breite Palette an verschiedenen Lösungen und Produkten im Bereich Video Protection und Schutz von Web- und Mobile-Anwendungen an [38]. Mit ihren Produkten schützt Verimatrix Daten und Inhalte wie Premium-Filme, Live-Sport, sensible Gesundheits- oder Finanzdaten. Nachfolgend wird auf einige Lösungen von Verimatrix eingegangen.

Multi-DRM Das Multi-DRM von Verimatrix ist eine cloud-native DRM-Lösung [39]. Sie ist automatisch skalierbar, flexibel und die Kunden bezahlen nur die Dienste, welche sie auch nutzen. Das System für die Bereitstellung der Lizenzschlüssel ist ausserdem global redundant aufgebaut. Die Endkunden bekommen jeweils die Schlüssel von dem Server, der ihnen am nächsten ist. Das führt zu einer geringeren Latenzzeit. Die gängigen Plattformen, sprich iOS, Android, tvOS, sowie auch die Verwendung in einem HTML5-Webbrowser werden standardmässig

unterstützt. Durch das bereitgestellte Client-Player-Paket können App Entwickler die Lösung einfach in Apps integrieren.

VCAS Lösungen Mit Video Content Authority System (deutsch: Video-Inhalt-Autoritätssytem) bietet Verimatrix ihren Kunden einen sicheren Multiscreen Dienst für DVB [40]. Durch das Angebot von Multiscreen können die Kunden auf mehreren Geräten gleichzeitig dieselben oder verschiedene Inhalte wiedergeben. Mit dieser Lösung kann die digitale Videoübertragung schnell und kosteneffizient implementiert werden. Durch eine gute Skalierbarkeit und zentrale Verschlüsselung können die Betreiber für IPTV auf einfache und preiswerte Weise sichere Inhalte für verschiedene Geräte anbieten. Zusammen mit dem Multi-DRM von Verimatrix können 4K-/UHD Inhalte für die Endnutzer zur Verfügung gestellt werden.

Wasserzeichen Verimatrix bietet sowohl client- wie auch serverseitige unsichtbare Wasserzeichen-Lösungen an [41]. Mit diesen kann Piraterie in wenigen Minuten erkannt werden. Die clientseitige Technologie ermöglicht dem Kunden von Verimatrix Mittschnitte, digitale und analoge Attacken sowie Manipulationen auf Geräte- oder Endkundenebene zu erkennen. Die serverseitige Lösung fügt die Wasserzeichen vor der Verschlüsselung des Streams ein. Mit diesem Wasserzeichen kann erkannt werden, wenn die Inhalte weiter verteilt werden.

2.6.2 Viaccess-Orca

Zum Schutz von Inhalten bietet Viaccess-Orca Lösungen sowohl Smartcard-basiert wie auch mit anderen Authentifizierungsverfahren an [34, 42, 43, 44]. Viaccess-Orca bietet Conditional Access System (CAS), DRM und Multi-DRM an, auch als «as-a-Service». Der Begriff CAS bezeichnet die eingesetzten Systeme beim Pay-TV für die Ver- und Entschlüsselung. Dadurch, dass die Lösung als Service betrieben und gewartet wird, werden auch lediglich der Wartungsaufwand und die aktuellen Kosten verrechnet. Viaccess-Orca gibt an, dass sie die Inhalte der Anbieter durch branchenführende Sicherheitstechnologien schützen. Was das jedoch bedeutet, kann nicht ausgemacht werden. Neben dem eigenen VO-DRM werden auch Microsoft PlayReady, Apple FairPlay Streaming und Google Widevine vollständig unterstützt [45].

2.6.3 Synamedia

Synamedias OTT-Security ist eine ganzheitliche Lösung, welche komplementäre Security-Technologien und -Services nutzt [36, 46]. So können Fernsehinhalte geschützt und nur für Kunden zur Verfügung gestellt werden, die auch dafür bezahlen. Ebenfalls bietet OTT-Security SDKs für verschiedene angeschlossene Geräte wie eine Set-Top-Box (STB), ein Tablet oder Smartphone. Auch werden verschiedene Konsummodelle unterstützt, darunter lineare Kanäle, Video on Demand oder Download-and-Go. Weiter wird die Möglichkeit geboten, dass nur eine bestimmte Anzahl Sessions gleichzeitig von einem Kundenkonto Inhalte konsumieren können. In-Home-Nähe, also dass Inhalte nur an einer bestimmten Adresse beziehungsweise in einem gewissen Radius um diese herum konsumiert werden können, kann ebenfalls erzwungen werden, was das Teilen von Kundenkonten zusätzlich erschwert. Durch das Angebot von Schlüsselrotation, ereignisbasierten Schlüssel und separate Schlüssel für Audio- und Videoinhalte wird die Sicherheit zusätzlich erhöht.
Mit dem Digital Content Manager (DCM) Packager von Synamedia werden Standard-DVB/MPEG-Kanäle in geeignete Adaptive Bitrate (ABR)-Formate konvertiert und entsprechende Manifest- und Indexdateien generiert [47]. Technologien mit ABR sind Apple HLS, Microsoft Smooth Streaming und MPEG-DASH. Es werden auch die gängigen DRM-Systeme wie Apple FairPlay Streaming, Microsoft PlayReady oder Google Widevine unterstützt.

2.6.4 Marlin DRM

Marlin DRM ist eine Open-Standard Spezifikation für den Schutz von digitalen Inhalten [37, 48]. Es wurde 2005 von Intertrust, Panasonic, Philips, Samsung und Sony gegründet, um einen offenen Standard und maximale Flexibilität zu bieten. Mit dem Marlin DRM können nicht nur HD- und UHD-Inhalte geschützt werden, sondern zum Beispiel auch Audio, eBooks oder Games. Wenn ein Unternehmen Marlin zum Schutz von Inhalten verwenden möchte, kann es mit verschiedenen Anbietern dafür zusammenarbeiten, die eine Lösung entwickelt und auf dem Markt haben. Diese Lösungen sind jedoch nicht kostenlos und Open-Source erhältlich, deshalb kommen sie für den Use-Case der Arbeit nicht in Frage.

Marlin DRM von Intertrust Die Marlin DRM konforme Lösung von Intertrust bietet einen gehosteten Service und SDKs für verschiedene Mobile-Apps und eingebettete Geräte. Marlin DRM von Intertrust war eine der ersten Implementationen, die vollständig auf der Marlin-Spezifikation basiert. In Japan, China, Südostasien und Europa wird die Lösung weit verbreitet eingesetzt.

Intertrust ExpressPlay DRM ExpressPlay DRM ist ebenfalls eine Lösung von Intertrust. Es ist ein integrierter und cloudbasierter Dienst, der Marlin kompatibel ist. Der Multi-DRM-Service unterstützt alle wichtigen DRMs: Apple FairPlay Streaming, Microsoft PlayReady, Google Widevine, Adobe Primetime und Marlin DRM. Die Lösung unterstützt die gleichzeitige Nutzung von Millionen von Zuschauern, sodass auch bei den grössten Live-Events genügend Kapazitäten zur Verfügung stehen. Die Systeme sind Geo-Redundant aufgebaut und haben eine Failover-Option, sodass eine hohe Verfügbarkeit gewährleistet wird.

Intertrust ExpressPlay XCA Der cloudbasierte Dienst ExpressPlay XCA bietet den Schutz von Rundfunkinhalten und nutzt dazu den Marlin DRM Standard. Es ermöglicht Pay-TV Betreibern und TV-Broadcastern Premium- und UHD-Inhalte sicher auf Smart-TVs zu übertragen. Dies ohne dass eine STB oder andere externe Sicherheitshardware benötigt werden.

2.6.4.1 Marlin Client SDKs

Marlin Client SDKs sind für eine Vielzahl von Endgeräten verfügbar [48]. Der Client kann in Smart-TVs oder STBs eingebettet werden. Es gibt Binary SDKs für Android und iOS. Im Allgemeinen sind die Vorteile der Verwendung des ExpressPlay Binary SDK, dass eine sichere Implementierung des Marlin-DRM für Android und iOS bereitgestellt wird. Des Weiteren wird dem Anbieter das Streaming von HLS und DASH-CENC für Android und iOS ermöglicht. Unterstützt werden sowohl Streaming- als auch Download-Szenarien. Es wird ebenfalls die iOS Jailbreak und Android Root Erkennung unterstützt. Weiter kann auch ein IPTV-Client integriert werden, an den die Inhalte per Multicast geliefert werden.

Android Android unterstützt Google Widevine native DRM für die Android Version 4.4 und höher. Auf einigen Geräten und älteren Android Versionen ist Google Widevine nicht nativ verfügbar. Für diese Fälle kann «ExpressPlay Binary SDK für Android» verwendet werden. So wird die Wiedergabe von HLS- und MPEG-DASH-Inhalten auch auf diesen Geräten unterstützt.

iOS iOS-Geräte, welche das Apple FairPlay Streaming DRM unterstützen, erfordern das Streamen von Inhalten im HLS-Format, entweder MPEG-TS mit Sample-AES-Verschlüsselung oder FMP4 mit AES-CBCS. Wie bei Android kann das «ExpressPlay Binary SDK für iOS» in eine native Anwendung integriert werden. So kann die Wiedergabe von geschützten HLS- und MPEG-DASH-Inhalten auf iOS-Geräten sichergestellt werden.

2.7 Weitere Grundlagen

2.7.1 Security Development Lifecycle (SDL)

Der Security Development Lifecycle ist ein Konzept für die Entwicklung von sicherer Software. Es wurde von Microsoft im Jahr 2005 eingeführt [49]. Der SDL selbst legt nur die Phasen des Prozesses fest. Der hier gezeigte SDL entspricht den Anpassungen aus den Vorlesungsunterlagen des »Software and System Security (SWS) 1» Unterrichts [50]. Er besteht aus mehreren Schritten, welche hier und in der Abbildung 2.4 erklärt werden.

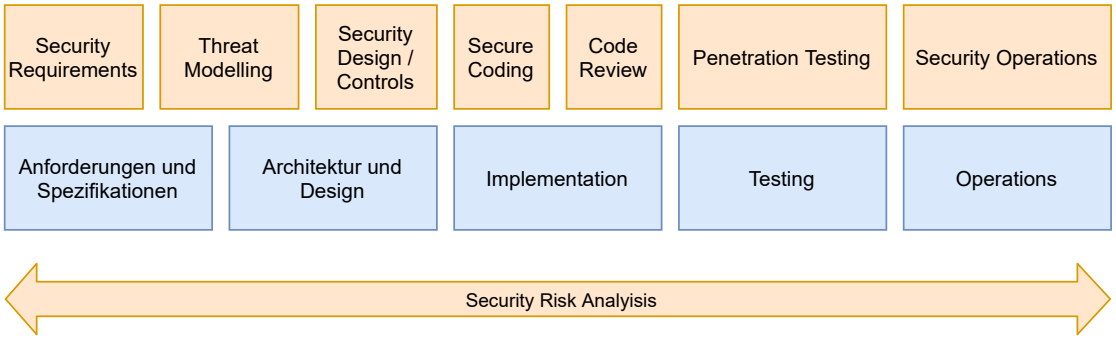


Abbildung 2.4: Aufbau des Security Development Lifecycle (SDL)

Die blauen Boxen stehen für die Standardabläufe in der Softwareentwicklung ohne den Einsatz des SDL Konzepts.

1. **Anforderungen und Spezifikationen**: In der Anforderungsphase werden mit den Stakeholdern die Wünsche und Ansprüche erfasst. Diese werden überprüft und zu Anforderungen umformuliert und legen somit die Ziele und Erwartungen an das Projekt fest.
2. **Architektur und Design**: In der Entwurfsphase wird aufgrund der Anforderungen eine Systemarchitektur festgelegt und visualisiert. Für die Darstellung werden oft Architekturdiagramme und Sequenzdiagramme verwendet.
3. **Implementation**: In der Implementationsphase wird die eigentliche Software geschrieben und mit Code Reviews überprüft. Ausserdem werden einzelne Komponenten mit Unit Tests überprüft.
4. **Testing**: Bevor die Software veröffentlicht wird, wird diese mit Integrationstests und Systemtests überprüft.
5. **Operations**: In dieser Phase wird das System gewartet und überprüft. Mithilfe von Monitoring- und Alarmierungssystem kann die Software überwacht werden.

Die orangen Boxen stehen für die zusätzlichen Schritte, welche eingebaut werden müssen, um eine Software zu entwickeln, bei der mehr Wert auf Sicherheit gelegt wird.

1. **Security Requirements**: Bei diesem Schritt werden sicherheitsrelevante Anforderungen definiert. Die Security Requirements bilden eine Ergänzung zu den funktionalen und nichtfunktionalen Anforderungen. Diese werden ohne Hilfsmittel mit Brainstorming und anderen Ideenfindungstechniken festgelegt. Sie bilden die Basis für das Threat Modelling und werden durch dieses mit weiteren Anforderungen ergänzt.
2. **Threat Modelling**: Threat Modelling ist eine Technik, um sicherheitsrelevante Fehler in der Architektur und dem Design zu finden. Bei dieser Technik geht man aus der Sicht des Angreifers vor. Es werden mögliche Angriffsziele und deren Schwachstellen definiert. Für die entdeckten Schwachstellen werden in einem zweiten Schritt neue sicherheitsrelevante Anforderungen definiert. Diese Anforderungen sollen die Schwachstellen abschwächen oder sogar den Angriff verhindern.

3. **Security Design** / **Controls**: Basierend auf den Security Requirements werden in diesem Schritt mögliche Kontrollmechanismen und Designs festgelegt. Es wird dabei ein Kompromiss zwischen Kosten und Nutzen der Mechanismen erarbeitet.
4. **Secure Coding**: Wie auch bei der Implementation wird in dieser Phase der eigentliche Code geschrieben. Es wird dabei darauf geachtet, dass der Code in einem sicheren Verfahren implementiert wird. Ausserdem dürfen nur Technologien und Bibliotheken verwendet werden, welche vom Entwickler verstanden werden.
5. **Code Review**: Es sollte ein Code Review durchgeführt werden, bei welchem der Sicherheit eine besondere Aufmerksamkeit geschenkt wird. Diese Reviews werden manuell von einer Drittperson oder auch automatisiert durchgeführt.
6. **Penetration Testing**: Nach der Entwicklung der Software sollte man beim Testen auch sogenannte Penetrationstests durchführen. Bei diesen versetzt man sich in die Lage des Angreifers und überprüft die Software auf Schwachstellen. Ausserdem wird versucht, diese Schwachstellen auszunutzen.
7. **Security Operations**: Das Projekt sollte auch im laufenden Betrieb überwacht und gewartet werden. Zu den sicherheitsrelevanten Arbeiten gehören das Aktualisieren der Systeme, die Sicherung der Daten und das Überwachen des Netzwerks.
8. **Security Risk Analysis**: Die Security Risk Analysis ist ein fliessender Prozess, welcher während des gesamten Projekts stattfindet. In diesem Schritt werden die erkannten Risiken und Schwachstellen analysiert. Das entstehende Risiko wird berechnet und geprüft.

2.7.2 Zwölf-Faktor-Methode

Die Zwölf-Faktor-Methode ist unabhängig von der Programmiersprache für verschiedene Apps geeignet und sehr flexibel [51, 52]. Die Methode soll dem Entwickler eine Hilfestellung für die Implementierung einer Cloud Native App sein. Es werden diejenigen Elemente davon verwendet, die für die jeweilige App benötigt werden und es können auch weitere hinzugefügt werden, beispielsweise wenn eine Datenbank verwendet wird. Die zwölf Faktoren sind:

I **Codebase**: Code ist zentral in einer Source Code Verwaltung und so haben alle Entwickler jederzeit den gleichen Stand zur Verfügung.

II **Abhängigkeiten**: Abhängigkeiten müssen explizit definiert und kontrolliert sein.

III **Konfiguration**: Die Konfigurationsdaten befinden sich ausserhalb des Codes.

IV **Backing Services (Unterstützende Dienste)**: Dienste wie Datenbanken, Caches, etc. sollen austauschbar sein.

V **Build, release, run (Stages trennen)**: Während der Laufzeit dürfen am Code keine Änderungen vorgenommen werden. Die verschiedenen Zustände müssen klar getrennt werden, sodass der Run-Zustand jederzeit stabil ist.

VI **Prozesse**: Die App wird als einer oder mehrere zustandslose Dienste ausgeführt.

VII **Port Binding**: Jeder genutzte Dienst wird über Ports angebunden und exportiert.

VIII **Concurrency (Nebenläufigkeit)**: Die Nebenläufigkeit ist durch einzelne Prozesse höher, diese können individuell gestartet, beendet und skaliert werden.

IX **Verfügbarkeit**: Durch die einzelnen Prozesse können diese schnell gestartet und gestoppt werden.

X **Dev-Prod-Gleichheit**: Development, Staging und Produktion sollen möglichst ähnlich sein.

XI **Logs**: Logs sollen als Ereignisstrom und in einem separaten Service behandelt werden.

XII **Admin-Prozesse**: Administrative und Management-Aufgaben sollen als einmalige Aktion behandelt werden.

2.7.3 Threat Analysis

Die Threat Analyse ist ein Prozess, welcher Schwachstellen aufzeigen soll. Es gibt dafür verschiedene Vorgehensweisen. Der Ablauf, der unten beschrieben wird, ist aus dem Software and System Security (SWS) Unterricht der ZHAW abgeleitet [50].

1. Definieren der Unternehmensziele der Anwendung
2. Definieren der Sicherheitsziele der Anwendung
3. Informationen über das System und die Daten ermitteln
4. Vermögenswerte des Systems ermitteln
5. Analyse und Zersetzung des Systems mithilfe von Data Flow Diagram (DFD) (Datenflussdiagramm) und Netzwerkdiagrammen
6. Ermitteln der Bedrohungstypen und -faktoren
7. Bedrohungen mit dem STRIDE Ansatz ermitteln
8. Requirements für die Entschärfung der Bedrohungen definieren.

2.7.3.1 STRIDE Threat Model

STRIDE ist ein englisches Akronym und wird verwendet, um Bedrohungen in Kategorien zu gruppieren [53, 50]. Es hilft dabei, Gefahren zu erkennen und einzuordnen. STRIDE steht für:

Spoofing Identity (Identitätsfälschung) Der Angreifer gibt vor, dass er jemand anders ist. Identitätsspoofing ist zum Beispiel der illegale Zugriff mit Benutzername und Passwort eines anderen Benutzers.

Tampering with Data (Manipulation von Daten) Die böswillige Veränderung von Daten oder Code, beispielsweise das Anpassen von Daten in einer Datenbank oder eine Änderung der Daten, die über eine unsichere Netzwerkverbindung gesichert werden.

Repudiation (Verleugnung) Hier ist das Ziel, dass die Handlungen verschleiert sind, sodass nicht nachgewiesen werden kann, wenn Ressourcen manipuliert wurden. In diese Kategorie gehört zum Beispiel ein Datenbank-Administrator, der in der Datenbank seinen Lohn anpasst und die Logs manipuliert, um die Spuren zu verwischen.

Information Disclosure (Offenlegung von Informationen) Der Angreifer hat Zugriff auf geschützte Informationen und kann diese lesen, obwohl er dazu nicht berechtigt wäre. Beispiele für solche Informationen sind Dateien mit Passwörtern oder das mitlesen von Kreditkarteninformationen, die über eine unsichere Netzwerkverbindung übertragen werden. Auch das Auslesen von Daten aus einer Datenbank mit SQL-Injection gehört dazu.

Denial of Service (DoS) (Unterbrechung der Dienstleistung) Der Angreifer unterbindet durch DoS-Attacken, dass ein berechtigter Benutzer Zugriff auf den Dienst hat. Zum Beispiel in dem ein Webserver zum Absturz gebracht wird.

Elevation of Privilege (Erhöhung der Benutzerrechte) Ein nicht berechtigter Benutzer schafft sich einen Zugriff als Benutzer mit genügend Rechten, um beispielsweise ein ganzes System zu gefährden oder zu zerstören.

Die STRIDE-Kategorien werden anhand der unten stehenden Tabelle 2.2 auf die Data Flow Diagram-Elemente angewendet, um die Bedrohungen für das System zu identifizieren. Das X zeigt an, welche Bedrohungskategorien auf welche Elemente angewendet werden sollen.

Tabelle 2.2: Anwenden von STRIDE auf das Data Flow Diagram, aus Software and System Security [50].

DFD Element Type	S	T	R	I	D	E
External Entity	X		X			
Data Flow		X		X	X	
Data Store		X	X	X	X	
(Multiple) Process	X	X	X	X	X	X

3 Vorgehen

Um das Wissen und das Verständnis für die Materie aufzubauen, wurde zuerst eine Literaturrecherche durchgeführt. Das Vorgehen in der Softwareentwicklung ist gemäss den Schritten des SDLs gegliedert. Diese Struktur hat auch Einfluss auf den Aufbau der schriftlichen Arbeit.

3.1 Methodik der Literaturrecherche

Das Thema Videoübertragung ist komplex, da es ständig im Wandel ist und sehr viele verschiedene Lösungen auf dem Markt existieren. Deshalb bestand der erste Teil der Arbeit aus der allgemeinen Informationssuche nach relevanten Quellen über die Themengebiete HLS und Videoübertragung. Mit diesem Basiswissen wurde eine vertieftere Literaturrecherche gestartet. Bei der Suche nach Fachartikeln waren die Suchmaschinen Google Scholar und ZHAW Swisscovery eine grosse Unterstützung. Im ersten Suchdurchlauf lag der Fokus hauptsächlich auf den folgenden Suchbegriffen:

- HTTP Live Streaming
- HLS Encryption
- Video Encryption
- Stream Encryption
- Key Management
- TV Encryption
- Apple FairPlay Streaming

Die gefundenen Artikel wurden im Projektteam aufgeteilt und tiefergehend studiert. Mit dem erlangten Wissen konnte das Thema stärker eingegrenzt werden und es wurde damit eine verfeinerte Suche durchgeführt. Im Fokus der Recherche standen nun auch andere Übertragungstechnologien wie DASH und MSS. Ausserdem lag ein neuer Schwerpunkt auf dem Thema DRM-Technologien. Die erweiterte Suche hatte zusätzlich noch Informationen über Multicast-Videoübertragungen und Möglichkeiten für die Verschlüsselung des Multicast Signals hervorgebracht.

3.2 Methodik der Softwareentwicklung

Die Sicherheit eines Softwareprojekts sollte schon zu Beginn der Arbeit einen grossen Stellenwert haben. Deshalb wurde in der Bachelorarbeit für die Entwicklung der Security Development Lifecycle verwendet. Aufgrund der zeitlichen Begrenzung der Arbeit und der Entwicklung eines Prototypen wurde auf einzelne Schritte des SDL verzichtet. In der Arbeit wurden die Security Requirements festlegt. Die Festlegung der Security Controls und Designs fanden nicht statt, jedoch wurden diese bei der Erstellung der Architektur beachtet. Das anschliessende Threat Modelling sorgte dafür, dass die Schwachstellen in der Architektur und im Projekt reduziert werden konnten. Beim Programmieren wurde auf die sicherheitsrelevanten Aspekte Rücksicht genommen und es wurden zusätzlich manuelle Code Reviews durchgeführt. Auf das Penetration Testing und auf die Festlegung der Security Operations wurde aus den oben erwähnten Gründen verzichtet. Eine reduzierte Security Risk Analysis bildete den Abschluss der Arbeit.

3.2.1 Requirements

Mit dem Ziel, die Requirements festzulegen, fand eine Sitzung mit dem Chief Technology Officer (CTO) von Init7 statt. Im Fachgespräch konnten die Anforderungen an das System ermittelt und anschliessend mit einem Brainstorming erweitert werden. Für die Festhaltung der funktionalen und nicht-funktionalen Anforderungen wurde nach dem Standard ISO 25010:2011 vorgegangen, welcher Qualitätsmerkmale und Methoden für eine gute Softwarequalität definiert [54]. Die Requirements wurden gemäss dem Akronym FURPS gegliedert und überprüft. Bei der Ermittlung der Anforderungen wurde ebenfalls die Zwölf-Faktor-Methode beachtet, damit die entwickelte Lösung mit wenig Aufwand in der Cloud zur Verfügung gestellt werden kann.

3.2.2 Analyse des bisherigen Zustands

Damit eine neue Architektur für das System entworfen werden konnte, war eine Analyse der momentanen Software und Architektur notwendig. Ein Mitglied des Projektteams hatte bereits beruflich mit dem TV-System von Init7 zu tun. Mithilfe vorhandener Architekturdiagramme und der Untersuchung des bestehenden Codes konnte einen Überblick über das System verschafft werden. Die Erkenntnisse dieser Analyse wurden in einem neuen Architekturdiagramm der bisherigen Lösung visuell dargestellt. Um die bestehenden Schwachstellen zu ermitteln, wurde ein einfaches Threat Model erstellt. Für die Visualisierung wurden Datenflussdiagramme und Netzwerkdiagramme verwendet. In einem Fachgespräch mit einem System Engineer von Init7 konnten die Details für die Erstellung des Netzwerkdiagramms ermittelt werden. Damit auch die Schwachstellen des bisherigen Zustands untersucht werden konnten, wurde auch ein Data Flow Diagram erstellt.

3.2.3 Architekturvorschlag

Die ermittelten Software und Security Anforderungen wurden vor der Erstellung der Architektur geprüft und im Team besprochen. Da ein Teil der Software bereits existierte, wurde als Grundlage für die Architektur das zu Beginn erarbeitete Architekturdiagramm der bisherigen Situation verwendet. In einer Teamsitzung wurde das Diagramm erweitert und mit sicherheitsrelevanten Bedenken hinterfragt. Die daraus resultierenden Diagramme wurden in einem Milestone Meeting mit dem CTO der Init7 besprochen.

3.2.4 Threat Analysis

Die Grundlagen für die Threat Analyse, wie zum Beispiel das DFD Diagramm wurden von einzelnen Projektmitgliedern erarbeitet und dann in einer Gruppenbesprechung überarbeitet. Aus dem DFD Diagramm konnten anschliessend über 100 mögliche Schwachstellen herausgearbeitet werden. Dafür wurde die STRIDE Methode verwendet [53]. Die einzelnen Buchstaben des Akronyms wurden gemäss Tabelle 2.2 den Elementen des DFDs in Abbildung 4.7 zugeordnet. Die möglichen Schwachstellen wurden in mehreren Sitzungen im Projektteam spezifiziert.

3.2.5 Implementation

Im Rahmen der Bachelorarbeit sollte nur ein Prototyp umgesetzt werden, deshalb wurde auf agile Methoden verzichtet. Die einzelnen Komponenten wurden jeweils in einem eigenem Git Repository entwickelt. Vor den eigentlichen Entwicklungsarbeiten wurden die Reposorities als Templates eingerichtet. Die Komponenten Key-Server, Auth Server und FairPlay Streaming Server bauen auf dem Django REST Framework auf. Die Planung und Aufteilung der Arbeit wurde mit Microsoft Planner vorgenommen. In einem wöchentlichen Meeting konnten die aktuellen Stände und auch die Schwierigkeiten besprochen werden. Nach dem Abschluss einer Komponente wurden durch Code Reviews und Pull Requests die Qualität des Codes nochmals überprüft. Bei der Implementierung wurde darauf geachtet, dass die Faktoren der Zwölf-Faktor-Methode eingehalten werden. Deshalb wurden vor der Implementation aus den zwölf Faktoren die folgenden Ziele festgelegt:

I **Codebase**: Der Code wird in GitHub Enterprise der ZHAW gespeichert und ist jederzeit für alle Entwickler einsehbar.

II **Abhängigkeiten**: Die verwendeten Python-Bibliotheken sind bei allen Komponenten in der Requirements.txt Datei festgehalten. Bei der Demo-App gibt es keine Abhängigkeiten, deshalb wird auf eine externe Definition verzichten.

III **Konfiguration**: Die Konfiguration der Komponenten ist immer ausserhalb des Codes in der Datei .env abgespeichert.

IV **Backing Services**: Die Komponenten laufen alle in Docker Containern und sind nicht für spezifische Backing Services entwickelt.

V **Build, release, run**: Die Zustände im Code sind durch die Dockerfiles getrennt.

VI **Prozesse**: Der Code hat niemals einen Zustand. Alle Zustandsänderungen werden in das Filesystem oder in Datenbanken abgespeichert.

VII **Port Binding**: Dieser Faktor kann aufgrund der vorgegebenen Struktur von FFmpeg nicht komplett umgesetzt werden.

VIII **Concurrency (Nebenläufigkeit)**: Für die Nebenläufigkeit werden Micro-Services und asynchrone Aufgabenwarteschlangen eingesetzt.

IX **Verfügbarkeit**: Die Komponenten werden in einzelne und voneinander unabhängige Services aufgeteilt.

X **Dev-Prod-Gleichheit**: Development, Staging und Produktion sind sich sehr ähnlich. Sie unterscheiden sich nur durch die Konfigurationen in den .env und docker-compose.yml Dateien.

XI **Logs**: Dieser Faktor wird aufgrund der Implementierung eines Prototyps nicht umgesetzt.

XII **Admin-Prozesse**: Für administrative Aufgaben werden spezifische Kommandos definiert.

3.3 Methodik der Evaluierung

3.3.1 Key Rotation

Das Schlüsselverwaltungssystem muss zuverlässig arbeiten und die Schlüsselanfragen zeitgerecht beantworten können. Aus diesem Grund wurde eine Simulation erstellt, mit welcher die Last mit verschiedenen Key Rotation Perioden geprüft werden konnte. Die Simulation wurde mit Matlab durchgeführt und ausgewertet. Mithilfe von Zufallszahlen und einem maximalen Wachstum von drei Konsumenten pro Minute wurde die Zuschauerzahl simuliert. Um die Simulation zu vereinfachen, wurde davon ausgegangen, dass ein Kunde immer weiterschaut, wenn er einmal dazu gekommen ist. Mit diesen Werten wurde anschliessend berechnet, wie viele Schlüsselanfragen innerhalb von 24 Stunden anfallen. Die Anzahl der Anfragen wurden anschliessend auf «Anfragen pro Stunde» umgerechnet. Der Plot dieser Simulation zeigte schliesslich auf, welche Key Rotation Perioden vorteilhaft sind. Der Code ist im Anhang unter «Simulation Key-Anfragen (Code)» ersichtlich.

3.3.2 Risiko Analyse

Um den Erfolg des Projekts zu prüfen, bildete eine Evaluation den Abschluss der Arbeit. Auch wenn in der Arbeit nur ein Prototyp entwickelt wurde, konnte dennoch untersucht werden, ob dieser Prototyp den gestellten Anforderungen entspricht. Die einzelnen Schritte des SDLs wurden deshalb analysiert. Zu Beginn der Arbeit wurden mehrere Security Requirements definiert. Um einen ersten Eindruck zu bekommen, wurde die Umsetzung dieser Anforderungen betrachtet. Konkret wurde in einer Analyse ermittelt, ob und wie diese Anforderungen umgesetzt wurden. Aufgrund der zeitlichen Beschränkung wurden in der Evaluierung jedoch keine Penetrationtests durchgeführt. Damit die Sicherheit trotzdem gewährleistet werden konnte, wurde das

Threat Model und die darin gefundenen Schwachstellen geprüft. Dabei wurde untersucht, ob die Schwachstellen behoben worden waren. Falls es noch offene Schwachstellen gab, wurde eine Analyse erstellt, welche das Risiko der Schwachstelle ermittelt und mögliche Lösungen beschreibt. Gemäss dem Vorgehen des Standards NIST 800-30 wurden für die Wahrscheinlichkeiten einer Ausnutzung der Schwachstelle drei verschiedene Kategorien (High, Medium, Low) definiert und in der Tabelle 4.4 abgebildet [55]. Die gleichen Kategorien wurden ausserdem für den möglichen Schaden spezifiziert und in der Tabelle 4.5 dargestellt. Mithilfe dieser Kategorien erfolgte die Bewertung der Schwachstellen und anschliessend eine Einteilung in verschiedene Risikokategorien gemäss der Tabelle 3.1.

Tabelle 3.1: Zuweisung der Wahrscheinlichkeit einer Ausnutzung und dem Schaden zum Risiko

	Schaden		
Ausnutzung	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

3.3.3 Leistungsanalyse

Um zu evaluieren, ob die Verschlüsselung Einfluss auf die Performance hat, wurden auf dem Test-Catch-Server mit dem Tool »checkmk« Messungen durchgeführt. Primär war die durchschnittliche Auslastung des Prozessors von Interesse. Weil es sich bei der Software um einen noch nicht sehr optimierten Prototyp handelte, wurde auf tiefergehende Analysen verzichtet.

Für die Tests wurde derselbe Sender in unterschiedlicher Weise aufgezeichnet. Vom 3. bis 14. Mai wurde die Aufzeichnung mit dem bestehenden System und FFmpeg als Transcoder durchgeführt. Dabei wurde der Multicast-Stream 1:1 in MPEG-TS-Chunks geschrieben. Dann wurde zwischen dem 14. und 26. Mai FFmpeg so konfiguriert, dass die Streams mit AES-128 verschlüsselt wurden. Dabei wurden jeweils die ganzen Chunks verschlüsselt und nicht einzelne Samples. Vom 26. bis 31. Mai wurde nicht aufgezeichnet und vom 31. Mai bis 2. Juni wurden nur die ersten Video- und Audiospuren des Streams aufgezeichnet. Sie wurden aber zusätzlich zu AVC und AAC transcodiert und ebenfalls verschlüsselt. Vom 2. bis 6. Juni wurde dann mit dem Shaka Packager von Google [56] experimentiert, welcher anstelle von MPEG-TS-Dateien FMP4-Dateien für die erste Videospur erzeugte. Shaka Packager nahm aber keine Transcodierung oder Verschlüsselung vor.

4 Resultate

4.1 Bisheriger Zustand

4.1.1 Clients

Beim untersuchten TV-Angebot der Init7 sind zurzeit zwei firmenintern entwickelte Client-Apps im Einsatz. Diese nutzen unter Android die ExoPlayer-Library von Google [57] und unter Apple tvOS das VLCKit [58]. Zudem wird eine Liste mit Multicast-Adressen veröffentlicht, welche Kunden in einem beliebigen Player nutzen können. Die URLs für die HLS-Playlisten sind nicht öffentlich.

4.1.2 Backend

Das TV-Backend besteht einerseits aus einkommenden MPEG-TS Multicast-Streams und andererseits aus einer eigenentwickelten Aufzeichnungs-Infrastruktur mit mehreren Komponenten. Auf die Multicast-Streams wird in dieser Arbeit nicht weiter eingegangen. Wie in Abbildung 4.1 zu sehen ist, gibt es hauptsächlich drei Serverrollen. Diese sind TV API Server, Catch Server und Cache Server. Vom TV API Server gibt es nur eine zentrale Instanz, welche für alle Kanäle genutzt wird. Von den beiden anderen Funktionen werden jeweils mehrere Instanzen eingesetzt, welche nur je einen Teil aller Sender bearbeiten.

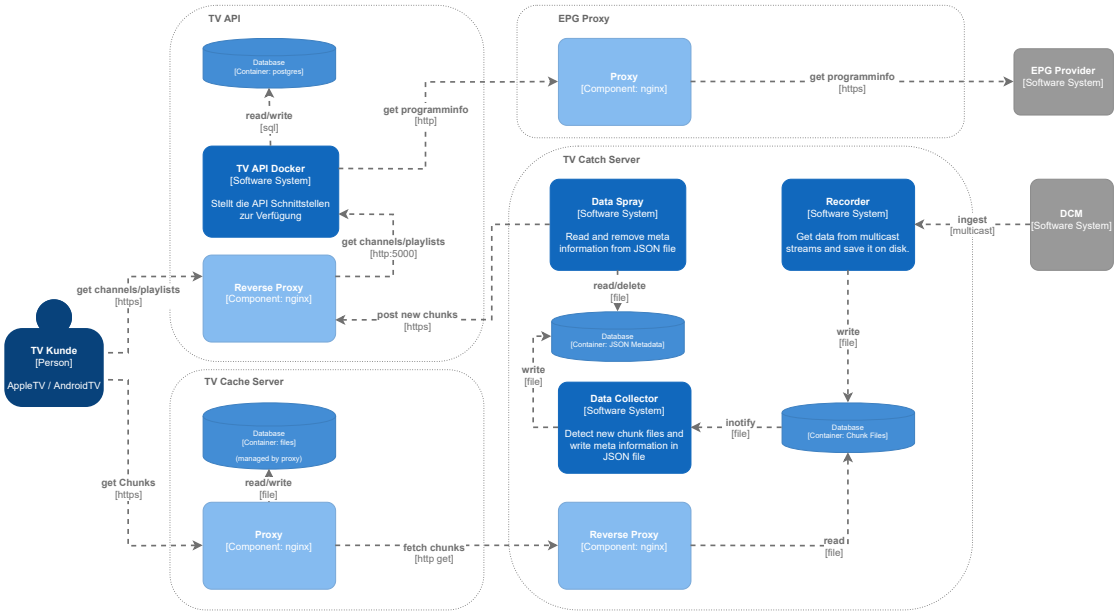


Abbildung 4.1: Bestehende Backend-Architektur der Init7 (Siehe 7.2.1 für grössere Version)

Catch Server Von den Catch Servern sind mehrere Instanzen vorhanden und ihre Funktion ist das Aufzeichnen der Streams. Die Multicast-Streams werden hier in sogenannte Chunks, also einige Sekunden lange Dateien, gespeichert. Ebenfalls werden die Metadaten dieser Chunks ausgelesen und an die TV API gesendet. Ein Catch Server zeichnet immer eine vordefinierte Menge an Streams bzw. TV-Kanälen auf. Ein Kanal kann dabei nicht zeitgleich auf mehreren Servern aufgezeichnet werden.

TV API Die TV API hat mehrere Funktionen. Zum einen liefert sie den Apps die Kanalliste und Programminformationen, zum anderen speichert sie die Informationen zu den Chunks ab und generiert daraus HLS-Playlisten. Von der TV API kann jeweils nur eine Instanz aktiv eingesetzt werden, sie ist also nicht horizontal skalierbar.

Cache Server Die Cache Server sind nur Reverse-Proxies mit Zwischenspeicher. Sie laden bei Bedarf die Chunk-Dateien von den Catch Servern und stellen diese den Kunden zur Verfügung. Die Cache Server sind dabei universell, das heisst, sie können die Chunks von allen Kanälen zwischenspeichern. Ihr eigentlicher Einsatzzweck ist die Effizienzsteigerung. Dazu werden wenn möglich jeweils alle Anfragen für einen Kanal auf dieselben Server geleitet. Die Caches sind georedundant aufgebaut, um die Verfügbarkeit zu erhöhen.

Die Init7 betreibt in Winterthur sechs Catch Server, sowie einen Cache Server. Zwei weitere Cache Server stehen in Zürich und einer in Genf. Dazu kommt die TV API und der EPG-Proxy, welche als virtuelle Maschinen in Winterthur aufgesetzt sind. Auf den Cache Servern ist Nftables aktiviert und konfiguriert. Als Serversoftware wird nginx eingesetzt. Die Caches sind jeweils direkt mit 10GBit/s am Core-Backbone Router angeschlossen. Sie greifen ohne Transport Layer Security (TLS) auf die Catch Server zu.

4.2 Requirements

Functionality (Funktionalität)

- Die Streaminginhalte müssen verschlüsselt sein.
- Der Schlüsselaustausch muss auf einem sicheren Kommunikationsweg erfolgen.
- Es wird ein Identity und Access Management verwendet, was nicht Teil der Arbeit ist. Aktuell wird das IP basiert verwaltet. Dies kann durch ein externer Microservice in Zukunft einfach ersetzt werden.
- Das System ist mit einem Metrics Endpoint erweiterbar. Dies ist nicht Teil dieser Arbeit und muss vom Provider umgesetzt werden.

Usability (Bedienbarkeit)

- Die App ist nach dem Prinzip «Install and Play» funktionsfähig.
- Es sind keine zusätzlichen Plugins oder Installationen notwendig.

Reliability (Zuverlässigkeit)

- Das Key Management ist monitorbar und kann sich selber wieder aufbauen (self-healing).

Performance (Effizienz)

- Da es sich um einen Prototyp handelt, wird die Performance nicht beachtet.

Supportability (Änder-/Wartbarkeit)

- Der Aufbau ist nach 12 Factor realisiert.
- Das Deployment ist mit Docker umgesetzt.
- Es ist modular aufgebaut, sodass die Umgebung mit weiteren Protokollen erweitert werden kann (Multi-Plattformen).
- Mit dem Django Framework ist die Umsetzung nach Init7 Standard realisiert.

Security Requirements

- Das Live- und Replay-TV sind verschlüsselt.
- Der aktuelle Stream darf nicht mit einem alten Schlüssel entschlüsselt werden können.
- Der Schlüssel wird periodisch in einem anpassbaren Intervall ausgetauscht.
- Der Schlüssel ist nur für Kunden des Providers abrufbar.
- Jeder Stream hat einen eigenen Schlüssel und kann nicht mit dem Schlüssel eines anderen Streams entschlüsselt werden.
- Der Schlüssel darf nur über die offiziellen Apps des Providers abrufbar sein. Der Client muss dafür authentifiziert werden. Dies wird für das DRM-Enforcement benötigt.

4.3 Architektur

Durch die Analyse der bestehenden Architektur wurde entschieden, diese durch einen Key Generator pro Sender, mehrere Key Server mit einer zentralen Datenbank, einen Key Retrieval Server und einen Authentifizierungsserver zu ergänzen. Der Architekturentwurf und die Verbindungen der einzelnen Komponenten werden in Abbildung 4.2 dargestellt. Die bereits bestehenden Services und Server sind blau markiert, die neuen Komponenten rot. Weiter gibt es eine neue Verbindung zwischen dem Key Generator und dem Recorder. Ebenfalls ist die Verbindung zwischen dem Data Collector und dem Recorder angepasst, diese beiden sind ebenfalls rot in dem Diagramm eingezeichnet. Des Weiteren muss der bestehende Data Collector angepasst werden, was jedoch nicht Teil dieser Arbeit ist.

Alle Services und Server sollen skalierbar und modular sein. So können in Zukunft auch andere DRM-Systeme oder Verschlüsselungsarten einfacher in die Lösung integriert werden. Diese Erweiterungen sind jedoch nicht Teil dieser Arbeit.

FairPlay Streaming Server Apple FairPlay Streaming wurde als DRM-Lösung gewählt, da HLS beim Provider bereits verwendet wird und eine Apple TV App existiert. Der Apple FairPlay Streaming Server ist für die zukünftige Erweiterbarkeit auf andere DRMs als eigener Microservice entwickelt. Er holt den Schlüssel beim Key Server ab und erzeugt dann den Entschlüsselungskontext. Durch den kann sich der Endkunde dann über seinen Apple TV die Sendung anschauen.

Key Generator Es gibt pro Kanal einen eigenen Key Generator, so kann flexibel auf die jeweiligen Vorgaben des Senders eingegangen werden. Der Key Generator erstellt jeweils die Schlüssel und sendet diese an die Key Server. Falls das Senden der Schlüssel gerade nicht möglich ist, müssen diese auf dem Key Generator zwischengespeichert werden. Somit wird sicher gestellt, dass keine Schlüssel verloren gehen. Des Weiteren stellt der Key Generator dem Recorder die Schlüsseldateien bereit.

Key Server Die Key Server speichern die vom Key Generator erhaltenen Schlüssel in einer zentralen PostgreSQL-Datenbank und stellen diese den DRM-Systemen zur Verfügung.

Data Collector, Recorder und Data Spray Mit den vom Key Generator erhaltenen Schlüsseln werden die aus dem Multicast Stream generierten Chunks verschlüsselt. Der Recorder informiert den Collector über neue Chunks, in dem er eine HLS-Playlist in eine Pipe schreibt. Der Collector bereitet diese Informationen dann für die TV API auf und schreibt sie in eine JSON-Datei. Der Data Spray sendet diese anschliesssend an die TV API. Sowohl der Data Collector wie auch der Recorder müssen von Init7 angepasst werden.

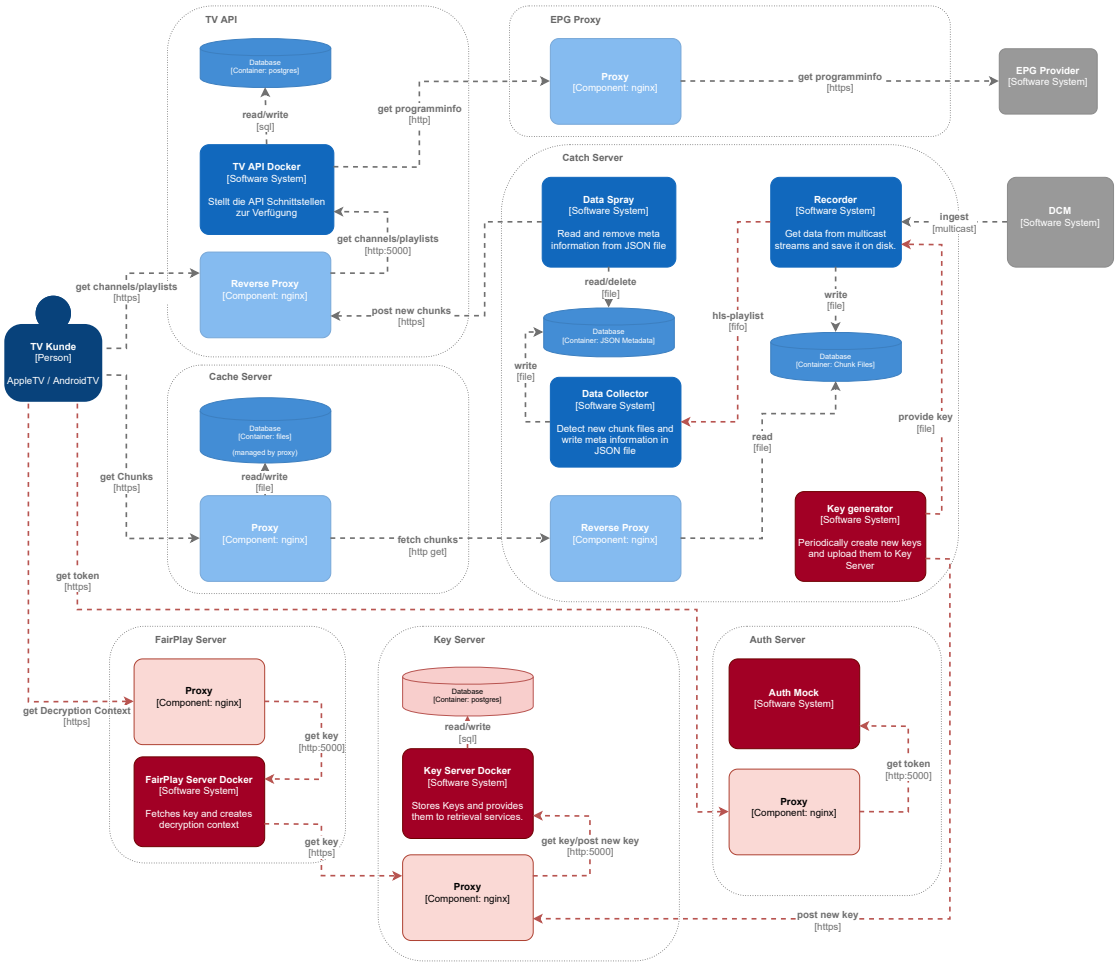


Abbildung 4.2: Zukünftige Backend-Architektur (Siehe 7.2.2 für grössere Version)

Authentifizierungsserver Der Authentifizierungsserver, im folgenden Auth Server genannt, stellt sicher, dass nur die Kunden des Anbieters über die entsprechende App fernsehen können. Für die Authentifizierung muss die Apple TV App beim Auth Server ein Token anfragen. Der Auth Server ist nicht im Umfang der Arbeit enthalten.

Im Sequenzdiagramm Abbildung 4.3 ist klar aufgezeichnet, wie die neu definierten Komponenten miteinander kommunizieren. Die App fragt die Playlist bei der TV API an und diese gibt die Playlist an die App zurück. Anschliessend holt die App beim Apple FairPlay Streaming Server den Entschlüsselungskontext. Falls dann kein gültiges Token existiert, wird beim Auth Server ein Token angefordert. So wird geprüft, ob es sich um einen Kunden handelt, der berechtigt ist, diesen Sender zu konsumieren. Sobald dieses verfügbar ist, kann der Entschlüsselungskontext angefragt werden. Der Apple FairPlay Streaming Server fragt anschliessend beim Key Server den Schlüssel an, erhält ihn und stellt den Schlüssel in einem Entschlüsselungskontext der App zur Verfügung. Dieser Ablauf wird jeweils beim Start einer Wiedergabe oder bei einem Wechsel des Schlüssels befolgt.

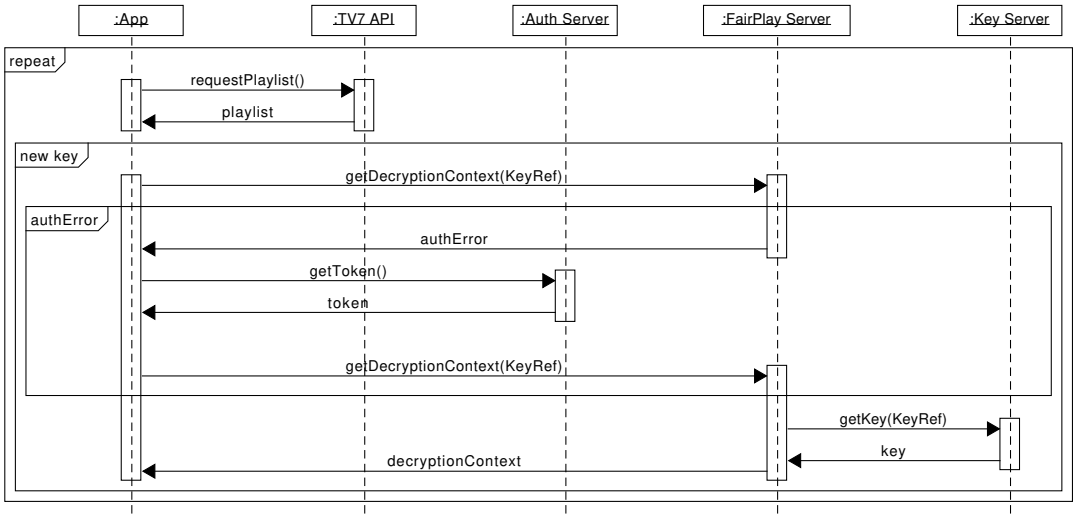


Abbildung 4.3: Sequenzdiagramm des FPS-Schlüsselaustschs

4.4 Bedrohungsanalyse (Threat Analysis)

In diesem Kapitel wird auf die wesentlichen Punkte aus der Bedrohungsanalyse eingegangen. Die gesamte Analyse befindet sich im Anhang. Als Unternehmensziel ist folgendes definiert: «Die TV-Plattform erlaubt es den Kunden Live und Replay Fernsehsendungen im HD-Format anzuschauen». Auf das System haben die Kunden des Anbieters Zugriff. Dabei wird beim Anbieter zwischen Privatkunden und Businesskunden unterschieden. Die Administratoren, welche beim Anbieter angestellt sind, haben ebenfalls Zugriff auf das System. Die externen Abhängigkeiten des Systems sind, dass das TV-Signal von UPC und GibSoloutions empfangen wird und die Programminformationen von EPG.Best abgerufen werden. Als Internetanbieter fungiert Init7.

4.4.1 Analyse und Zersetzung des Systems

4.4.1.1 Bestehende Infrastruktur

Das bestehende System kann Fernsehinhalte nicht verschlüsseln. Dessen Netzwerkaufbau ist in Abbildung 4.4 ersichtlich. Die Cache und Catch Server stehen direkt im Internet und werden jeweils mit einer Nftables Firewall geschützt. Es existieren jeweils mehrere Catch und Cache Server. Die TV API und der EPG Proxy Server werden durch eine virtuelle Firewall mit dem Internet verbunden. Die Benutzer greifen über Hypertext Transfer Protocol Secure (HTTPS) auf die TV API und die Cache Server zu.

Das Data Flow Diagram (DFD) zeigt auf, dass es in dieser Konfiguration dem TV-Konsumenten mit jedem Gerät und jeder Software möglich ist, den TV-Stream zu schauen. Wie in Abbildung 4.5 ersichtlich ist, greift er dazu auf die TV API und die Cache Server zu. Diese werden mit Informationen von den Catch Servern und dem EPG Proxy versorgt. Die TV-Inhalte werden von den Multicast Streams der DCM Server geholt. Die Administratoren haben auf alle Services und Server Zugriff. Mit den roten gestrichelten Linien wird der vertrauenswürdige Bereich markiert. Dieser ist so gewählt, da diese Bereiche komplett vom TV-Anbieter verwaltet werden. Sowohl die Server, das Netzwerk und die Software werden vom Anbieter zur Verfügung gestellt.

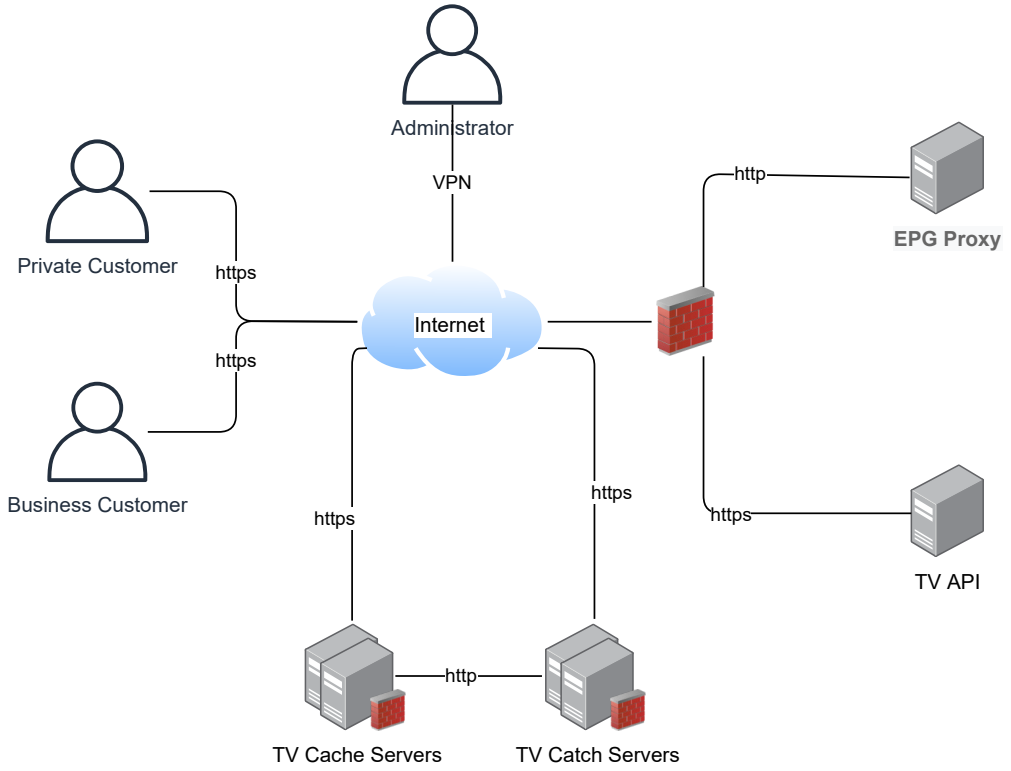


Abbildung 4.4: Aktuelles Netzwerkdiagramm von Init7 (Siehe 7.2.3 für grössere Version)

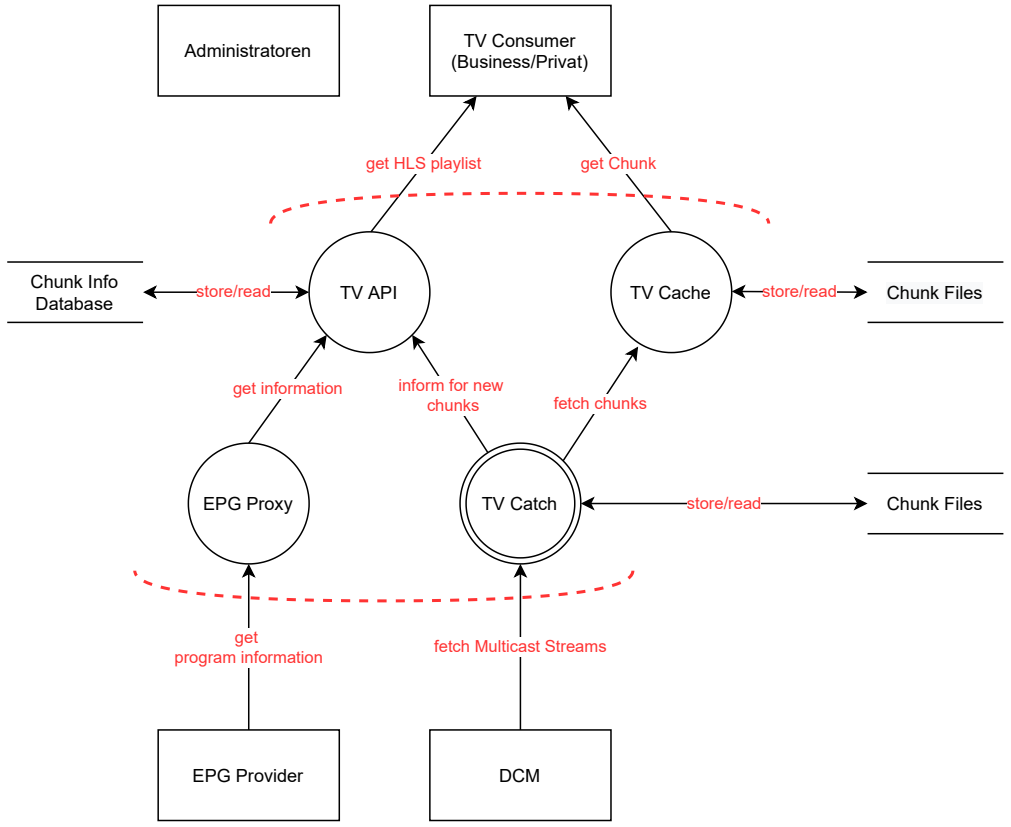


Abbildung 4.5: Aktuelles Data Flow Diagram (DFD) (Siehe 7.2.5 für grössere Version)

4.4.1.2 Soll-Zustand

Beim Soll-Zustand sind die zusätzlichen Server eingebunden, die in Abschnitt 4.3 definiert sind. Ebenfalls sind die Anpassungen für die verbesserte Sicherheit des bestehenden Zustandes integriert. Zu den wichtigsten Änderungen gehört die Umstellung des gesamten Netzwerkverkehrs auf HTTPS. Ausserdem ist es nicht notwendig, dass die Catch Server aus dem Internet erreichbar sind. Diese werden deshalb zusammen mit dem Key Management Server in ein separiertes Netzwerk genommen, welches den Zugriff von aussen nicht zulässt. Wie in Abbildung 4.6 ersichtlich ist, haben zusätzlich der Apple FairPlay Streaming Server und auch die Cache Server Zugriff in dieses abgeschottete Netzwerk. Diese Server sind auf die Kommunikation mit den Catch Server oder dem Key Management Server angewiesen. Eine weitere Änderung ist, dass alle Server mit einer Nftables Firewall abgesichert werden sollen, was die Sicherheit zusätzlich erhöht.

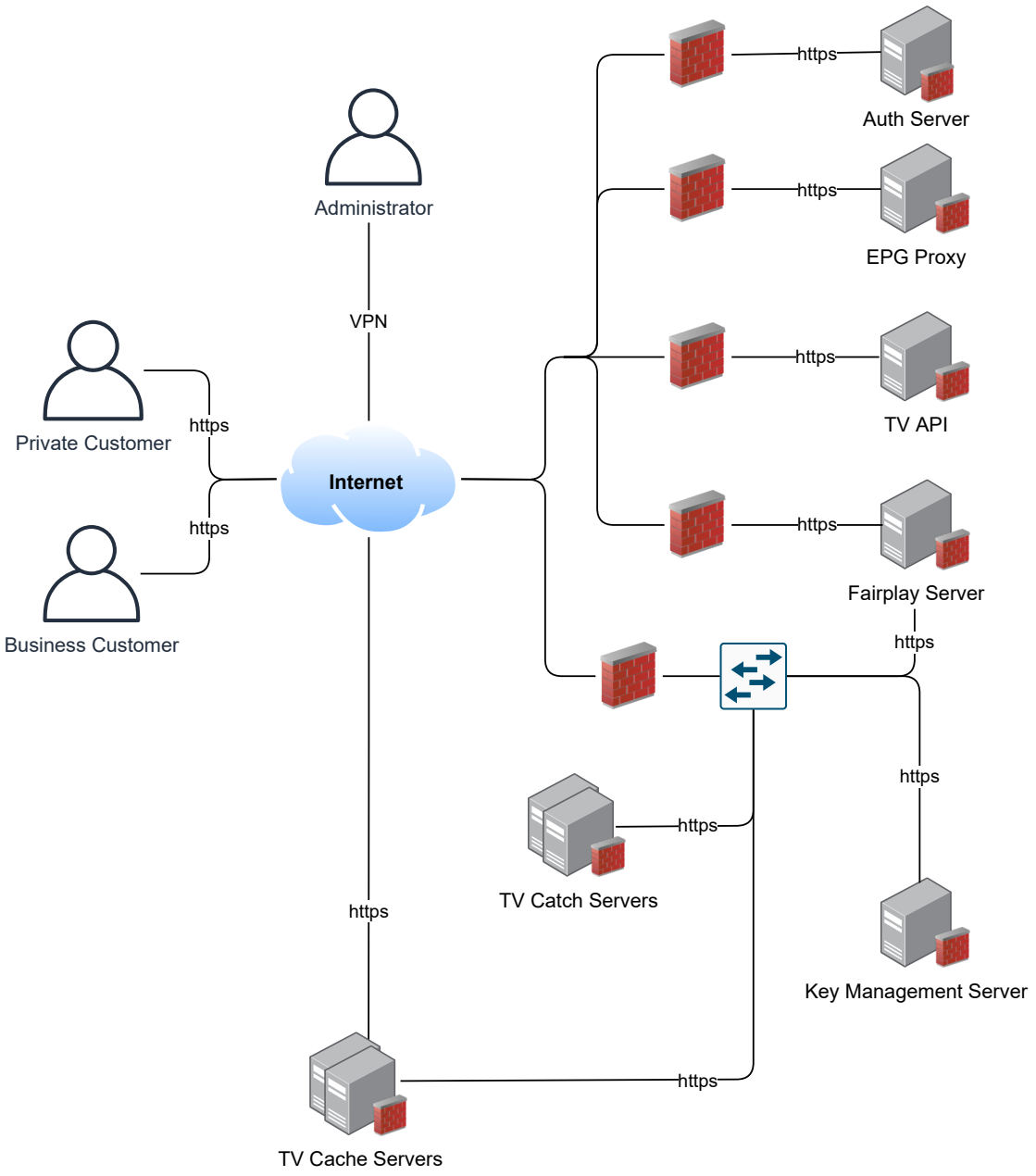


Abbildung 4.6: Entworfenes zukünftiges Netzwerkdiagramm (Siehe 7.2.4 für grössere Version)

In das DFD der neuen Lösung sind ebenfalls die zusätzlichen Server und Services integriert. Dazu gehören der Auth Server, welcher die Authentifizierungstokens ausstellt und die gesamte Key Management Infrastruktur. Eine Neuerung ist ebenfalls, dass der Stream nur noch mit der App vom TV-Anbieter angeschaut werden können. Mit der pink gestrichelten Linie haben wir einen tiefer liegenden Vertrauensbereich geschaffen. Diese Datenflüsse finden im abgeschotteten Netzwerk statt und haben ausschliesslich mit dem Schlüsselaustausch zu tun.

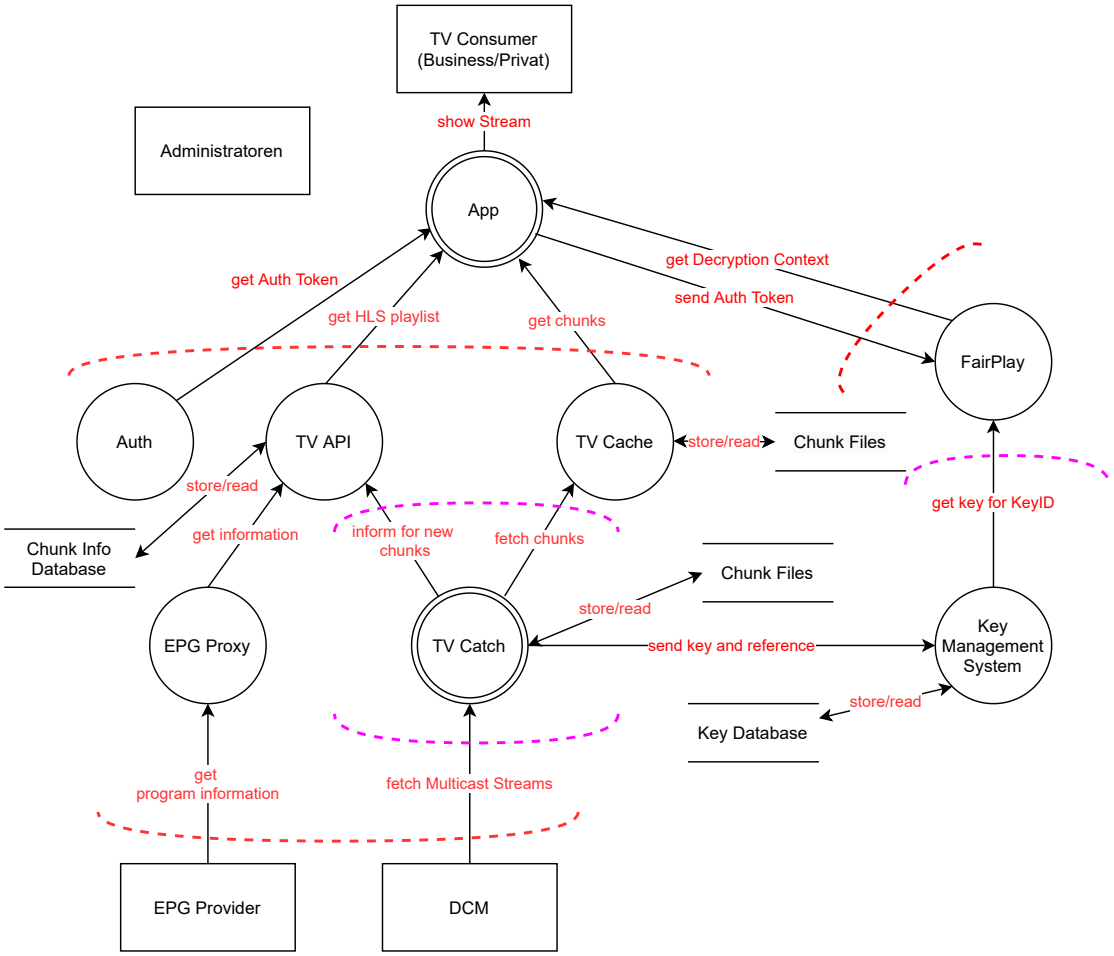


Abbildung 4.7: Aktuelles Data Flow Diagram (DFD) (Siehe 7.2.6 für grössere Version)

4.4.2 Bedrohungsakteure (Threat Agents)

Die Analyse der Bedrohungsakteure hat folgende offengelegt:

Script Kiddies Sie wollen prüfen, ob sie in der Lage sind, das System zu hacken. Ihr einziges Ziel ist Spass.

Mitarbeiter Ehemalige Mitarbeiter möchten sich rächen oder das System sabotieren, da sie ihren Job verloren haben. Ein weiterer Grund könnten finanzielle Absichten der Mitarbeiter sein.

Cyber Criminal Sie wollen Malware ins System einschleusen und an andere Benutzer verbreiten. Ihr Ziel ist der finanzielle Gewinn.

Konkurrenz Sie wollen das System sabotieren, dem Anbieterimage schaden und dadurch Kunden zum Wechsel des Anbieters bewegen. Ihr Ziel sind die Verursachung eines Image-Schadens und der finanzielle Gewinn.

Für einen Anbieter wie Init7, welcher mit einer freien Routerwahl und dem schnellsten Internet der Schweiz wirbt, sind Script Kiddies eine realistische Bedrohung. Bei solchen Anbietern ist die Anzahl der Kunden mit viel IT-Wissen höher und solche Kunden versuchen häufiger unerlaubt in ein System einzudringen. Deshalb darf dieser Faktor nicht unterschätzt werden. Unwahrscheinliche Bedrohungsakteure sind Staaten, da sie keine Motivation haben, TV-Dienste zu stören und Hacktivisten, da die politische Motivation fehlt.

4.4.3 Bedrohungen (Threats)

Die Bewertung hat 112 Bedrohungen aufgezeigt. Eine vollständige Übersicht von allen Bedrohungen ist im Anhang enthalten. Die Bedrohungen sind jeweils als realistisch oder unrealistisch eingestuft. Es fällt auf, dass bei einem Angriff wiederholt dieselben vier Vorteile entstehen. Einerseits sind die Angreifer in der Lage, das System mit gefälschten Streams zu befüllen. Durch diese können die Kunden beeinflusst werden. Andererseits könnten unbefugte Personen den Stream anschauen. Dadurch können dem Anbieter Einnahmen entgehen. Eine weitere bedeutende Bedrohung sind DoS-Attacken. Aufgrund von DoS-Attacken können einzelne Komponenten, beispielsweise ein Catch Server, ausfallen. Das führt zu einem Teilausfall der Sender und zu einer Lücke im Recording. Der vierte Nutzen für die Angreifer ist, dass sie sich Adminrechte auf die Systeme verschaffen können. Sei das beispielsweise über eine Brute-Force Attacke, einen Man-in-the-Middle-Angriff oder über einen gestohlenen Secure Shell (SSH) Zugriff.

4.4.4 Schadensbegrenzung (Mitigation)

Damit die relevanten Bedrohungen abgeschwächt werden können, sind Massnahmen definiert. Ein wichtiges Thema bei der Schadensbegrenzung sind Redundanzen. Durch redundante Anbieter und (Speicher-)Systeme können Ausfälle von Services und Server vermieden werden. Ein weiterer Punkt sind Administratoren und Administratoren-Accounts. Jeder Administrator soll nur den Zugriff haben, den er benötigt. Das verhindert, dass ein Administrator an einem System, auf welches er nicht zugreifen sollte, Änderungen vornehmen kann. Dabei spielt es keine Rolle, ob diese absichtlich oder versehentlich passieren. Weiter soll es keine geteilten Benutzer geben, wie beispielsweise den root-Account. So kann anhand der Logs jederzeit nachvollzogen werden, wer welche Anpassungen vorgenommen hat. Die Anzahl der Loginversuche soll ebenfalls eingeschränkt sein. Das hilft Brute-Force Attacken zu verhindern. Auch netzwerkseitig muss einiges beachtet werden. Es sollen zum Beispiel Firewalls mit White- und Blacklisting zum Einsatz kommen. Auf den Firewalls sollen nur die Ports und Verbindungen freigegeben sein, die verwendet werden, alles weitere soll geblockt sein. Ebenfalls soll der komplette Kommunikationsweg unter Kontrolle sein, das verhindert Netzwerk-Spoofing. Alle Massnahmen sind in der Tabelle 4.1 aufgelistet.

Tabelle 4.1: Massnahmen zur Abschwächung oder Erschwerung der Ausnutzung von Bedrohungen

Nr.	Beschreibung
R1	Der Key Generator muss die Schlüssel so lange behalten, bis sie erfolgreich an den Key Server gesendet wurden.
R2	Redundanz schaffen, indem mehrere Anbieter für den gleichen Dienst verwendet werden.
R3	Mehrere Instanzen des Systems für Hochverfügbarkeit/Redundanz aufschalten.
R4	Einsatz von selbstheilenden Systemen mit Orchestrierung. (Benötigt ausserdem den Einsatz von Health Checks)
R5	Verwendung von Monitoring und Alarmierung.
R6	Einsatz von Metric collection und Autoscaling.
R7	Einsatz von redundanten Speichersystemen. (RAID, Ceph, Object-Storage, Datenbank-Cluster)
R8	Für Algorithmen und bekannte Implementierung wird auf bewährte Bibliotheken zurückgegriffen. (Keine Fehler durch eigenen Code)
R9	Durchführen von Code- und Konfig-Reviews. (automatisiert und manuell)
R10	Kontrolle des kompletten Kommunikationswegs, um Netzwerk-Spoofing zu verhindern.
R11	Limitierung der Anzahl Loginversuche pro Zeiteinheit.
R12	Einsatz einer Firewall (White- und Blacklisting, Portsperrung) Nur Ports und Verbindungen freigeben, welche auch verwendet werden.
R13	Einsatz von http Public Key Pinning, um Man-in-the-Middle Attacken zu verhindern.
R14	Kommunikation findet nur per TLS statt.
R15	Aktivierung verstärkter Authentifizierung (Benutzername/Passwort, Kennwortrichtlinien, Token, 2FA)
R16	Sensibilisierung der Administratoren auf Phishing und Social Hacking.
R17	Administratoren gezielt auswählen. Soviel wie nötig, so wenig wie möglich.
R18	Abgeschottetes Netzwerk ohne Zugriff von aussen (nur per Jump Host)
R19	Externer Zugriff von Administratoren nur mit 2-Faktor Authentifizierung zulassen.
R20	Chunk Dateien müssen verschlüsselt werden.
R21	Zugriff auf die Ressourcen und Server nur im Init7 Kundennetzwerk zulassen.
R22	Gespeicherte Schlüssel werden nach Rotation gelöscht.
R23	Apps signieren.
R24	App auch auf Website anbieten.
R25	Apps nur aus offiziellem Store unterstützen.
R26	Apps regelmässig im Store überprüfen und Nachahmer den Store-Betreibern melden.
R27	Root-Rechte/Jailbreak auf den Endgeräten erkennen und Betrieb der App verweigern.
R28	Datenübertragung muss authentifiziert sein.
R29	Schlüssel werden periodisch rotiert.
R30	Quelladresse im Request Header mit Inhalt des JWT Tokens abgleichen.
R31	Schlüsselaustausch findet nur verschlüsselt statt.
R32	Es werden pro TV-Sender verschiedene Schlüssel verwendet.
R33	Schlüssel oder Token hat nur eine begrenzte Lebensdauer.
R34	Schlüssel verschlüsselt in der Datenbank speichern.
R35	Für die Dienste und die Datenzugriffe werden nur Serviceuser mit begrenzten Zugriffsrechten verwendet.
R36	Verhindern von SQL-Injection durch die Verwendung von Prepared Statements.
R37	Einsatz von DRM-Lösungen, welche für die jeweiligen Endgeräte vorgesehen sind.
R38	Einschränken der Anzahl gleichzeitige Sessions/Logins.
R39	Es gibt keine geteilten Administratoren Logins, sondern nur personalisierte Logins.
R40	Zentralisierte (Applikations-)Log-Collection, kein bearbeiten der Admins.
R41	Datenbankadministratoren dürfen keinen Admin-Zugriff auf dem System haben.
R42	Arbeit mit Versionsverwaltung (Bsp. Git) und Continous Deployment.
R43	Ablauf der Administratoren Kennwörter nach einer gewissen Dauer.

4.5 Implementation

Als DRM-Lösung wird ein FPS eingesetzt und client-seitig wird Apple TV unterstützt. Die Umsetzung der einzelnen Systeme ist in den nachfolgenden Abschnitten beschrieben. Sofern anwendbar, wurde das Django Framework eingesetzt. Das Deployment ist gemäss Init7 Standard mit Docker und Docker Compose realisiert. Alle Systeme authentifizieren sich untereinander über in Django integrierte Token und Benutzer. Die einzige Ausnahme bildet die Kommunikation zwischen FFmpeg und dem Key Generator, welche auf Dateien basiert.

4.5.1 Key Server

Der Key Server speichert die Schlüsselinformationen und stellt sie für die verschlüsselte TV-Wiedergabe zur Verfügung. Er ist mit Django und dem Django REST Framework (DRF) umgesetzt. Es ist möglich, mehrere Instanzen des Key Servers laufen zu lassen. Die Schlüsselinformationen werden vom Key Generator an den Key Server übermittelt und in einer zentralen PostgreSQL Datenbank gespeichert. Es werden die Attribute Key Referenz (key_ref), der Schlüssel selber (key), der Initialisierungsvektor (init_vector) und das Ablaufdatum (expire_date) in die Datenbank geschrieben. Es ist eine geplante Aufgabe implementiert, die jede Minute alle abgelaufenen Schlüssel löscht. Das Intervall für die Löschung ist anpassbar, so kann mit wenig Aufwand auf die jeweiligen Anforderungen eingegangen werden. Die Aufgabe ist als Celery Task umgesetzt. Celery besteht aus dem celerybeat, das ist der periodische Task Scheduler und dem celeryworker, welcher die Aufgaben dann ausführt. Die vom celerybeat erstellten Aufgaben werden in eine Redis Jobqueue gespeichert. Da Redis-Datenbanken die Daten nicht auf der Festplatte, sondern im Arbeitsspeicher ablegen, ist das Speichern und der Zugriff sehr schnell. Da das Ablaufdatum vom Key Generator mitgegeben wird, können verschiedene Server auch unterschiedliche Ablaufdaten haben, ohne dass am Key Server etwas angepasst werden muss. Die Schlüsselinformationen können mit Angabe der Key Referenz über einen GET Request via HTTPS abgerufen werden. Die Kommunikation zwischen dem Key Generator und dem Key Server beziehungsweise zwischen dem Apple FairPlay Streaming Server und dem Key Server erfolgt über HTTPS. Ausserdem kommunizieren die beiden Server nicht direkt mit dem Key Server, sondern über einen Ngnix Reverse Proxy, welcher TLS terminiert. Da der ganze Netzwerkverkehr via Proxy geht, erschwert das Angreifern unter anderem DoS-Angriffe und erhöht generell die Sicherheit.

4.5.2 Key Generator

Der Key Generator ist für die Erstellung der Schlüssel zuständig. Pro Sender existiert ein eigener Key Generator mit jeweils eigenen Queue- und Schlüsseldateien. Die Schlüssel werden von geplanten Tasks generiert. Sowohl der Schlüssel als auch der Initialisierungsvektor müssen zufällig erstellt werden. Ein Zufallszahlengenerator ist für sichere kryptografische Verfahren Voraussetzung. Diese Aufgabe wird durch das Python Modul «secrets» realisiert [59, 60]. Für die Generierung des Schlüssels wird die secrets-Funktion «token_bytes» genutzt. Diese gibt einen zufälligen Byte-String zurück. Der Initialisierungsvektor wird mit der Funktion «token_hex» erstellt. Es wird eine zufällige Hex-Zeichenkette zurückgegeben. Die erzeugten Schlüsselinformationen werden speziell für FFmpeg abgespeichert [61] und per HTTPS Request an den Key Server gesendet. Im Request Body stehen die Key Referenz, der Schlüssel im Hex-Format, der Initialisierungsvektor im Hex-Format und das Ablaufdatum. Wenn der Schlüssel vom Key Server angenommen wird, also dieser ein HTTP-Statuscode 200 zurückmeldet, werden die Schlüsselinformationen nicht abgespeichert. Sollte der Key Server nicht erreichbar sein oder aus anderen Gründen die Schlüssel nicht annehmen, werden die Schlüsselinformationen vom Key Generator in eine Queue Datei geschrieben. Die Planung des Tasks kann in der .env Datei angepasst werden. Ein zweiter Task, welcher unabhängig läuft, schickt die Elemente der Queue regelmässig an den Key Server. Sofern die Elemente der Queue angenommen werden, entfernt das Programm sie aus der Queue. Der ganze Ablauf ist im Sequenzdiagramm in Abbildung 4.8 abgebildet. Dadurch, dass die Schlüssel entweder direkt gesendet oder in eine Datei gespeichert werden, ist der Key Generator zustandslos und er kann sich selbst wieder aufbauen. Jedes Mal, wenn ein neuer Schlüssel erstellt wird, müssen aufgrund von FFmpeg die Key Informationen in einem bestimmten Format abgelegt werden. Es werden zwei Dateien erzeugt. Die Key Datei

beinhaltet den Schlüssel im Binary-String Format. Die Key Info Datei enthält die Key URI, also die URL des FairPlay Servers inklusive der Key Referenz, den Initialisierungsvektor und und den Pfad zur Key Datei. Die Key Referenz ist eine Universally Unique Identifier (UUID) und wird mit UUID4 erstellt. UUID4 bedeutet, dass die UUID zufällig generiert wird. Die Wahrscheinlichkeit für ein Duplikat ist gemäss Onlinerecherchen [62] bei dieser Version eins aus einer Milliarde bei 103 Billionen Version-4 UUIDs. Des Weiteren ist im Key Generator ein Logger implementiert. Dieser protokolliert die Änderungen am System, beispielsweise wenn ein neuer Schlüssel generiert wird, ob der Schlüssel dem Key Server erfolgreich übermittelt worden ist oder nicht.

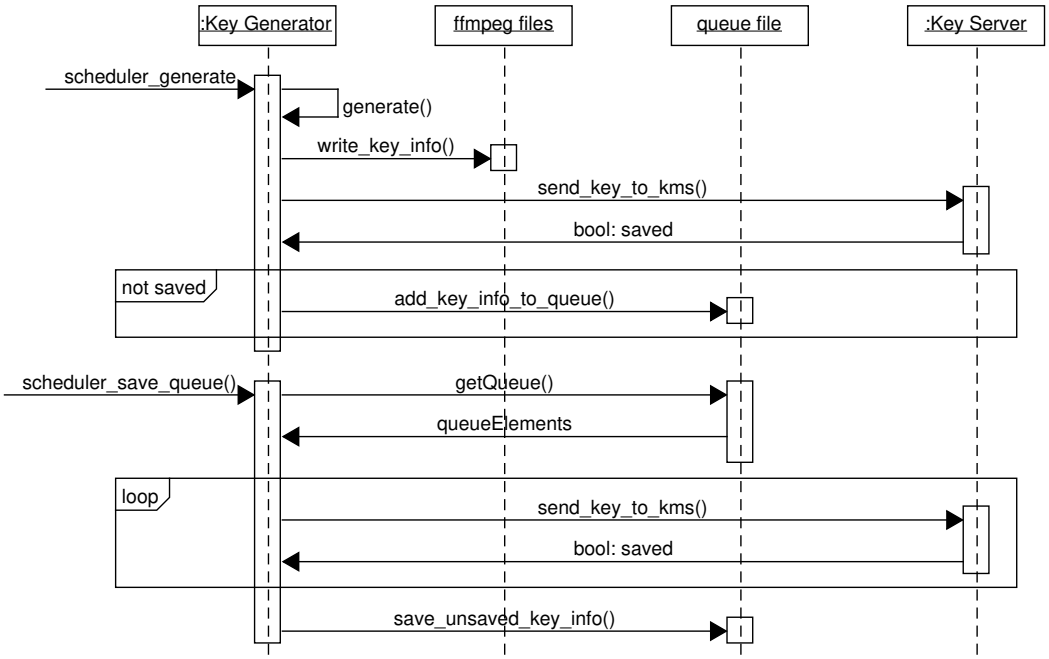


Abbildung 4.8: Sequenzdiagramm Key Generator

4.5.3 FairPlay Streaming Server

Der Apple FairPlay Streaming Server ist wie der Key Server ebenfalls mit Django und dem Django REST Framework aufgebaut. Die App kann beim Apple FairPlay Streaming den Decryption Context abholen. Das Key Security Module erhält dazu vom App den Server Playback Context (SPC) und generiert dann mit diesem und der key_ref das Content Key Context (CKC). Anschliessend wird das CKC der App zur Verfügung gestellt. Das KSM ist als separate Python C Erweiterung gebaut [63]. Dies ist eine «shared dynamically loaded library» (deutsch: gemeinsame, dynamisch geladene Bibliothek), welche die KSM Referenzimplementierung von Apple verwendet. Die Hauptfunktion ist die Generierung des CKC anhand des SPC und der key_ref. Neben dem SPC und der key_ref müssen bei deren Funktionsaufruf noch der Application Secret Key (ASK), der Private Key vom FPS Zertifikat und eine Callbackfunktion mitgegeben werden. Die Callbackfunktion wird dafür benötigt, den Schlüssel und Initialisierungsvektor vom Key Server zu laden. Der ASK und der Private Key werden beim ersten Aufruf der Funktion im Arbeitsspeicher abgelegt. Wenn diese geändert werden müssen, muss der Python Interpreter neu gestartet werden. Aktuell wird der Schlüssel beim Key Server mit einem hinterlegten Django Auth-Token abgerufen. Die Kommunikation von der App zum FPS Server findet nicht direkt statt, sondern über einen Ngnix Reverse Proxy, welcher TLS terminiert.

4.5.4 Demo App

Die Demo App ist sehr minimalistisch aufgebaut. Sie dient lediglich dazu, die Funktionen der anderen Komponenten aufzuzeigen. Die Implementation basiert auf der Klasse AVAssetResourceLoader [64]. Die App enthält einen Player, der nach Eingabe einer URL HLS-Streams, zu sehen in Abbildung 4.9, abspielen kann. Weiter hat die App eine Key Manager Klasse. Um den Stream abzuspielen, schickt die App dem Key Server den Server Playback Context (SPC) und die Key Referenz. Die Key Referenz wird separat mitgeschickt, da es sich um einen Prototyp handelt, der die Referenzimplementierung des Key Security Module nutzt und diese somit vom KSM benötigt wird. Daraufhin bekommt die Demo App vom Key Server den Content Key Context (CKC) zur Verfügung gestellt. Der Code der App ist entsprechend vorbereitet, dass sie vom Authentifizierungsserver das Authentifizierungstoken abholen und dieses verwalten könnte. Da der Auth Server nicht implementiert ist, kann diese Funktion nicht getestet werden.

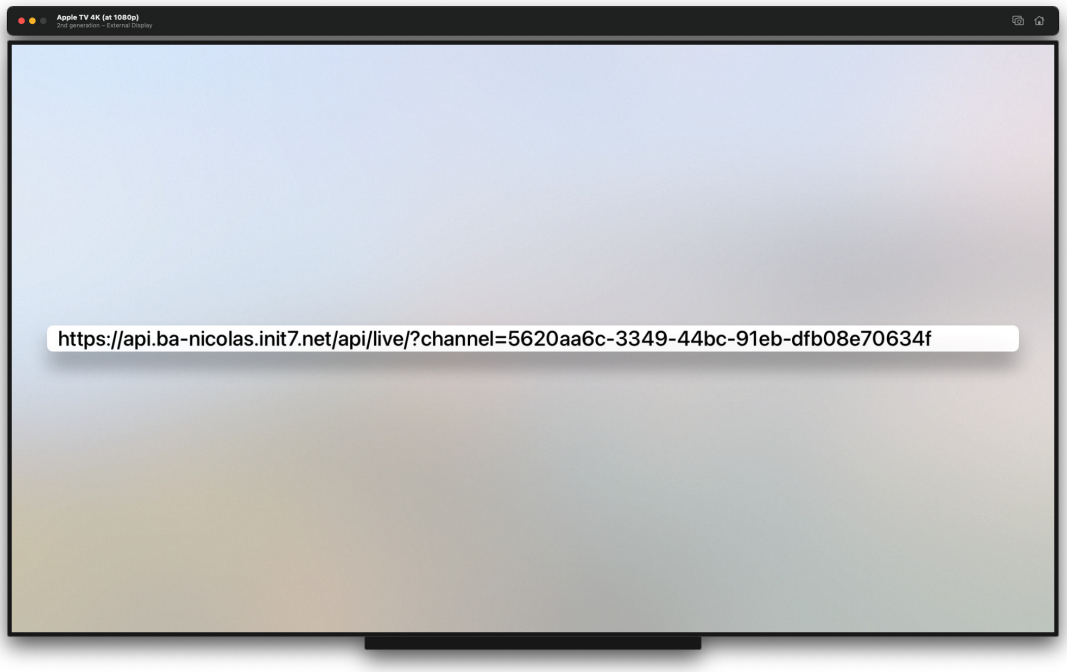


Abbildung 4.9: URL-Eingabefeld der Demo App.

4.5.5 Anpassungen bestehendes System

Für einen funktionsfähigen Prototypen müssen auch Anpassungen am bestehenden System gemacht werden. Die TV API ist so angepasst, dass sie die Key Referenzen in die Chunk-Tabelle speichern und dann ausgeben kann. Parallel zur Arbeit hat Init7 in einem Proof of Concept (PoC) einen neuen TV Packagers erstellt, der neu Recorder und Data Collector in einem ist. Diese Anpassung ist bei der Programmierung des Prototyps bereits verwendet worden. Der Packager nutzt FFmpeg um die Streams aufzuzeichnen und zu verschlüsseln. Ausserdem teilt er diesem mit, wo sich die Schlüsseldateien befinden. Er erhält von FFmpeg dann jeweils pro Segment eine Referenz auf den verwendeten Schlüssel und gibt diese neu an den Data Spray weiter. FFmpeg wird beim Anbieter bereits eingesetzt und verwendet für die Verschlüsselung AES-128. Bei der Umsetzung des Prototyps stellt sich heraus, dass Apple FairPlay Streaming nur Sample-AES unterstützt und FFmpeg nur AES-128 aber kein Sample-AES kann. Als Alternative zu FFmpeg bietet sich der Google Shaka Packager an. Der Umbau der Infrastruktur auf die Verwendung von Shaka Packager ist aufwändig und nicht als Teil der Arbeit definiert. Ausser der Wiedergabe des verschlüsselten Streams funktionieren alle neuen System des Prototyps wie erwartet. Dies ist anhand des Datenstreams und des Schlüsselaustausches getestet.

4.6 Evaluation

4.6.1 Key Rotation

Die Auswertung der Simulation zeigt, dass es trotz des maximalen Zuwachs von drei Kunden pro Minute bis zu 6.5 * 10^4 Anfragen pro Stunde geben kann. Ausserdem ist in der Abbildung 4.10 ersichtlich, dass die entstehende Kurve umgekehrt proportional ist. Ab 20 Key-Rotationen pro Minute flacht die Kurve stark ab.

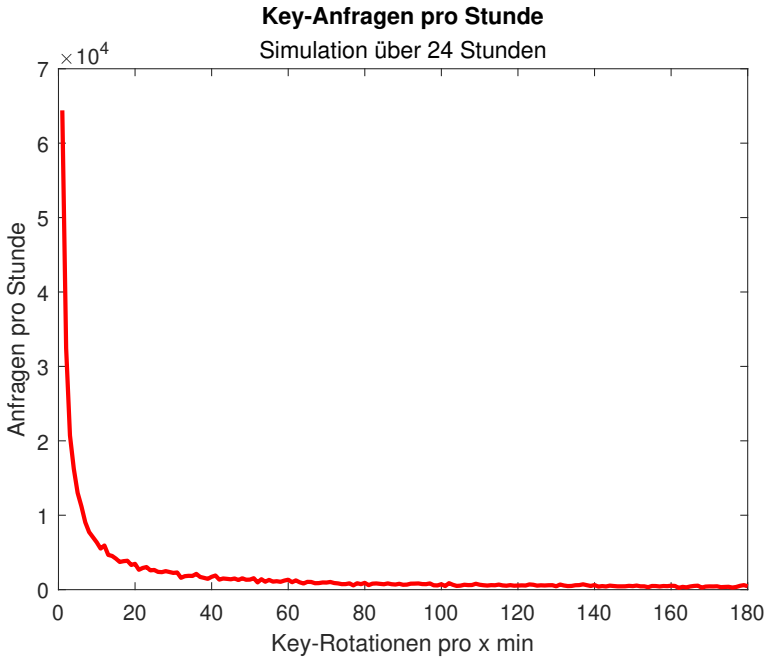


Abbildung 4.10: Simultation Key Rotation

4.6.2 Anforderungen

Der erste Schritt der Evaluierung ist die Überprüfung der Security Requirements. In der Tabelle 7.3 im Anhang sind die Security Requirements aufgelistet. Ebenfalls ist das Ergebnis der Analyse enthalten, welche zeigt, ob und wie die Anforderungen umgesetzt sind. Von 43 Anforderungen sind im Projekt 16 vollumfänglich implementiert. Die angewandten Anforderungen sind mit einer Begründung in der Tabelle 4.2 ersichtlich. Weitere vier Anforderungen sind teilweise implementiert und in der Tabelle 4.3 aufgelistet. Die restlichen Spezifikationen sind bewusst nicht umgesetzt.

Tabelle 4.2: Umgesetzte Sicherheitsanforderungen mit Begründung

Nr.	Weshalb
R1	Der Key Generator schreibt die Schlüssel auf das Dateisystem, wenn die Übertragung zum Key Server nicht erfolgreich ist. Diese werden später nochmals übertragen.
R8	Code wurde bei der Implementierung von Algorithmen immer die Referenzimplementation verwendet.
R10	Das System läuft in einem Netzwerk, welches unter eigener Kontrolle befindet.
R14	Kommunikation wurde mit HTTPS umgesetzt. Es werden gültige Zertifikate verwendet.
R20	Dies war das Hauptziel der Arbeit.
R21	Der Zugriff auf die Ressourcen ist über IP Filterung geregelt.
R22	Das Verfallsdatum kann vom Systembetreiber gewählt werden. Die Schlüssel werden gelöscht, sobald sie nicht mehr gebraucht werden.
R23	Apps sind durch Apple signiert.
R25	Nur offizielle Apps mit dem richtigen Zertifikat können auf die Streams zugreifen.
R28	Für die Kommunikation zwischen den Microservices werden Tokens verwendet.
R29	Key Generator erstellt periodisch neue Schlüssel. Der Ablauf wird durch den Key Server geregelt.
R31	Der Schlüsselaustausch findet über CEK Nachrichten statt. Diese sind durch FairPlay Streaming verschlüsselt.
R32	Pro Sender wird ein separater Key Generator eingesetzt.
R35	Pro Dienst gibt es einen Benutzer mit eingeschränkten Rechten. Dieser kann mit einem Token authentifiziert werden.
R36	Für die Implementation der Datenbank wurde das Django Rest Framework verwendet.
R37	Es wird FairPlay Streaming für Apple Geräte eingesetzt.

Tabelle 4.3: Teilweise umgesetzte Sicherheitsanforderungen mit Begründung

Nr.	Weshalb
R3	Im Code wurde darauf geachtet, dass die Systeme mit mehreren Instanzen redundant laufen können. Wurde aber aufgrund des Prototyps nicht umgesetzt.
R9	Es wurden manuelle Reviews durchgeführt. Auf automatisierte Reviews wurde verzichtet.
R33	Token Lebensdauer wurde noch nicht festgelegt
R42	Das System wurde mit Github entwickelt. Jedoch ist das Deployment auf die Testinfrastruktur manuell.

4.6.3 Risikoanalyse

Für die Anwendung von NIST-800-30 sind die Wahrscheinlichkeiten für die Ausnutzung einer Schwachstelle kategorisiert und in der Tabelle 4.4 festgehalten. «High» ist die höchste Stufe. Diese Stufe ist den Schwachstellen zugewiesen, welche nicht sehr gut abgesichert sind oder einen sehr hohes Interesse bei Angreifern auslösen. Die nächste Stufe in der Tabelle ist «Medium». Die Schwachstellen mit einer unvollständigen Absicherung und mit einem mittleren Wert haben diese Wahrscheinlichkeit erhalten. Die tiefste Stufe ist «Low» und wird für die Klassifizierung der Schwachstellen mit geringem Nutzen oder auch einer guten Absicherung eingesetzt.

Tabelle 4.4: Einstufungen für die Wahrscheinlichkeit einer Ausnutzungen der Schwachstelle.

Wahrscheinlichkeit	Beschreibung
High	- Schwachstelle bietet sehr hohen Nutzen. - Angreifer sind sehr motiviert und haben Ressourcen. - Schwachstelle nicht gut abgesichert.
Medium	- Schwachstelle bietet mittleren Nutzen. - Schwachstelle ist teilweise abgesichert.
Low	- Schwachstelle bietet keinen oder nur geringen Nutzen. - Schwachstelle ist gut abgesichert.

Die Auswirkungen auf das System sind in der Tabelle 4.5 aufgelistet. Die Stufe «High» sagt hier aus, dass der Angreifer einen erweiterten Zugriff auf das System hat und auch die Datenkonsistenz beeinflussen kann. Ebenfalls findet diese Kategorie auch ihre Verwendung bei Schwachstellen, welche einen Ausfall des Streamings für viele Personen zur Folge hat oder

den Angreifer in die Lage versetzt, auf die unverschlüsselten Streams zuzugreifen. Schwachstellen mit der Kategorie «Medium» können einen Teilausfall des Systems zur Folge haben. «Low» bedeutet, dass die Auswirkungen minimal sind und nur einen Einfluss auf nicht-wichtige Daten haben.

Tabelle 4.5: Einstufungen für die Auswirkungen bei einer Ausnutzungen der Schwachstelle.

Auswirkung	Beschreibung
High	- Ausfall des Streaming-Dienstes für viele Nutzer. - Unerlaubter Zugriff auf Streaming. - Datenkonsistenz ist beinträchtigt. - Angreifer hat Adminzugriff auf Teilsysteme.
Medium	- Ausfall des Streaming-Dienstes für wenige Nutzer. - Manipulation von Teilsystemen.
Low	- Zugriff auf nicht-nutzbare Daten. - Beinflussung von Teilsystemen mit niedriger Relevanz

Unter der Berücksichtigung der umgesetzten Anforderungen ist die Risikoanalyse im Anhang entstanden. Von 109 Risiken sind 92 mit der Kategorie «Low» und 17 weitere mit der Kategeorie «Medium» versehen. Eine Bewertung von «High» besitzt keine der untersuchten Schwachstelle. Das Ziel der Risikoanalyse ist, dass alle Risiken auf ein Minimum reduziert werden können. Deshalb wird in den folgenden Paragraphen erläutert, wie die Bewertung der Risiken mit Stufe «Medium» zustande gekommen ist und wie das Risiko zukünftig reduziert werden könnte. In Klammern sind jeweils die Requirements aufgelistet, welche noch nicht umgesetzt sind.

T1, T2 - Weitergabe des Streams über VPN Ein Kunde des TV-Anbieters kann anderen Personen den Stream über VPN-Technologien freigeben. Der Kundenstamm unseres Use-Case Partners hat viele Kunden mit einem hohen Wissenstand in Informatik, deshalb ist die Wahrscheinlichkeit «Medium», dass jemand diese Schwachstellen ausnützt. Die Auswirkungen sind ebenfalls «Medium», da durch diesen Schaden potentielle Kunden verloren gehen. Das Risiko kann abgeschwächt werden, in dem das Authentifizierungsverfahren verstärkt (R15) und pro Kunde eine Limitation der gleichzeitigen Logins eingeführt wird (R38).

T8, T90, T95, T106, T107 - Angreifer beschafft oder fälscht sich ein Token Wenn ein Angreifer sich ein gültiges Token erstellen oder abgreifen kann, ist er in der Lage die Streams zu entschlüsseln. Da dies die Hauptmotivation der Angreifer ist, haben diese Schwachstellen die Wahrscheinlichkeitsstufe «Medium» erhalten. Die Konsequenzen sind, dass potentielle Kunden verloren gehen oder der Vertrag mit einem Fernsehsender verletzt werden könnte. Aufgrund dessen sind die Auswirkungen eines erfolgreichen Angriffs mit der Stufe «Medium» bewertet. Wegen des Risikos muss der Tokenaustausch mit Public Key Pinning erfolgen (R13) und die Überprüfung der Tokens erweitert werden (R30, R33).

T42, T64, T94 - Verfügbarkeit von kritischen Systemen Wenn ein Angreifer mit einer DoS-Attacke kritische Systeme, wie das Multicast Eingangssignal, die TV API oder den FairPlay Server ausser Gefecht setzen kann, hat dies einen Ausfall beim Streaming zur Folge. Die Wahrscheinlichkeit eines solchen Ausfalls ist «Medium», da diese Systeme eine Angriffsfläche aufgrund der fehlenden Redundanz bieten. Als Folge ist bei T42 nur ein Teil des Senderangebots betroffen, deshalb ist der Einfluss mit «Medium» bewertet. Bei den restlichen Schwachstellen ist das gesamte Angebot betroffen und hat aufgrund dessen die Bewertung «High». Dieses Risiko kann mit redundanten Systemen (R2,R3) und einem erweiterten Monitoring System abgeschwächt werden (R4,R5,R6).

T49, T112 - Speichersysteme Eine Bedrohung muss nicht immer eine Person sein, sondern es kann sich dabei auch um einen Hardware- oder Datenbankausfall handeln. Ein Hardwareausfall ist ohne Redundanz möglich, deshalb haben diese Schwachstellen die Wahrscheinlichkeit «Medium» erhalten. Bei einem Ausfall der Chunk Info Datenbank (T112) können keine Playlisten mehr erstellt werden, sodass kein Kunde mehr vom Streamingangebot profitieren kann. Dieser Ausfall betrifft alle Kunden und deshalb ist der Einfluss mit «High» eingestuft. Bei den Festplatten der Cache Server (T49) ist der Einfluss nur mit «Medium» kategorisiert, da aufgrund der Redundanz nicht alle Sender betroffen sind. Das resultierende Risiko «Medium» kann reduziert werden durch die Verwendung von redundanten Speichersystemen (R7) und einem guten Monitoring (R5).

T62, T86 - Anpassung von Code oder Daten Sofern ein Angreifer Zugriff auf die Systeme erhält, kann er den Code oder auch Daten anpassen. Diese Art von Angriff wird als wahrscheinlich erachtet und ist deshalb mit «Medium» bewertet. Bei der TV API und dem Key Management System sind die Einflüsse «High», da die Konsequenzen das gesamte Senderangebot und auch den gesamten Kundenstamm betreffen. Da solche Zugriffe nur über die Administratorenlogins möglich sind, müssen diese gut abgesichert werden (R39,R40,R41).

T96, T97 - Angreifbarkeit durch Administratoren Durch die erweiterten Zugriffsrechte haben Administratoren sehr viel Macht. Ein Mitarbeiter mit Administratorenrechten könnte der Firma grossen Schaden zufügen. Ebenfalls besteht die Gefahr, dass es einem Angreifer gelingt, ein Administratoren-Konto zu übernehmen. Damit hätte er auch einen erweiterten Zugriff. Phishing Attacken sind sehr häufig und daher mit der Wahrscheinlichkeit «Medium» eingestuft. Im Prototyp haben Administratoren uneingeschränkten Zugriff, deshalb ist der Einfluss mit «High» kategorisiert. Das Risiko kann minimiert werden, in dem Administratoren gegen Phising sensibilisiert werden (R16). Die Administratoren sollten gezielt ausgewählt werden und mit einem persönlichen Account nur die nötigen Zugriffsrechte erhalten (R17,R39,R41). Zusätzlich würden eine stärkere Authentifizierung und Passwortablaufregeln die Situation entschärfen (R15,R19,R43).

T98 - Fälschung des Apps Der App Store ist ein externes System, worauf der TV-Anbieter nur einen eingeschränkten Einfluss hat. Ein Angreifer könnte die Kunden täuschen, in dem er ein App mit einem leicht verwechselbaren Namen aufschaltet (Bsp. TV App *Original). Da jeder in der Lage ist, solche Apps im Store zu veröffentlichen, ist hier die Wahrscheinlichkeit auf «Medium». Mit diesem Angriff kann der Angreifer Kundendaten (Bsp. Login) von den Kunden abgreifen. Dies könnte es dem Angreifer in einer späteren Version des Authentifizierungssystems erlauben, die Streams mit einem fremden Login zu konsumieren. Dem Betreiber bleibt nichts anderes übrig, als regelmässige die Apps im Store zu prüfen und Nachahmer den Store-Betreibern zu melden (R26).

4.6.4 Leistung

Aufgrund des Prototyp-Status der Implementation konnten keine detaillierten Messungen der Leistung erstellt werden. Die Messungen wurden auf einem Server mit Intel Xeon Prozessor E3-1265L v2 und vier Kingston 9965525-118.A00LF 8GB gemacht. Insgesamt standen also vier CPU-Cores mit acht Threads und maximal 2.5 GHz Taktrate sowie 32 GB Arbeitsspeicher zur Verfügung. Für die Tests wurde eine 1 TB Festplatte mit 7200 rpm (WDC WD1003FBYZ-010FB0) eingesetzt.

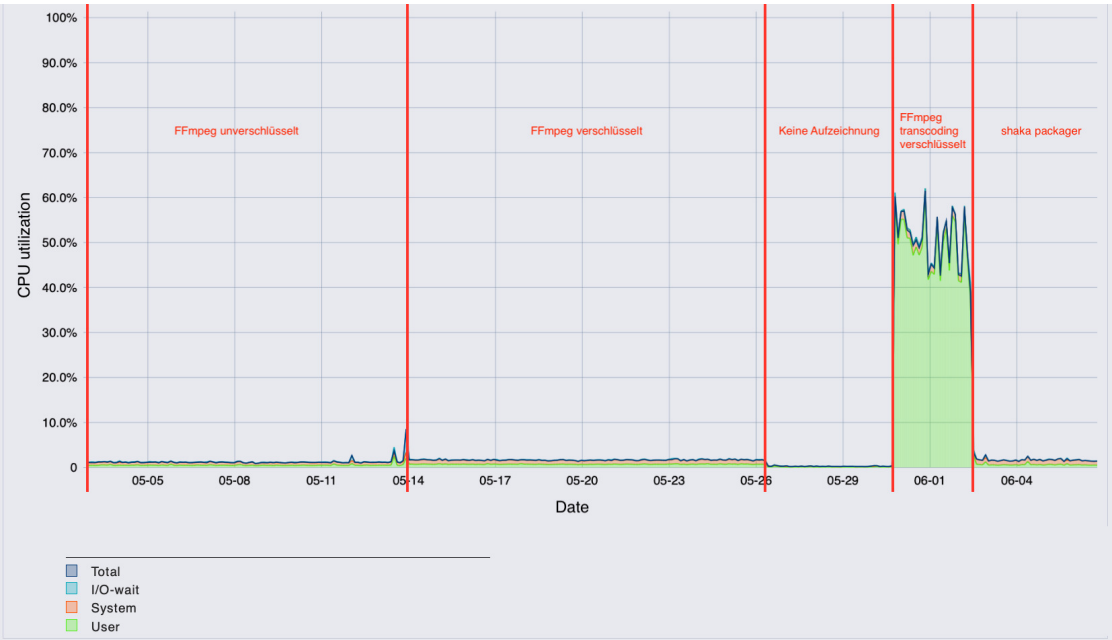


Abbildung 4.11: CPU-Leistungsgraph des Catch Servers (Siehe 7.2.7 für grössere Version)

Im CPU-Graph in Abbildung 4.11 erkennt man für den ersten Test mit dem bestehenden System eine CPU-Auslastung von ungefähr 1 %. Beim zweiten Test mit FFmpeg mit Verschlüsselung ist die durchschnittliche CPU-Auslastung mit ungefähr 1.47 % minimal höher. Ohne Aufzeichnung betrug die Auslastung erwartungsgemäss nahezu 0 %. Wurden die Spuren transcodiert und verschlüsselt, lag die durchschnittliche Nutzung bei 40.23 %. Beim letzten Versuch mit dem Shaka Packager betrug die Auslastung ungefähr 1.38 %.

5 Diskussion und Ausblick

5.1 Diskussion

Die Arbeit hatte drei konkrete Ziele: einen Architekturentwurf, Entscheidungen und Überlegungen klar zu dokumentieren und einen Prototyp zu entwickeln. Wie sich herausstellte, gab es für den spezifischen Anwendungsfall Live-Streaming DRM sehr wenig akademische Literatur und gewisse Quellen im Internet verbreiteten falsche Informationen. Aus diesem Grund bewährte sich die Fokussierung der Arbeit auf die ersten beiden Ziele. Es zeigt ebenfalls, dass diese Arbeit sehr relevant ist, weshalb der Entwurf und die Entscheidungen umso systematischer und detaillierter analysiert und erfasst wurden.

Die kleine Menge an verfügbarer Literatur bedeutet für Anbieter einen beträchtlichen Aufwand, soll ein DRM-System intern entwickelt und betrieben werden. Sie müssen zuerst sehr viel Fachwissen erarbeiten oder entsprechende Fachpersonen einstellen. Deshalb gibt es diverse spezialisierte Anbieter, deren Fachgebiet die Beratung zu Streaming- und DRM-Systeme und deren Verkauf ist. Gerade für kleinere Firmen ist es kosteneffizienter, solche Lösungen einzukaufen.

5.1.1 Bisheriger Zustand

Die Analyse des bestehenden Systems der Init7 zeigt klar, dass bei dessen Entwicklung Sicherheit keine Anforderung war. Die Server sind zum Teil frei aus dem Internet erreichbar und kommunizieren untereinander ohne Authentifizierung oder Verschlüsselung. Sogar die Kunden greifen nur mit IP-basierter Authentifizierung auf das System zu. Das System ist komplex aufgebaut und die Komponenten sind stark miteinander gekoppelt. Es gibt keine einheitlichen Schnittstellen zwischen den Systemen und es sind viele verschiedene Tools, Frameworks und Libraries im Einsatz. All diese Punkte gestalten es schwierig, das System zu warten und effizient zu erweitern. Zudem gibt es durch die starke Kopplung diverse Single-Points-of-Failure und es ist anspruchsvoll, die Systeme redundant zu betreiben.

5.1.2 Anforderungen

Von den FURPS-Anforderungen konnten einige nicht umgesetzt werden, da sie abhängig vom Einsatzzweck sind oder sich nicht im Umfang dieser Arbeit befanden. Dafür war es problemlos und einfach möglich, die Restlichen umzusetzen. Zu erwähnen ist ebenfalls, dass viele funktionale Anforderungen direkt wieder als Sicherheitsanforderungen in der Bedrohungsanalyse auftauchten. Aufgrund der Geschäftsanforderungen ergab sich, dass mindestens zwei verschiedene DRM-Systeme, also eine Multi-DRM fähige Lösung, eingesetzt werden muss. Da das VLCKit der tvOS App keine DRM-Unterstützung bietet, muss dieses ersetzt werden.

5.1.3 Architekturentwurf

Der Architekturentwurf zeigte sich als sehr solide und für den angedachten Einsatzzweck geeignet. Der ausgeprägte Einsatz von Microservices vereinfachte diverse Herausforderungen, verhinderte einige Sicherheitslücken und sorgte für einen problemlosen und zuverlässigen Schlüsselaustausch.

Zusätzlich zeigte sich, wo der Entwurf noch zu verbessern ist. Beispielsweise ist der Einsatz einer zentralen Schlüsseldatenbank nicht optimal. Ein Angreifer muss nur die Datenbank kompromittieren und kann dadurch die ganze DRM-Lösung ausser Kraft setzten. Es ist ebenfalls beträchtlich aufwändiger, die Lösung zu skalieren und vor Ausfällen zu schützen. Weiter ist es mit der vorgeschlagenen Architektur nicht trivial, ein Vorladen der Schlüssel, wie in Unterabschnitt 2.5.2 erwähnt, zu implementieren, da sie erst bei Gebrauch generiert werden. Der Key

Generator müsste dann direkt mit der TV API kommunizieren können, um ihr die Schlüsselreferenzen im vornherein bereitzustellen. Es bietet keinen Vorteil, die Schlüssel im Voraus dem Key Server zu senden, denn dieser weiss nicht, welcher Schlüssel zu welchem Kanal gehört.

5.1.4 Bedrohungsanalyse

Das Vorgehen aus Unterabschnitt 3.2.3, die erarbeiteten Analysen im Team zu diskutieren, hat sich als sehr gut erwiesen. Es gab bei einigen Punkten Meinungsverschiedenheiten und Ergänzungen, welche zu einem umfassenderen Gesamtüberblick führten. Folgende Schlüsse der Bedrohungsanalyse sind speziell zu erwähnen:

- Man darf den eigenen Systemen nicht blind vertrauen. Die Authentifizierung darf nicht nur über die IP-Adresse vollzogen werden, sondern muss immer über geheime Kenndaten, wie einem Passwort oder einem Token, stattfinden. Diese dürfen nur für eine einzige Verbindung von zwei Entitäten benutzbar sein. Ansonsten muss ein Angreifer lediglich ein System kompromittieren und hat damit Zugriff auf alle weiteren Systeme.
- Eine weitere grosse Gefahr sind Mitarbeiter oder Administratoren. Egal ob eine Kompromittierung versehentlich oder böswillig erfolgt, diese Personengruppe hat auf alle Systeme Zugriff und arbeitet täglich mit ihnen. Dadurch können sie mit wenig Aufwand einen hohen Schaden anrichten. Erschwerend kommt hinzu, dass kein ausreichendes Logging betrieben wird, welches forensische Analysen zulassen würde.
- Ein System verteilt aufzubauen hat nicht nur Vorteile bei der Skalierung eines Dienstes, sondern auch ganz direkt Einfluss auf dessen Robustheit. Wird nicht alles zentral betrieben, ist der Schaden bei einem Angriff auf ein einzelnes, kleines System viel kleiner, als wenn ein grosses, zentrales System kompromittiert wird.
- Diverse Sicherheitsanforderungen sind sehr einsatzspezifisch. So konnte die Anforderung ein überwachtes und selbstheilendes System nicht evaluiert werden.

5.1.5 Implementation

Da HLS mit den bestehenden Client-Plattformen nur Apple FairPlay Streaming unterstützt und im bestehenden System nur HLS zur Verfügung stand, wurde als DRM ein FPS Server eingesetzt. Sollte Android TV unterstützt werden, müsste auf das Streaming-Protokoll DASH gewechselt werden. Dann würde Apple TV nicht mehr unterstützt werden. Der Umbau auf DASH hätte den Rahmen dieser Arbeit überstiegen. Die Entwicklung dieses FPS Servers und speziell des KSM war aufwändiger als erwartet und benötigte mehr Zeit als geplant. Die dazu verwendete Referenzimplementierung in C wird von Apple nicht für den produktiven Einsatz empfohlen. Man sieht dies klar in der Gestaltung der öffentlichen Schnittstellen des C Codes, welche Callbacks und statische Variablen nutzt. Dies erschwert die Interoperabilität mit anderen Programmiersprachen.

Bei der Implementation kam das Fehlen von Literatur zum Thema am meisten zum Tragen. Während der Planung wurde wesentlich unterschätzt, wie eng Aufzeichnung und Packaging mit der Verschlüsselung zusammenhängen. Es wurde nicht erkannt, dass die verwendeten Containerformate und Codecs für ein zukünftiges Multi-DRM-System weit weniger flexibel gewählt werden können als angenommen. Zu spät wurde erkannt, dass das bisher eingesetzte Tool FFmpeg CENC nicht unterstützt. Es wurden aufwändige Versuche unternommen, FFmpeg mit Shaka Packager von Google zu ersetzen. Diese hatten jedoch keinen Erfolg, da die bestehende HLS-Architektur und insbesondere auch die TV API kein CMAF unterstützen.

Zuletzt wurde auch die proprietäre Art von Apple FairPlay Streaming zum Problem. Um einen End-to-End Test durchzuführen, war es nötig, von Apple das «FPS Deployment Package» zu beantragen, welches den letzten Teil der Krypto-Implementation, den ASK und das benötigte FPS Zertifikat beinhaltet. Das war ein langsamer, manueller und administrativ aufwändiger Prozess, bei welchem viele Details der geplanten Nutzung angegeben werden mussten.

Die weiteren, geplanten Implementationsdetails und -anforderungen haben sich bewährt. So half der Einsatz von Docker und Django, eine einfache Entwicklungsumgebung zu schaffen, welche es erlaubte, schnell und auf einem abstrakten Niveau Applikationslogik zu implementieren.

5.1.6 Evaluation

Die Anzahl der Schlüsselanfragen können abhängig von der Schlüsselrotationsdauer auch bei wenigen Zuschauern schon enorm ansteigen. Deshalb ist es wichtig, das Zeitintervall richtig zu wählen. Gemäss der Abbildung 4.10 sollte das Intervall höher als 20 Minuten sein, da die Anzahl der Anfragen andernfalls exponentiell ansteigen. Das Zeitintervall sollte allerdings nicht zu hoch gewählt werden, da sonst eine Angriffsfläche für Brute-Force Attacken entsteht.

Bei der Entwicklung des Architekturentwurfes wurden gewisse Bedrohungen unterschätzt, welche sich anschliessend in der Risikoanalyse als gefährlicher herausgestellt haben. So wurde gegen die Gefahr von Mitarbeitenden oder Single-Points-of-Failure wenig unternommen. Auch ist die Verwendung eines VPNs für den Business-Case eine wesentlich grössere Gefahr als erwartet. Es gab insgesamt aber keine sehr grossen Bedrohungen und alle können mitigiert werden.

Die Leistungseinbussen für die Verschlüsselung sind beim Prototyp minimal. Im Vergleich zum Transcodieren ist dieser Leistungszuwachs absolut vernachlässigbar und es bedarf keiner grösseren Kapazitätssteigerung der Systemleistung.

5.2 Ausblick

Bei der entwickelten Anwendung handelt es sich um einen Prototypen. Es gibt daher viele Möglichkeiten, wie das Projekt weiterentwickelt werden kann. Diese werden in diesem Kapitel noch weiter erläutert.

5.2.1 Weiterentwicklung Prototyp

27 Security Requirements sind im Prototypen nicht vollumfänglich umgesetzt. Um allen Sicherheitsüberlegungen des Architekturentwurfs zu entsprechen, müssten diese in einem weiteren Schritt ebenfalls implementiert werden. Viele Anforderungen beziehen sich auf die Infrastruktur und die Betreiber des Systems. Deshalb wäre es sinnvoll, dass dieser eine Testumgebung einrichtet. Das Monitoring und Logging wurden im Prototyp ebenfalls vernachlässigt. Da könnte man noch ein System implementieren, welches zuverlässig überwacht und Unregelmässigkeiten meldet. Die einzelnen Serverkomponenten laufen zur Zeit nicht alle redundant. Um die Ausfallsicherheit zu erhöhen und den Erfolg von DoS-Attacken zu minimieren, sollte dies noch umgesetzt werden. Zusätzlich gäbe es eine Möglichkeit, die Architektur zu verbessern. Die Kommunikation vom Key Generator zum Catch Server ist über Dateien realisiert. Dies aus dem Grund, dass der FFmpeg Packager nur eine solche Kommunikation zu lässt. Durch die Entwicklung eines neuen Packager könnte diese Kommunikation per Message-Queue oder https realisiert werden.

5.2.2 Erweiterung Multi-DRM System

Die Architektur sieht vor, dass zusätzlich zu Apple FairPlay Streaming auch noch weitere DRM-Systeme an den Key Server angeschlossen werden können. Damit dies umgesetzt werden kann, muss der Packager neu implementiert werden. Der bestehende Packager der Init7 unterstützt die speziellen Formate nicht, deshalb konnte dies im Projekt nicht umgesetzt werden. Damit beispielsweise HLS und DASH in einem Multi-DRM System betrieben werden können, muss der Packager die Dateien im CMAF Format, also FMP4 abspeichern [24] . Ebenfalls sollte die Verschlüsselung gemäss der Common Encryption funktionieren. Mit einem neuen Packager kann man einen neuen Microservice schreiben, welcher Google Widevine Anfragen beantwortet.

5.2.3 Optimierung Schlüsselaustausch

Beim Schlüsselaustausch gibt es noch Möglichkeiten für Optimierungen. Im aktuellen System werden die Schlüssel immer erst abgefragt, wenn sie gebraucht werden. Durch unsere Analyse von anderen Lösungen haben wir festgestellt, dass es sinnvoll ist, diese Schlüssel vorgängig zu generieren und zu verteilen. Andernfalls könnte es bei einem hohen Anfragevolumen zu Ausfällen kommen. Die Anfragen steigen massiv an, wenn viele Personen gleichzeitig den Fernseher einschalten. Dies könnte beispielsweise bei einem Finalspiel der Fussball WM der Fall sein.

5.2.4 Authentifizierungssystem

Im aktuellen System wird die Authentfizierung über IP-Filterung geregelt. Dies hat zur Folge, dass die Kunden den TV-Stream nur zu Hause konsumieren können. Für Geräte mit mobilem Internet stellt dies ein Problem dar, da diese Geräte nicht immer im gleichen IP-Range sind. Deshalb wäre es sinnvoll, auf eine alternative Authentifizierungsmethode umzusteigen. Die Architektur beschreibt bereits, wie so eine Lösung mit JSON Web Token Authentifizierung umgesetzt werden könnte. Für die Umsetzung wird das Python-Modul Simple JWT vorgeschlagen [65]. Simple JWT stellt das Backend für eine JSON Web Token Authentifizierung für das Django REST Framework bereit. Der Einsatz eines Authentifizierungsservers würde zusätzlich das Problem lösen, dass Kunden den Stream über VPN Technologien weitergeben können.

5.2.5 Redundanz und Cloud-Unterstützung

Die Datenbanken und Dateisysteme sind in der momentanen Architektur Single-Points-of-Failure. Für die Hochverfügbarkeit könnte man da verteilte Systeme und weitere cloudbasierte Ansätze verwenden. Eine gute Quelle für weitere Ansätze bietet die Facharbeit «Transitioning Broadcast to Cloud» [26].

5.2.6 Optionen von DRM-Systemen nutzen

DRM-Systeme wie Apple FairPlay Streaming, Google Widevine und Microsoft PlayReady bieten zusätzlich zu der Verschlüsselung noch weitere Möglichkeiten. Beispielsweise können das Ablaufdatum des Schlüssels festgelegt oder die Art der Abspielgeräte eingeschränkt werden. Dadurch ist es möglich, die Inhalte auch ohne aktive Internetverbindung zu konsumieren. Diese Möglichkeit wird bei der aktuellen Lösung nicht unterstützt. Bei HLS können mit der Erstellung einer Masterplaylist auch verschiedenen Auflösungen und Qualitäten ausgeliefert werden. Bezüglich optimaler Wahl dieser Konfigurationen gibt es noch Forschungspotenzial.

6 Verzeichnisse

6.1 Literaturverzeichnis

[1] moneyland.ch AG. *Netflix und Spotify mit massiv mehr Nutzern*. 15. Apr. 2020. URL: https://www.moneyland.ch/de/streaming-schweiz-studie-2020 (besucht am 27. 05. 2021).

[2] Stefan Ehrbar. „Einigung mit Teleclub - CH Media gewinnt Kampf um TV-Rechte für Champions League – SRG kritisiert die Uefa“. In: *Luzerner Zeitung* (). URL: https://www.luzernerzeitung.ch/wirtschaft/ch-media-gewinnt-kampf-um-tv-rechte-fur-champions-league-srg-kritisiert-die-uefa-ld.1236210 (besucht am 27. 05. 2021).

[3] *Senderübersicht TV7*. URL: https://www.init7.net/de/tv/senderuebersicht/ (besucht am 21. 03. 2021).

[4] Init7 (Schweiz) AG. *TV7*. Version 2.0.3. 11. Juni 2021. URL: https://apps.apple.com/ch/app/tv7/id1362706442 (besucht am 21. 03. 2021).

[5] Init7 (Schweiz) AG. *TV7 – Apps bei Google Play*. 11. Juni 2021. URL: https://play.google.com/store/apps/details?id=net.init7.tv&hl=de_CH&gl=CH (besucht am 21. 03. 2021).

[6] *Verschlüsselungsverfahren im Überblick*. IONOS Digitalguide. 24. Jan. 2018. URL: https://www.ionos.de/digitalguide/server/sicherheit/verschluesselungsverfahren-ein-ueberblick/ (besucht am 02. 05. 2021).

[7] Jolly Shah und Vikas Saxena. „Video Encryption: A Survey“. In: *arXiv:1104.0800 [cs]* (5. Apr. 2011). arXiv: 1104.0800. URL: http://arxiv.org/abs/1104.0800 (besucht am 17. 03. 2021).

[8] Yogita Negi. „A Survey on Video Encryption Techniques“. In: *International Journal of Emerging Technology and Advanced Engineering* 3.4 (Apr. 2013), S. 234–237. ISSN: 22502459.

[9] Akinbowale Nathaniel Babatunde u. a. „Survey of Video Encryption Algorithms“. In: 5.1 (Juni 2017). URL: https://core.ac.uk/download/pdf/268593983.pdf (besucht am 24. 02. 2021).

[10] Alexander Wong und William Bishop. *An Efficient, Parallel Multi-Key Encryption of Compressed Video Streams*. 2. 2006, S. 69–74. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.217.1593&rep=rep1&type=pdf.

[11] Mohammad Banikazemi und Raj Jain. „IP Multicasting: Concepts, Algorithms, and Protocols“. In: *Survey Paper, Ohio State University* (1997). URL: https://www.cse.wustl.edu/~jain/cis788-97/ftp/ip_multicast/index.html.

[12] Alex MacAulay, Boris Felts und Yuval Fisher. *WHITEPAPER - IP Streaming of MPEG-4: Native RTP vs MPEG-2 Transport Stream*. Whitepaper. Envivio Inc, 2005. URL: http://pdf.textfiles.com/manuals/STARINMANUALS/Envivio/Manual/White%20Paper%20-%20IP%20Streaming%20of%20MPEG-4%20-%20Native%20RTP%20vs%20MPEG-2%20Transport%20Stream.pdf.

[13] *HLS Authoring Specification for Apple Devices*. Apple Developer Documentation. 22. Juni 2020. URL: https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices (besucht am 04. 06. 2021).

[14] Roger Pantos und William May. *HTTP Live Streaming*. Internet Request for Comments. ISSN: 2070-1721. Aug. 2017. URL: https://tools.ietf.org/html/rfc8216 (besucht am 17. 03. 2021).

[15] Iraj Sodagar. „The MPEG-DASH Standard for Multimedia Streaming Over the Internet“. In: *IEEE MultiMedia* 18.4 (Apr. 2011). Conference Name: IEEE MultiMedia, S. 62–67. ISSN: 1941-0166. DOI: 10.1109/MMUL.2011.71. URL: https://ieeexplore.ieee.org/document/6077864?arnumber=6077864.

[16] Ron Garrison. *Structure of a MPEG-DASH MPD - A Detailed Guide*. OTTVerse. Section: Video Streaming. 1. Apr. 2021. URL: https://ottverse.com/structure-of-an-mpeg-dash-mpd/ (besucht am 04. 06. 2021).

[17] *Microsoft PlayReady – Product Overview*. URL: https://www.microsoft.com/playready/features/DigitalMediaStrategy.aspx (besucht am 18. 03. 2021).

[18] openspecs-office. *[MS-SSTR]: Smooth Streaming Protocol*. 30. Okt. 2020. URL: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sstr/8383f27f-7efe-4c60-832a-387274457251 (besucht am 04. 06. 2021).

[19] *What is HDS streaming? | HLS vs. HDS*. Cloudflare. URL: https://www.cloudflare.com/de-de/learning/video/what-is-http-dynamic-streaming/ (besucht am 04. 06. 2021).

[20] Lars Borg. *HTTP Dynamic Streaming Specification Version 3.0 FINAL*. 2013. URL: https://wwwimages2.adobe.com/content/dam/acom/en/devnet/hds/pdfs/adobe-hds-specification.pdf.

[21] Stefan Pham u. a. „On the current state of interoperable content protection for internet video streaming“. In: *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*. 2018 IEEE Seventh International Conference on Communications and Electronics (ICCE). Hue, Vietnam: IEEE, 18. Juli 2018, S. 13–17. ISBN: 978-1-5386-3679-4. DOI: 10.1109/CCE.2018.8465735.

[22] Apple Inc. *FairPlay Streaming Programming Guide*. 11. Juli 2019. URL: https://developer.apple.com/services-account/download?path=/Developer_Tools/FairPlay_Streaming_Server_SDK/FairPlay_Streaming_Server_SDK_v4.4.zip (besucht am 23. 05. 2021).

[23] Krishna Rao Vijayanagar. *EME, CDM, AES, CENC, and Keys - The Essential Building Blocks of DRM*. OTTVerse. Section: DRM. 28. Aug. 2020. URL: https://ottverse.com/eme-cenc-cdm-aes-keys-drm-digital-rights-management/ (besucht am 11. 05. 2021).

[24] Krishna Rao Vijayanagar. *Apple FairPlay Streaming DRM - How Does It Work?* OTTVerse. Section: DRM. 30. Aug. 2020. URL: https://ottverse.com/apple-fairplay-drm-how-does-it-work/ (besucht am 26. 05. 2021).

[25] *Widevine DRM Architecture Overview*. 6. März 2017. URL: https://www.whymatematica.com/wp-content/uploads/2018/08/Widevine_DRM_Architecture_Overview.pdf (besucht am 04. 05. 2021).

[26] Yuriy Reznik, Jordi Cenzano und Bo Zhang. „Transitioning Broadcast to Cloud“. In: *Applied Sciences* 11.2 (Jan. 2021). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, S. 503. DOI: 10.3390/app11020503. URL: https://www.mdpi.com/2076-3417/11/2/503 (besucht am 24. 02. 2021).

[27] International Organization for Standardization. *ISO/IEC 23001-7:2016*. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/80/68042.html (besucht am 05. 06. 2021).

[28] *About the Common Media Application Format with HTTP Live Streaming*. Apple Developer Documentation. URL: https://developer.apple.com/documentation/http_live_streaming/about_the_common_media_application_format_with_http_live_streaming (besucht am 04. 06. 2021).

[29] *Angebote | Swisscom*. URL: https://www.swisscom.ch/de/business/broadcast/angebote.html#iptv (besucht am 24. 04. 2021).

[30] *TV-as-a-Service | Swisscom*. URL: https://www.swisscom.ch/de/business/broadcast/angebote/iptv/tv-as-a-service.html#tab=0 (besucht am 24. 04. 2021).

[31] *IPTV | Swisscom*. URL: https://www.swisscom.ch/de/business/broadcast/angebote/iptv.html (besucht am 24. 04. 2021).

[32] Ömer Kayali und Kim Berkemeyer. *RTL & ProSieben sagen SD-Abschaltung ab: HD-Dilemma zwingt Privatsender zum Umdenken*. CHIP Online. 27. Jan. 2020. URL: https://www.chip.de/news/RTL-und-ProSieben-sagen-SD-Abschaltung-ab-HD-Dilemma-zwingt-TV-Sender-zum-Umdenken_146064759.html (besucht am 29. 05. 2021).

[33] *VERIMATRIX: Content Security, Code & Mobile App Protection*. VERIMATRIX. URL: https://www.verimatrix.com/ (besucht am 29. 05. 2021).

[34] *TV Platforms & Content Protection | Viaccess-Orca*. URL: https://www.viaccess-orca.com (besucht am 29. 05. 2021).

[35] *Widevine | Widevine*. URL: https://www.widevine.com (besucht am 29. 05. 2021).

[36] *Video Experience Solutions*. Synamedia. URL: https://www.synamedia.com/ (besucht am 29. 05. 2021).

[37] *Marlin Community - Open Standard DRM Technology*. Marlin Community. URL: http://www.marlin-community.com/ (besucht am 29. 05. 2021).

[38] *Verimatrix Company Overview*. VERIMATRIX. URL: https://www.verimatrix.com/about-us/ (besucht am 24. 05. 2021).

[39] *Multi-DRM Solution for Digital Rights Management*. VERIMATRIX. URL: https://www.verimatrix.com/products/multi-drm/ (besucht am 07. 06. 2021).

[40] *VCAS - Video Content Authority System*. VERIMATRIX. URL: https://www.verimatrix.com/products/vcas/ (besucht am 07. 06. 2021).

[41] *Forensic Watermarking for Video*. VERIMATRIX. URL: https://www.verimatrix.com/products/watermarking/ (besucht am 07. 06. 2021).

[42] *Content Protection | Viaccess-Orca*. URL: https://www.viaccess-orca.com/content-protection (besucht am 06. 06. 2021).

[43] *CAS & DRM as a Service | Viaccess-orca*. URL: https://www.viaccess-orca.com/content-security-services (besucht am 06. 06. 2021).

[44] *Conditional Access System*. URL: https://www.viaccess-orca.com/conditional-access-system (besucht am 06. 06. 2021).

[45] *DRM Solutions | Viaccess-Orca*. URL: https://www.viaccess-orca.com/drm-solutions (besucht am 06. 06. 2021).

[46] *OTT Security - Safeguard your OTT Content, Service and Revenue*. URL: https://www.synamedia.com/wp-content/uploads/2020/05/SYN102-OTT-Security-AAG.pdf (besucht am 06. 06. 2021).

[47] *Data Sheet - DCM PackagerProduct Overview*. URL: https://www.synamedia.com/wp-content/uploads/2019/09/VN-DCM-Packager-Data-Sheet.pdf (besucht am 06. 06. 2021).

[48] Marlin Developer Community. *Open-standard DRM technology*. URL: https://go.intertrust.com/hubfs/assets/Media/Marlin-open-standard-DRM-technology.pdf (besucht am 05. 06. 2021).

[49] Steve Lipner und Michael Howard. *Entwicklungszyklus für sichere Software*. Microsoft Docs. 30. Mai 2005. URL: https://docs.microsoft.com/de-de/previous-versions/technical-content/ms995349(v=msdn.10) (besucht am 09. 04. 2021).

[50] Marc Rennhard und Stephan Neuhaus. „Software and System Security 1 - Security Requirements Engineering and Threat Modeling - Part 1/2“. ZHAW School of Engineering, 2020. URL: https://olat.zhaw.ch/auth/RepositoryEntry/180846598/CourseNode/74113252610159/path%3D~~09%20Security%20Requirements%20Engineering%20and%20Threat%20Modeling/0 (besucht am 18. 01. 2021).

[51] Ann-Cathrin Klose. *Mit 12 Factors zur guten Cloud-App*. entwickler.de. 13. Nov. 2015. URL: https://entwickler.de/online/development/12-factor-app-manifest-cloud-189818.html (besucht am 07. 04. 2021).

[52] *The Twelve-Factor App Dies ist eine Übersetzung.* URL: https://12factor.net/de/ (besucht am 07. 04. 2021).

[53] *The STRIDE Threat Model*. Microsoft Docs. 11. Dez. 2009. URL: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20) (besucht am 28. 03. 2021).

[54] International Organization for Standardization. *ISO/IEC 25010:2011*. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/57/35733.html (besucht am 05. 06. 2021).

[55] Joint Task Force Transformation Initiative. *Guide for conducting risk assessments*. NIST SP 800-30r1. Edition: 0. Gaithersburg, MD: National Institute of Standards und Technology, Sep. 2012. DOI: 10.6028/NIST.SP.800-30r1. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf (besucht am 06. 06. 2021).

[56] *google/shaka-packager*. original-date: 2014-04-21T17:34:50Z. 4. Juni 2021. URL: https://github.com/google/shaka-packager (besucht am 06. 06. 2021).

[57] *google/ExoPlayer*. original-date: 2014-06-13T21:19:18Z. 11. Juni 2021. URL: https://github.com/google/ExoPlayer (besucht am 11. 06. 2021).

[58] *VideoLAN / VLCKit*. Version v3.3.16. 19. Mai 2021. URL: https://code.videolan.org/videolan/VLCKit (besucht am 06. 06. 2021).

[59] *secrets — Generate secure random numbers for managing secrets*. Python 3.9.5 documentation. 2021. URL: https://docs.python.org/3/library/secrets.html (besucht am 09. 06. 2021).

[60] 1Blademaster. *A quick tour of the Python Secrets module*. DEV Community. 4. Apr. 2021. URL: https://dev.to/1blademaster/the-python-secrets-module-1c3 (besucht am 09. 06. 2021).

[61] *FFmpeg Formats Documentation*. URL: https://ffmpeg.org/ffmpeg-formats.html#Options-8 (besucht am 10. 06. 2021).

[62] *Universally Unique Identifier*. In: *Wikipedia*. Page Version ID: 212661071. 4. Juni 2021. URL: https://de.wikipedia.org/w/index.php?title=Universally_Unique_Identifier&oldid=212661071 (besucht am 06. 06. 2021).

[63] *4. Building C and C++ Extensions*. Python 3.9.5 documentation. 10. Mai 2021. URL: https://docs.python.org/3/extending/building.html (besucht am 07. 06. 2021).

[64] *AVAssetResourceLoader*. Apple Developer Documentation. URL: https://developer.apple.com/documentation/avfoundation/avassetresourceloader (besucht am 07. 06. 2021).

[65] *jazzband/djangorestframework-simplejwt*. original-date: 2017-05-05T17:41:35Z. 11. Juni 2021. URL: https://github.com/jazzband/djangorestframework-simplejwt (besucht am 11. 06. 2021).

6.2 Glossar

Adaptive Bitrate Mit Adaptivem Bitrate Streaming kann die Komprimierungsstufe und die Videoqualität dynamisch an die verfügbare Bandbreite angepasst werden. 19, 59

Application Secret Key Ein 16 Byte grosser geheimer Schlüssel, der zusammen mit dem Apple FairPlay Streaming Zerifikat für die Verschlüsselung des FPS Schlüsselaustasches verwendet wird. 40, 59

Celery Celery Distributed Task Queue ist eine Task Queue, mit der sich grosse Mengen von Nachrichten synchron und asynchron verarbeiten lassen. 39

Chunk Eine Datei, dessen Inhalt aus einem zeitlich begrenzten Ausschnitt oder Segment eines kontinuierlichen (Video-)Datenstroms besteht. 12, 14, 28–30

Counter mode Vewendet die Blockchiffre, um eine Stromchiffre zu erzeugen. Daten werden durch XOR-Verknüpfung mit dem Schlüsselstrom ver- und entschlüsselt. Der Schlüsselstrom kann auch vorberechnet werden. 14, 59

FFmpeg Eine quelloffene Sammlung von Programmen und Bibliotheken, welche für die digitale Audio- und Videoverarbeitung verwendet werden können. Das Audio- und Videomaterial kann aufgenommen, konvertiert, gestreamt und verpackt werden. 3, 4, 27, 28, 39, 41, 46, 48

MPEG Transport Stream Von der Moving Picture Experts Group entwickeltes Transportstream-Protokoll für Video/Audio-Übertragung. 11, 60

nftables nftables ist der Nachfolger von iptabels. Es ist ein Subsystem des Linux Kernels und ermöglicht die Filterung und Klassifizierung von Netzwerkpaketen, Datagrammen und Frames. 30, 33, 35

Over the Top Content Bereitstellung von kostenlosem und kostenpflichtigen Bewegtbildern, die über eine Internetverbindung zur Verfügung gestellt werden, ohne dass der Provider darauf einen Einfluss des Inhalts hat. 18, 60

Pay-per-View Es ist eine Bezahlmethode für Fernsehsendungen, bei der der Zuschauer nur die tatsächlich angesehenen Sendungen bezahlt. 11, 60

Pipe Eine Pipe ist eine Form der Interprozesskommmunikation eines Betriebssystemes. Im wesentlichen ist es ein Buffer, welcher nach dem First-in First-Out Prinzip funktioniert. 31

Set-Top-Box Auch Beistellbox genannt, ist ein Gerät, dass an den Fernseher angeschlossen wird und dem Kunden zusätzliche Nutzungsmöglichkeiten bietet. Sie empfängt digitale Daten, decodiert sie und gibt sie als Fernsehsignal weiter. 19, 60

Transmuxer Eine Soft- oder Hardware, welche die Datenströme eines Videostreams oder einer Videodatei in ein neues Containerformat packt. 3, 16

6.3 Abbildungsverzeichnis

6.4 Tabellenverzeichnis

6.5 Akronyme

AAC Advanced Audio Coding. 12, 28

ABR Adaptive Bitrate. 19

AC-3 Dolby Digital. 12

AES Advanced Encryption Standard. 3, 10, 12, 14–16, 20, 28, 41

ASK Application Secret Key. 40, 48

AVC Advanced Video Codec. 12, 28

CAS Conditional Access System. 19

CBC Cipher Block Chaining. 12, 14–16

CDM Content Decryption Modul. 14–16

CDN Content Delivery Network. 12, 18

CENC Common Encryption. 15, 16, 48, 49

CKC Content Key Context. 15, 40, 41

CMAF Common Media Application Format. 16, 48, 49

CTO Chief Technology Officer. 26

CTR Counter mode. 14, 16

DASH Dynamic Adaptive Streaming over HTTP. 6, 9, 10, 13–16, 18–20, 25, 48, 49

DCM Digital Content Manager. 19, 33

DES Data Encryption Standard. 10

DFD Data Flow Diagram. 7, 23, 24, 26, 33, 34, 36, 57, 58, 69, 70

DoS Denial of Service. 23, 37, 44, 49

DRF Django REST Framework. 39, 40

DRM Digital Rights Management. 3, 4, 6, 9, 12–20, 31, 39, 47–50, 58

DVB Digital Video Broadcasting. 11, 17–19

DVMRP Distance Vector Multicast Routing Protocol. 11

EPG Electronic Program Guide. 17

FMP4 fragmented MP4. 13, 16, 20, 28, 49

FPS Apple FairPlay Streaming. 3, 4, 6, 9, 14–16, 19, 20, 25, 31–33, 35, 39–41, 48–50, 56, 57

HD High Definition. 8, 18, 20, 33

HDS Adobe HTTP Dynamic Streaming. 6, 9, 10, 13, 16

HEVC High Efficiency Video Codec. 12

HLS HTTP Live Streaming. 6, 8–10, 12–16, 18–20, 25, 29–31, 41, 48–50, 57

HTTP Hypertext Transfer Protocol. 12, 13, 39

HTTPS Hypertext Transfer Protocol Secure. 33, 35, 39, 43

IDEA International Data Encryption Algorithm. 10

IGMP Internet Group Management Protocol. 11

IP Internet Protocol. 8, 47, 48

IPTV Internet Protocol Television. 11, 17, 20

ISOBMFF ISO base media file format. 16

ISP Internet Service Provider. 17

KMS Key Management System. 3, 8, 16, 18

KSM Key Security Module. 15, 40, 41, 48

M3U MP3-URL. 12, 13

MP4 MPEG-4. 14

MPD Media Presentation Description. 13

MPEG Moving Picture Experts Group. 10, 11, 13, 19, 20

MPEG-TS MPEG Transport Stream. 11, 20, 28, 29

MSS Microsoft Smooth Streaming. 6, 9, 10, 13, 16, 19, 25

OTT Over the Top Content. 18, 19

PIM Protocol Independent Multicast. 11

PKCS7 Public-Key Cryptography Standards #7. 12

PoC Proof of Concept. 41

PPV Pay-per-View. 11

PR Microsoft PlayReady. 6, 9, 14, 16, 19, 20, 50

RSA Rivest-Shamir-Adleman. 10

RTP Real-Time Transport Protocol. 11

SD Standard Definition. 8

SDK Software Development Kit. 16, 19, 20

SDL Security Development Lifecycle. 3, 4, 6, 9, 21, 25, 27, 57

SPC Server Playback Context. 15, 40, 41

SSH Secure Shell. 37

STB Set-Top-Box. 19, 20

SWS Software and System Security. 21, 23, 24, 58

TLS Transport Layer Security. 30, 38–40

TS Transport Stream. 14

UDP User Datagram Protocol. 11

UHD Ultra High Definition. 17, 19, 20

URL Unified Resource Locator. 13, 29

UUID Universally Unique Identifier. 40

VCAS Video Content Authority System. 19

VoD Video on Demand. 8, 11, 12, 19

WV Google Widevine. 6, 9, 14–16, 18–20, 49, 50

XML Extensible Markup Language. 13

7 Anhang

Im Abgabeordner befinden sich die folgenden Inhalte:

- Bachelorarbeit PDF: BA21_gueu_09.pdf
- Code: Git_Repos_BA21_gueu_09.zip

7.1 Projektmanagement

7.1.1 Aufgabenstellung

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Schlüsselmanagement und Verschlüsselung für Live-Videostreams
BA21_gueu_09

BetreuerInnen:	Gürkan Gür, gueu
Fachgebiete:	Information Security (IS)
Studiengang:	IT
Zuordnung:	Institut für angewandte Informationstechnologie (InIT)
Industriepartner:	Init7 (Schweiz) AG (8406 Winterthur)
Gruppengrösse:	3

Kurzbeschreibung:
Kunden von init7 erhalten TV direkt per Multicast-Signal oder per HLS (sowohl Live- als auch Replay-TV). Diese Streams sind soweit unverschlüsselt und können vom Kunden relativ einfach aufgezeichnet werden. Gewisse Privatfernsehsender verlangen aber für die Ausstrahlung in HD einen Kopierschutz bzw. eine Verschlüsselung der Streams.

In dieser Arbeit soll eine Komponente für das Streaming-System konzeptioniert und umgesetzt werden, welche diese Verschlüsselung macht. Eine besondere Herausforderung und das Kernstück der Arbeit wird dabei das Schlüsselmanagement sein.

In Scope sind:

* Eine Beschreibung der bestehenden Architektur * Eine Beschreibung der Anforderungen an das zu erstellende System - Welche Streams sollen verschlüsselt werden

- Welche Streams sollen verschlüsselt werden - Wozu soll die Verschlüsselung dienen? (Absicherung des Transports, Verhinderung einer Aufzeichnung, ...) * Ein Threat Modeling der vorgeschlagenen Lösung als Teil des Requirement Engineering (mit DFD/STRIDE und/oder attack trees) * Eine Übersicht über bestehende Technologien zum Verschlüsseln von Videostreams * Klärung der Frage, wer welche Schlüssel besitzen muss, wie die da hin kommen und wie die auf den Zielsystemen geschützt werden können und müssen. Das wird der theoretische Hauptteil der Arbeit sein. * Prototypische Implementation des Schlüsselmanagements. Das kann der praktische Hauptteil sein.

Out of Scope sind:

* Produktive Implementationen (wenn die Prototypen auch produktiv einsetzbar sind, umso besser, aber muss nicht) * Anbindung an oder Implementation von Standards wie FairPlay. Hier reichen gemockte Interfaces. Grund: FairPlay ist vermutlich sehr kompliziert und würde die Arbeit unzulässig verkomplizieren.

Unklar:

* Muss die Lösung mit bestehenden Systemen wie FairPlay kompatibel sein?

Voraussetzungen:
- Kenntnisse in Kryptographie

Die Arbeit ist vereinbart mit:
Nicolas Da Mutten (damutnic)
Daniela Egli (eglida03)
Andreas Meier (meiera25)

Thursday 10. June 2021 14:44

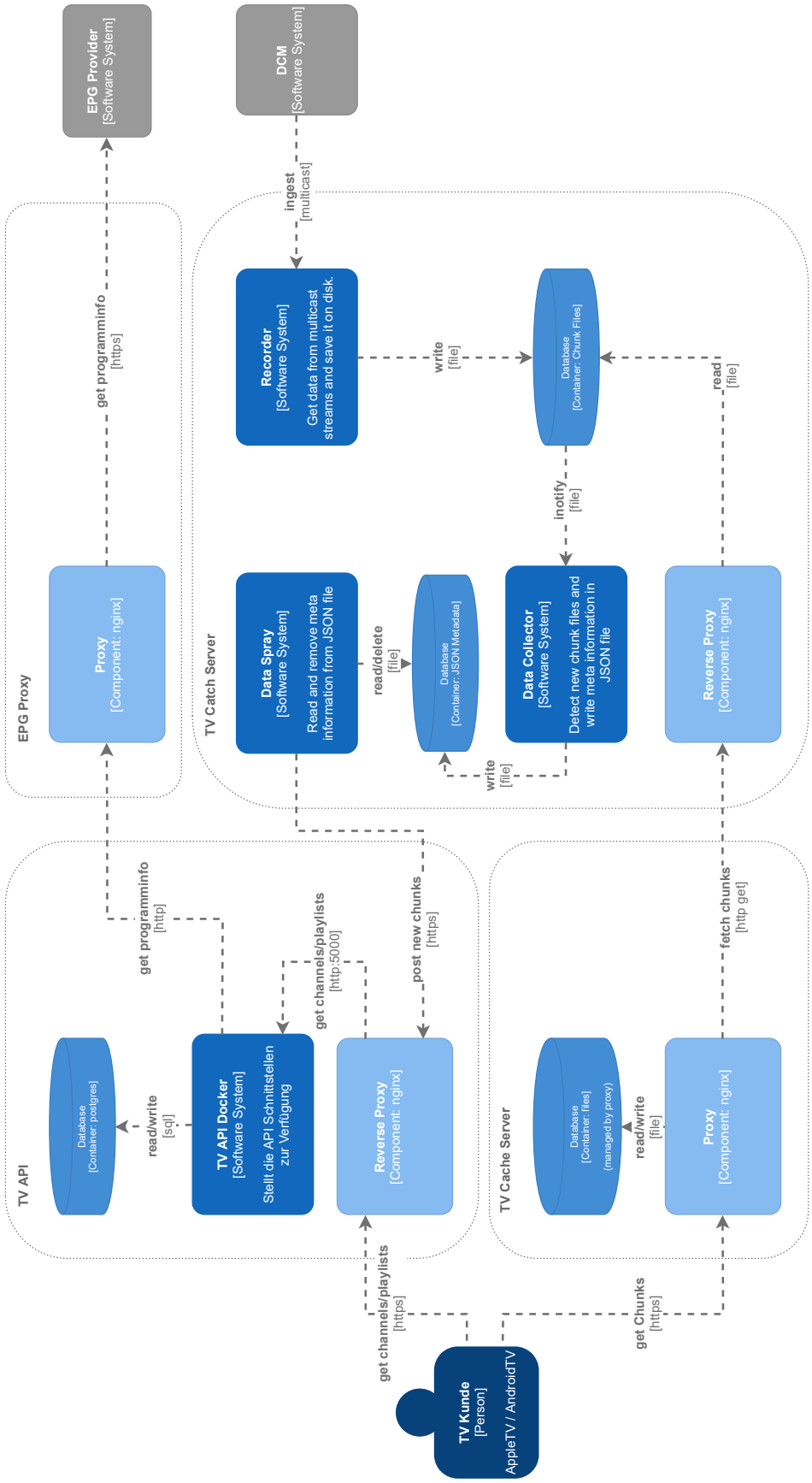
7.1.2 Zeitplan

Tabelle 7.1: Projektplan

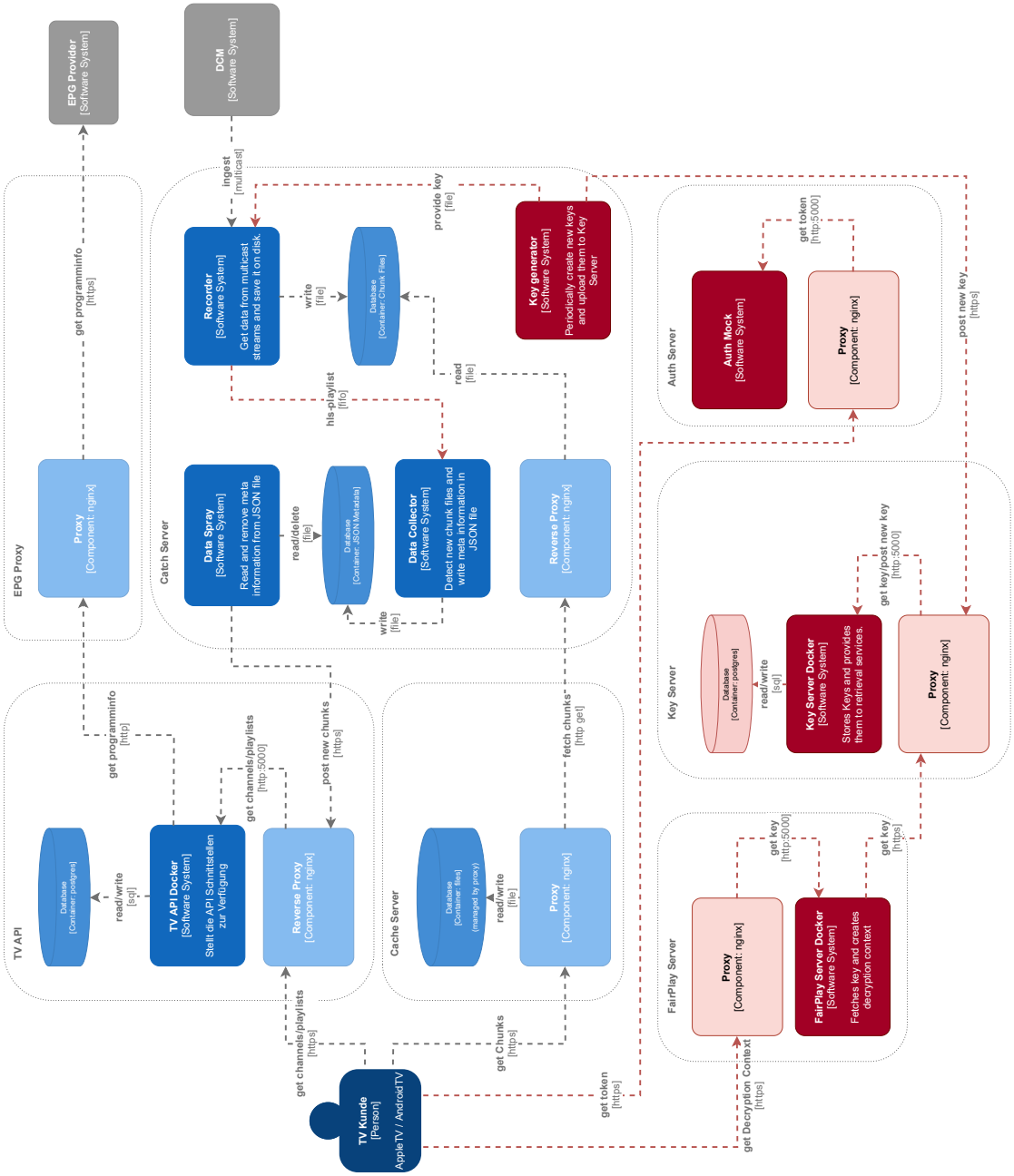
Woche	Start	Phase	Todo
1	15.02.2021	«Einarbeitung und Recherche»	IST-Konstrukt analysieren und verstehen
2	22.02.2021		Recherche zu Verschlüsslungsmöglichkeiten und Schlüsselaustausch
3	01.03.2021		Recherche zu Videosignalübertragung
4	08.03.2021		Dokumentation der Recherchierten Themen in BA
5	15.03.2021	«System Requirements und Threat Modelling und Risk Analysis»	Definieren von System Requirements und Verbessern mit Threat Modelling
6	22.03.2021		Risk Analysis durchführen und dokumentieren
7	**29.03.2021**	«Code»	Einrichten der Umgebung / Modul Mockup in Testumgebung einbinden
8	05.04.2021		Code and Test
9	12.04.2021		Code and Test
10	19.04.2021		Code and Test
11	**26.04.2021**	«Risk Analysis»	Schwachstellensuche im Prototyp
12	03.05.2021		Beheben oder Dokumentation was verbessert werden könnte
13	10.05.2021		Risk Analysis nachführen
14	17.05.2021	«Doku»	Doku
15	**24.05.2021**		Doku
16	31.05.2021		Doku
17	07.06.2021	«Reserve und Refining»	Abschluss

7.2 Diagramme und Tabellen

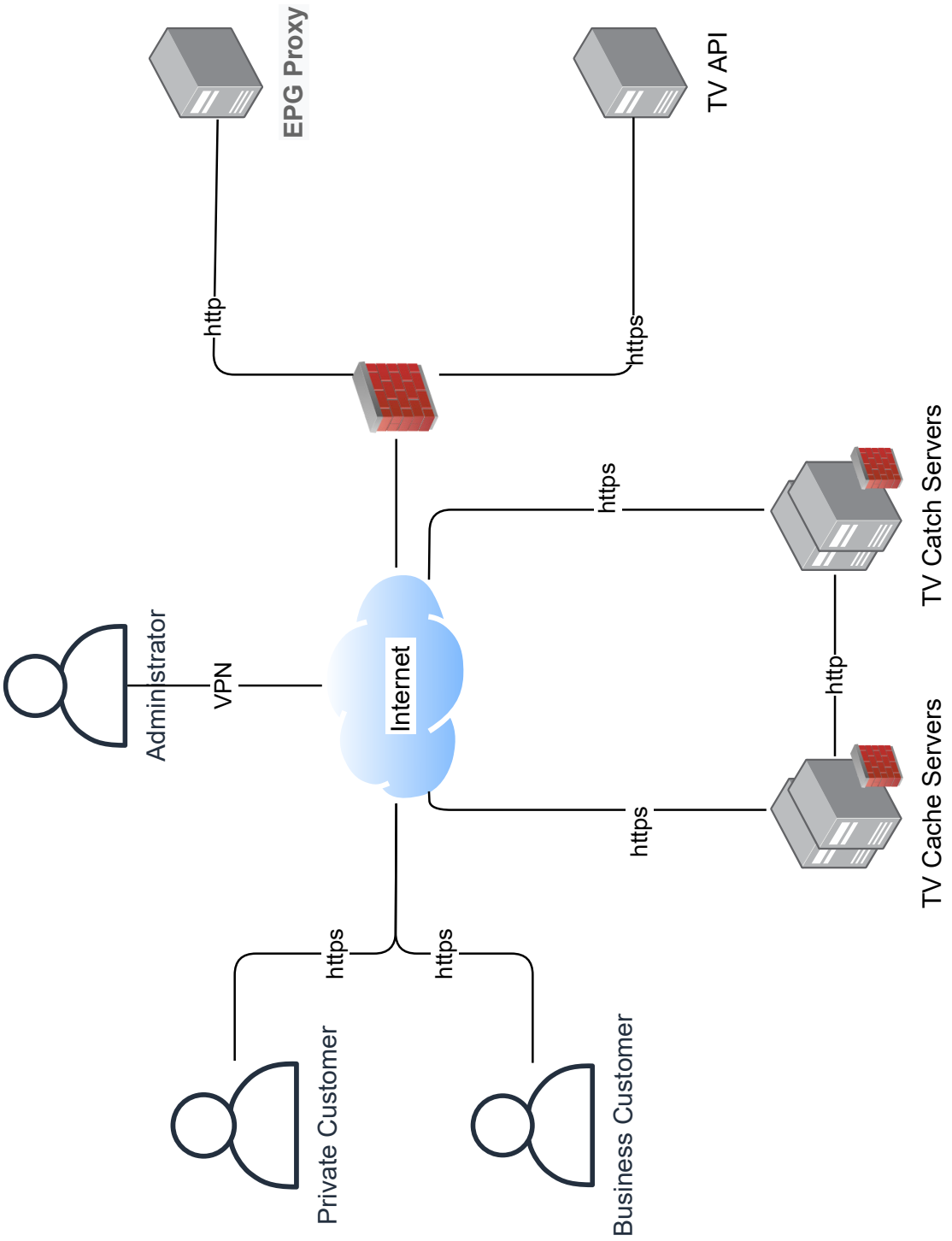
7.2.1 Bestehende Backend-Architektur der Init7



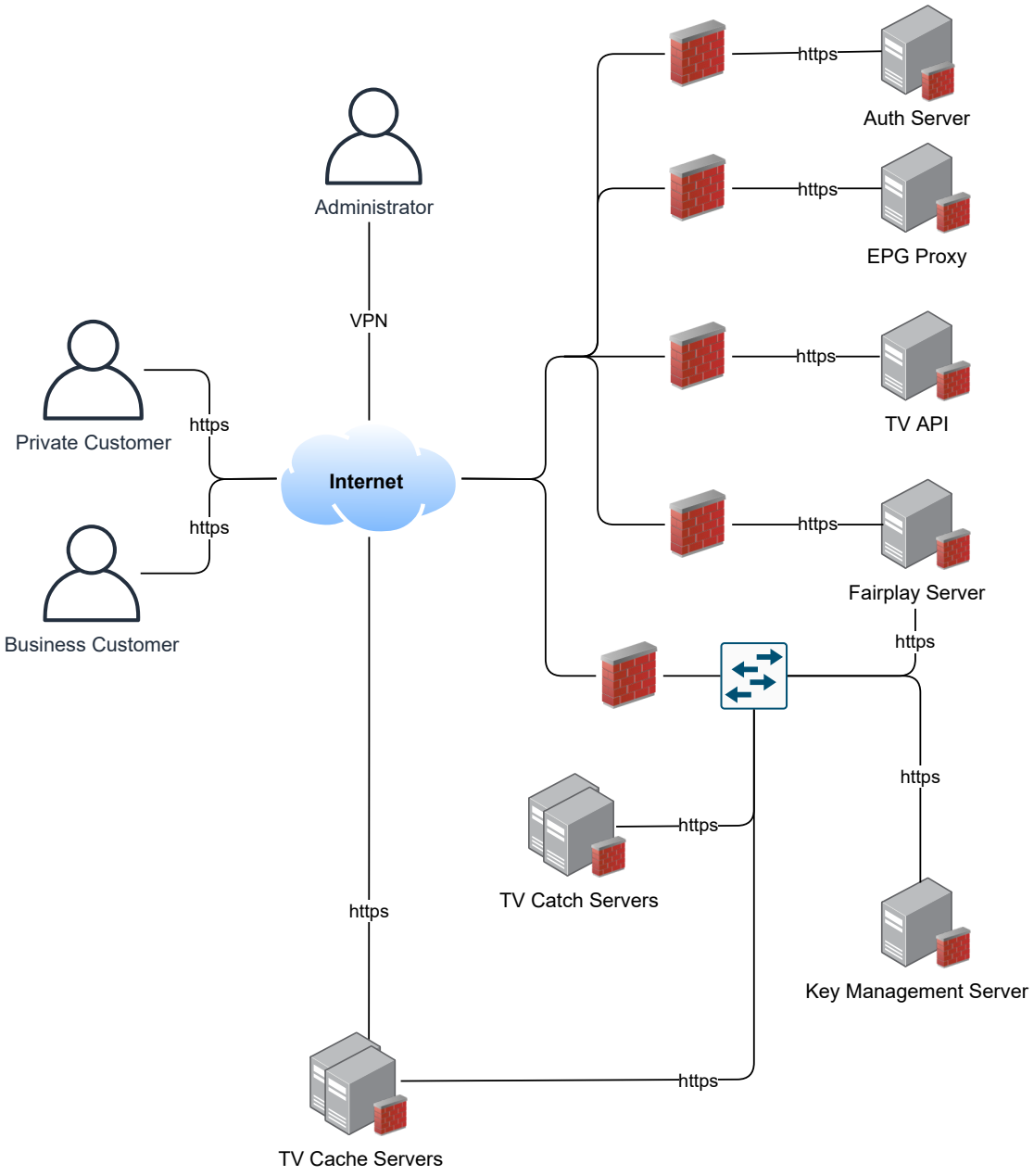
7.2.2 Zukünftige Backend-Architektur



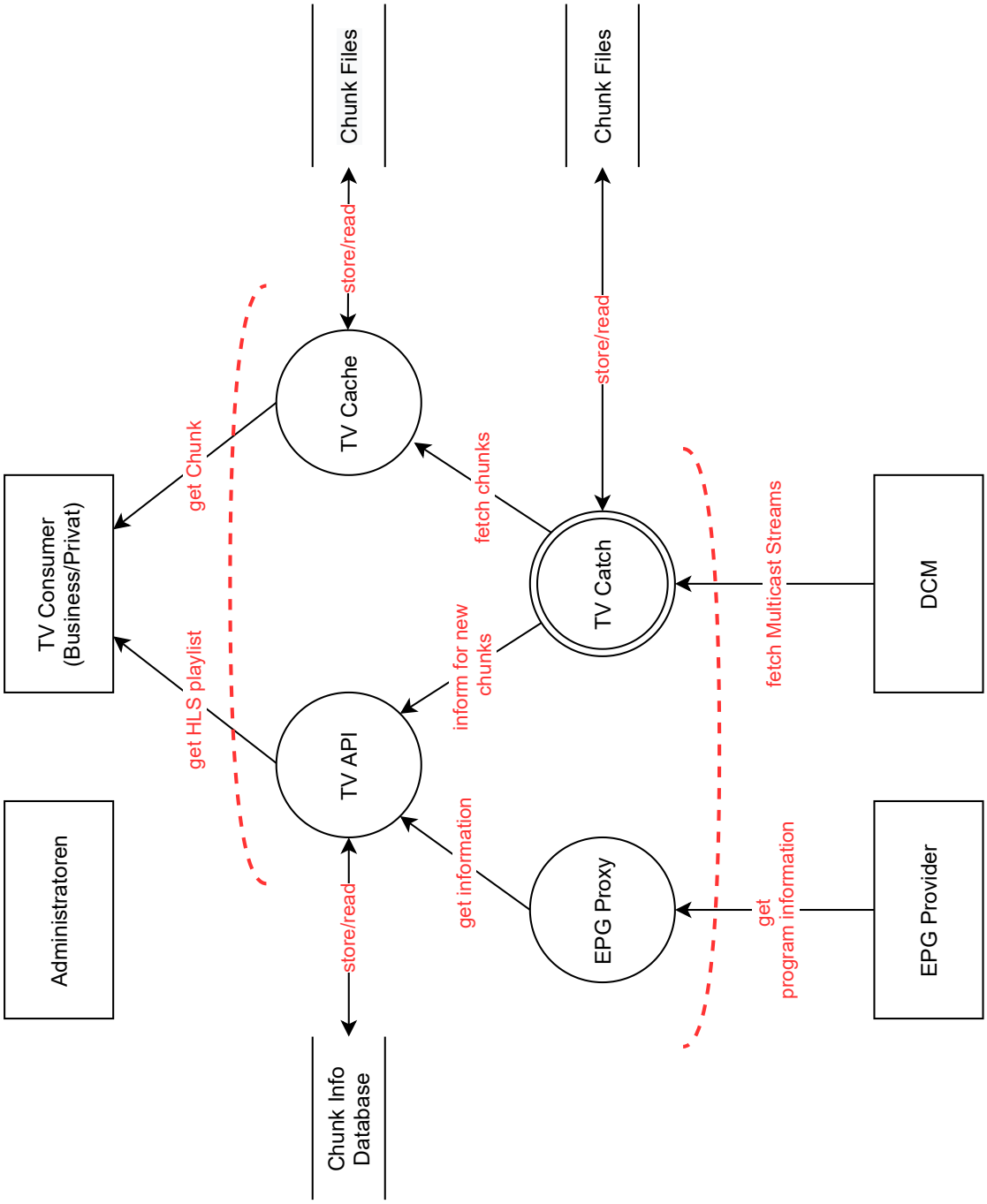
7.2.3 Aktuelles Netzwerkdiagramm Init7



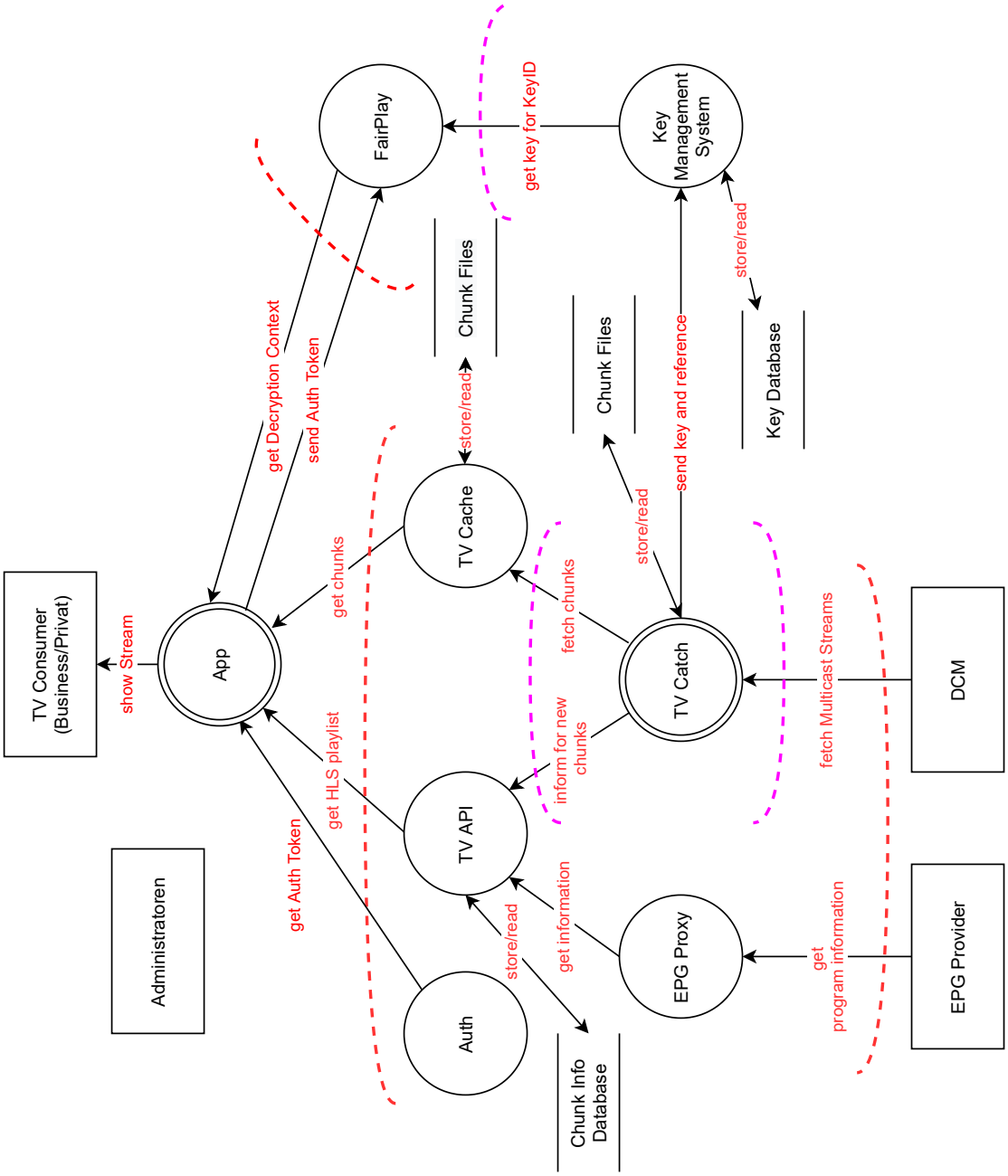
7.2.4 Zukünftiges Netzwerkdiagramm



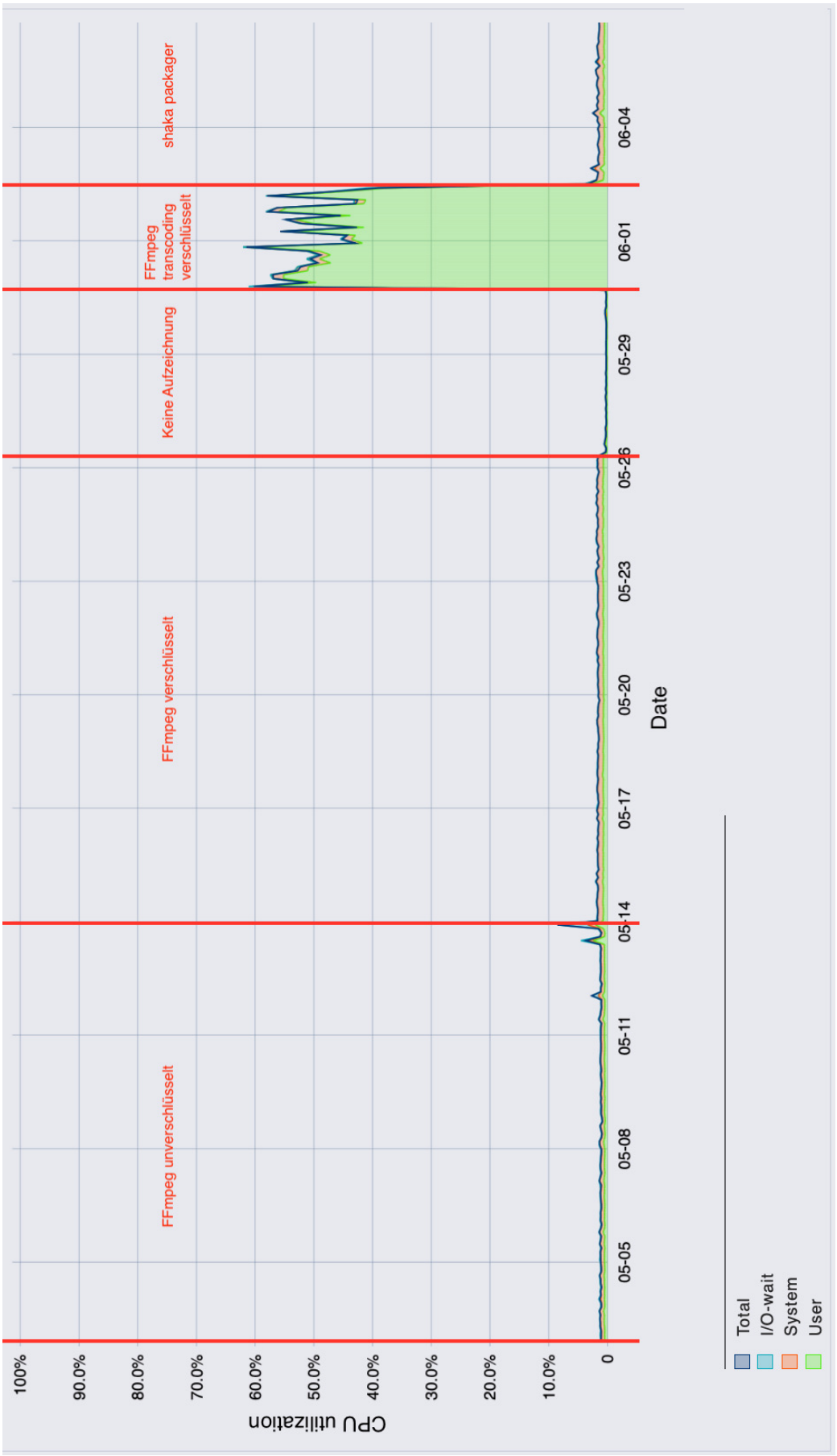
7.2.5 Aktuelles Data Flow Diagram (DFD)



7.2.6 Zukünftiges Data Flow Diagram (DFD)



7.2.7 CPU-Leistungsgraph des Catch-Servers



7.2.8 TV-Kanäle

Tabelle 7.2: Senderangebot der Init7 (Schweiz) AG (exportiert am 3. März 2021)

2M Maroc	Disney Channel	Oman TV	SPORT1 HD CH
3+ HD	DMAX HD	one HD	SRF Info HD
3Sat HD	Dubai Sports 1	One TV	SRF1 HD
4+ HD	Dubai TV	ORF1 HD	SRFzwei HD
4E	Duna TV	ORF2 HD	Star TV HD
5 USA	DW Europe	PCNE Chinese	STERK TV
5+ HD	E4	Phoenix HD	Super RTL (Schweiz)
5STAR	EuroNews	ProSieben (Schweiz)	SWISS1 HD
6+ HD	Euronews F	ProSieben Maxx	SWR BW HD
6ter	Eurosport	Puls 8 HD	tagesschau24 HD
Abu Dhabi TV	Fashion TV	Qatar TV	Tele 5
Al Jazeera English HD	FB TV	Rai 1 HD	Tele Bielingue HD
Al Sharqiya	Film 4	Rai 4	Tele M1 HD
al-Arabiya	France 2 HD	Rai Due	Tele Ticino HD
Alsace 20	France 24	Rai Movie	TELE TOP HD
Anixe HD	France 24 E	Rai News	Télé Versoix
ARD Alpha	France 3	Rai Premium	Tele Z HD
Arirang World	France 4	Rai Sport+ HD	Tele Züri HD
Arte F HD	France 5	Rai Storia	Tele1 HD
Arte HD	Gulli	Rai Tre	TeleBärn HD
BBC Four HD	HABERTURK	Rai Yoyo	TeleBasel HD
BBC NEWS HD	Helvetia One TV	rbb Berlin HD	TeleD HD
BBC One HD	hr-fernsehen HD	Real Time	TF1 HD
BBC Two HD	HSE24	RMC Story	TFX
BBC World News Europe HD	HSE24 TREND	Rotana Drama	Thai Global Network
BFM TV	IRIS	Rouge TV HD	TLC
Bibel TV HD	Italia 1 HD	RSI LA 1 HD	TMC
Bloomberg	ITV 1 HD	RSI LA 2 HD	Travel
blue Zoom	ITV 2	RTK 1	TRT 3-SPOR
BN	ITV 3	RTL 2 (Schweiz)	TRT TURK
BR HD	ITV 4	RTL 9 HD	TTN
BVN TV	Jordan TV	RTL CH	TV Südostschweiz HD
C1R Europe	Kabel1 (Schweiz)	RTL Nitro	TV Suisse Plus
C8 HD CH	KANAL 7 AVRUPA	RTP 1	TV24 HD
Canal 24 Horas	Kika HD	RTRS plus	TV25 HD
Canal Alpha HD	Kuwait TV	RTS 1 HD	TV5MONDE EUROPE
Canal Alpha JU HD	LA 7	RTS 2 HD	TVE Internacional
Canale5 HD	la télé HD	RTSH 3	TVM3
CBBC HD	LFM TV	RTSH Sat	TVO HD
CBeebies HD	M6 HD	Russia Today	TVP Polonia
CCTV4	MBC Europe	S1 HD	TVR International
Channel 4 HD	MDR Sachsen HD	Sama Dubai	VOX (Schweiz)
Channel 5 HD	More4+1	Sat.1 (Schweiz)	W9 HD
CITV	MTV Schweiz HD	Sat1 Gold	WDR HD Köln
CNBC Europe	MySports HD D	Saudi 1	Welt der Wunder
CNN International	MySports HD F	Schweiz 5	Welt HD
Comedy Central CH HD	n-tv	Servus TV HD	wetter.tv
Das Erste HD	NDR HD	Sharjah TV	ZDF HD
DELUXE MUSIC	Nickelodeon CH HD	SIXX	ZDF Info HD
Die Neue Zeit TV	NRJ12	Sky News	ZDF Neo HD

7.3 Weitere Dokumente

7.3.1 Threat Model

In der Arbeit wurde ein Threat Model erstellt. Mithilfe dieser Analyse konnten die Security Requirements ermittelt werden. Die vollständige Analyse befindet sich auf den nächsten Seiten.

THREAT MODELLING (BEDROHUNGSANALYSE)

UNTERNEHMENS- UND SECURITY-ZIELE

Unternehmensziel:

Die TV-Plattform erlaubt es den Kunden Live und Replay Fernsehsendungen im HD-Format anzuschauen.

Security-Ziele:

- Der Stream kann nicht kopiert oder aufgezeichnet werden.
- Nur Kunden des TV-Anbieters können die Inhalte schauen.
- Nach Ablauf einer Frist können die Inhalte nicht mehr wiedergeben werden.
- Zu üblichen Nutzungszeiten ist die Konsumation der Inhalte möglich

ALLGEMEINE INFORMATIONEN

Welche Funktionen stellt das System zu Verfügung?	Es werden TV-Inhalte zur Verfügung gestellt. Diese werden dem Konsumenten verschlüsselt übertragen. Das System ist auch für die Distribution der Entschlüsselungsinformationen verantwortlich.
Wer kann auf das System zugreifen?	Auf das System haben die Kunden des Anbieters Zugriff. Dabei wird beim Anbieter zwischen **Privatkunden** und **Businesskunden** unterschieden. Die **Administratoren**, welche beim Anbieter angestellt sind, haben ebenfalls Zugriff auf das System.
Was verarbeitet das System?	Es werden TV-Inhalte verarbeitet. Zu diesen Inhalten zählen Audio- und Videosignale. Es werden ausserdem Programminformationen und kryptografische Materialien verarbeitet.
Wie arbeitet das System?	Diese Frage wird in unserem Architektur Teil beantwortet.
Was für externe Abhängigkeiten hat das System?	Das TV-Signal wird von UPC und GibSolutions empfangen. Die Programminformationen werden von EPG.Best abgerufen. Als Internetanbieter fungiert Init7.
Welches Service-Level muss das System erfüllen?	Das System ist nur in der Schweiz verfügbar. Es soll nicht Hochverfügbar sein, jedoch soll es zu den üblichen Nutzungszeiten verfügbar sein. (16:00 Uhr – 01:00 Uhr)
Welche Security Requirements oder Security Controls wurden bereits definiert?	- Verfügbarkeit nur im internen Netzwerk möglich. - Externe Verbindungen nur mit HTTPS - Die Webapplikation darf nur gesichert kommunizieren (TLS 1.0 and 1.1) - Die Applikation unterstützt nur sichere Verschlüsselungen (AES mit Schlüssellänge ≥ 128 Bits, kein MD5).

VERMÖGENSWERTE DES SYSTEMS

Das zu untersuchende System beinhaltet folgende Vermögenswerte:

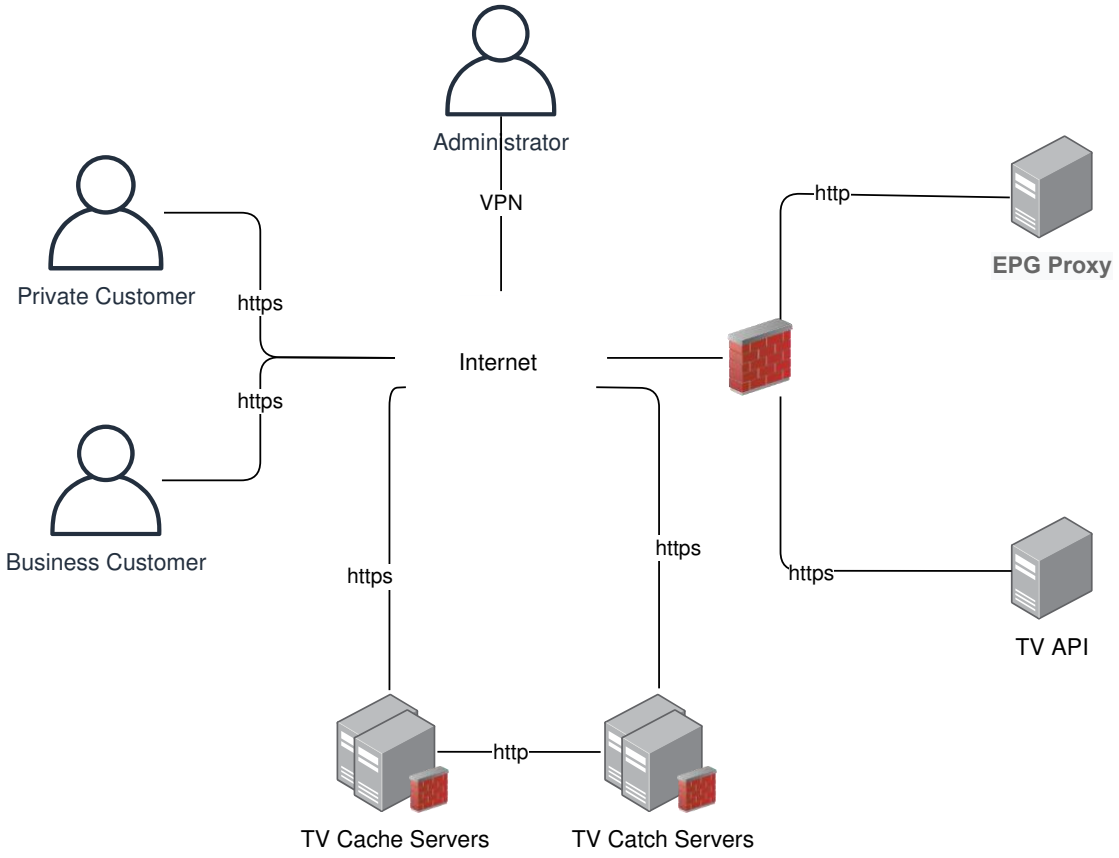
- **Serverseitige Systeme** (Integrität von Systemen und Daten)
- **Chunks und Keys** (Daten)
- **Stream- und Key-Austausch** (Prozesse)
- **Authentifizierungsprozess**
- **Anmeldedaten von Benutzern** (IP-Range)
- **Anmeldedaten des Administrators**
- **Logs**

ANALYSE UND ZERSETZUNG DES SYSTEMS

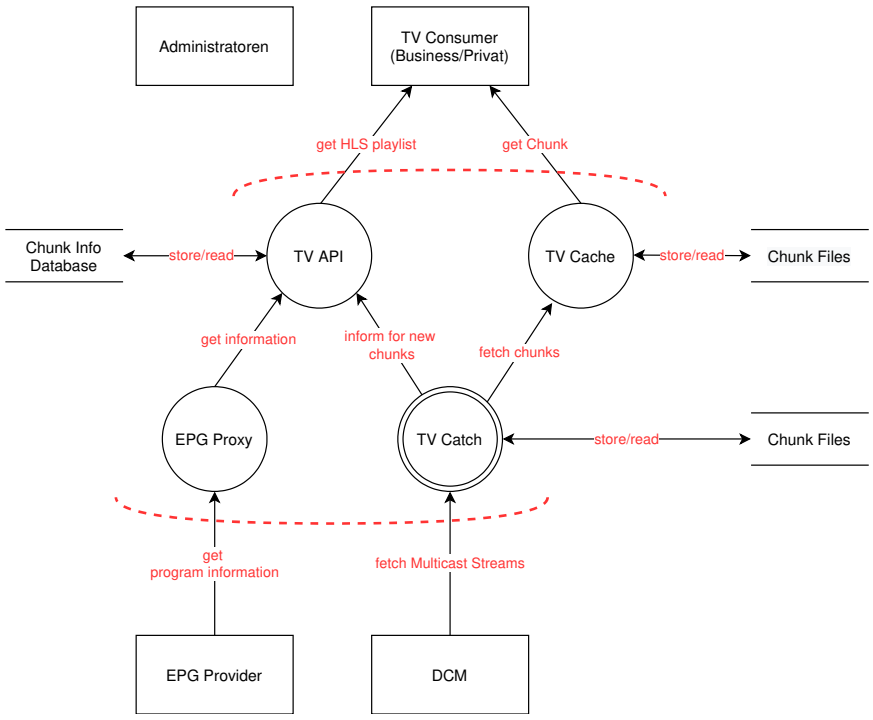
Das System existiert bereits in einem unverschlüsselten Zustand. Wir wollen die Sicherheitslücken des IST-Zustands ermitteln, damit wir auch Verbesserungen der Bestehenden Infrastruktur ermitteln können. In einem zweiten Schritt haben wir noch die von uns erschaffene Architektur analysiert und zersetzt.

IST-ZUSTAND

Den IST-Zustand haben wir mit Informationen aus dem GitLab des bestehenden Codes und Mitarbeitern des TV-Anbieters ermittelt. Der momentane Netzwerkaufbau haben wir in der untenstehenden Grafik visuell dargestellt. Die Cache und Catch Server stehen direkt im Internet und werden jeweils mit einer IPTables Firewall geschützt. Es existieren jeweils mehrere Catch und Cache Server. Die TV API und der EPG Proxy Server werden durch eine virtuelle Firewall mit dem Internet verbunden. Die Benutzer greifen über https auf die TV API und die Cache Server zu.

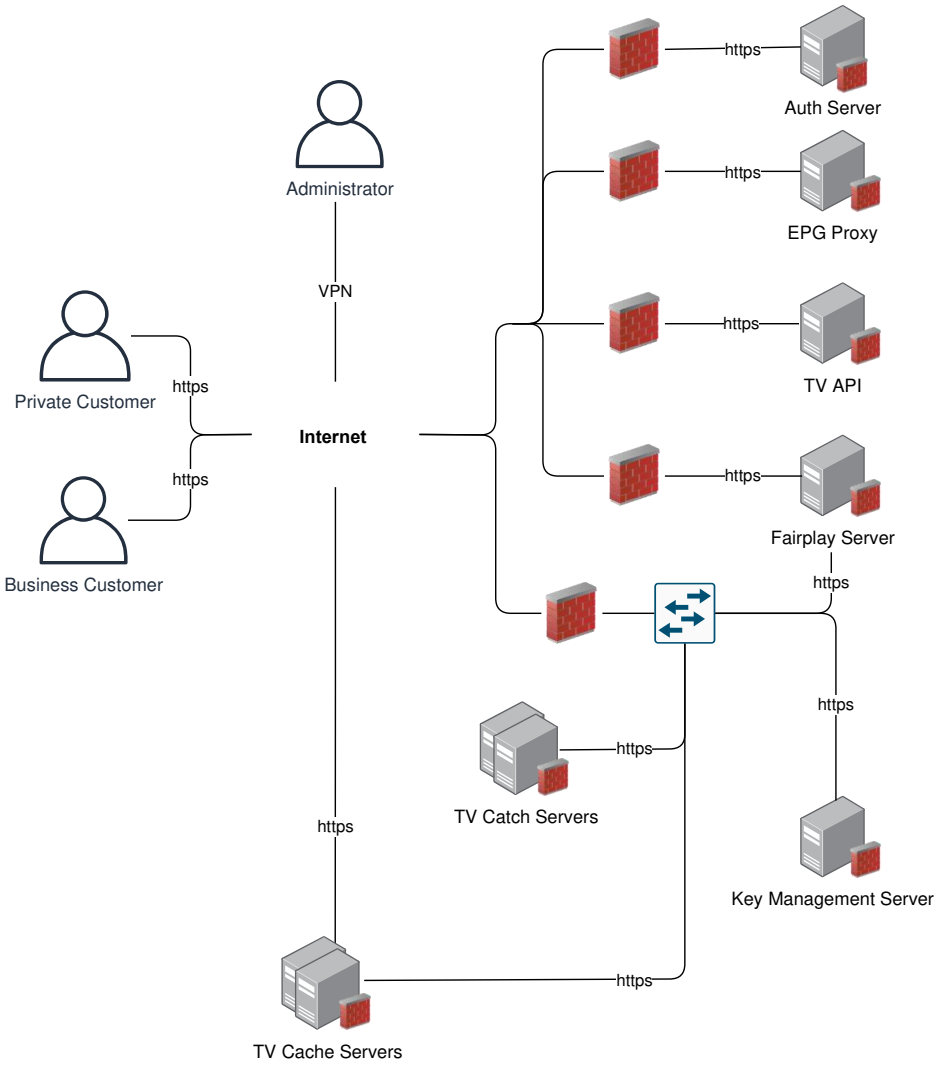


Das Data-Flow-Diagramm konnten wir mithilfe von Informationen aus einer bestehenden Architekturgrafik und der Analyse des Codes erstellen. In dieser Konfiguration ist es dem TV-Konsumenten mit jedem Gerät und Software möglich, den TV-Stream zu schauen. Wie in der Grafik ersichtlich ist, greift er dazu auf die TV API und die Cache Server zu. Diese werden mit Informationen von den Catch Servern und dem EPG Proxy versorgt. Die TV-Inhalte werden von den Multicast Streams der DCM Server geholt. Die Administratoren haben auf alle Services und Server Zugriff. Mit den roten gestrichelten Linien haben wir den vertrauenswürdigen Bereich markiert. Diesen haben wir so gewählt, da diese Bereiche komplett vom TV-Anbieter verwaltetet werden. Sowohl die Server, das Netzwerk und die Software werden vom Anbieter zur Verfügung gestellt.

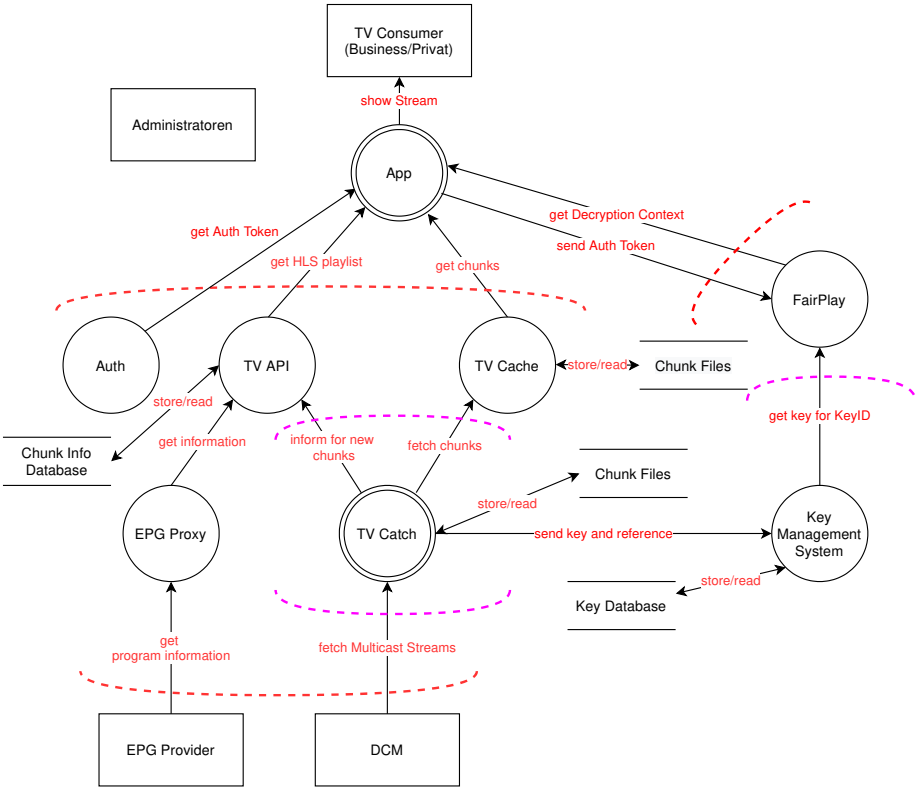


SOLL-ZUSTAND

Im Soll-Zustand haben wir unsere zusätzlichen Server für die neue Architektur eingebunden und auch Anpassungen für die Sicherheit des IST-Zustands integriert. Zu den wichtigsten Änderungen gehört die Umstellung der gesamten Netzwerkverkehrs auf https. Ausserdem ist es nicht notwendig, dass die Catch Server aus dem Internet erreichbar sind. Diese würden wir deshalb zusammen mit dem Key Management Server in ein separiertes Netzwerk nehmen, welches den Zugriff von aussen nicht zulässt. Wie in der untenstehenden Grafik ersichtlich ist, hätten zusätzlich der FairPlay Server und auch die Cache Server Zugriff in dieses abgeschottete Netzwerk. Diese Server sind auf die Kommunikation mit den Catch Server oder dem Key Management Server angewiesen. Eine weitere Änderung ist, dass alle Server mit der IPTables Firewall abgesichert werden sollen.



Im Datenflussdiagramm der neuen Lösung, haben wir die zusätzlichen Services integriert. Dazu gehören der Auth Server, welcher die Authentifizierungstokens ausstellt und die gesamte Key Management Infrastruktur. Eine Neuerung ist ebenfalls, dass der Stream nur noch mit der App vom TV Anbieter angeschaut werden können. Mit der violetten gestrichelten Linie haben wir einen tiefer liegenden Vertrauensbereich geschaffen. Diese Datenflüsse finden im abgeschotteten Netzwerk statt und haben nur mit dem Schlüsselaustausch zu tun.



Threat Agents (Bedrohungsakteure)

Agents	Ziel
Script Kiddies	Sie wollen prüfen, ob sie in der Lage sind das System zu hacken. Ihr einziges Ziel ist Spass.
Mitarbeiter	Ehemalige Mitarbeiter möchten sich rächen oder das System sabotieren, da sie ihren Job verloren haben. Oder sie wollen Profit machen. Ziel: finanzieller Gewinn oder Sabotage.
Cyber Criminal	Sie wollen Malware ins System einschleusen und an User verbreiten. Oder sie möchten Daten verschlüsseln. Ziel: finanzieller Gewinn.
Konkurrenz	Sie wollen das System sabotieren, dem Image schaden und dadurch Kunden zum Wechsel des Anbieters bewegen. Ziel: finanzieller Gewinn und Imageschaden

Unwahrscheinliche Bedrohungsakteure sind: Staaten (keine Motivation), Hacktivisten (keine politische Motivation).

THREATS (BEDROHUNGEN)

Anhand der STRIDE-Methode wurde für jede Kategorie unseres Systems aufgelistet, was schiefgehen kann.

Nr.	Element	Kat.	Beschreibung	Nutzen für Angreifer	Req	Realistisch
T1	TV Konsument	S	Angreifer kann Stream über VPN Technologie an weitere Personen weitergeben.	Unbefugte Personen können den Stream schauen.	R15, R38	Ja
T2		R	Angreifer kann Stream über VPN Technologie an weitere Personen weitergeben.	Unbefugte Personen können den Stream schauen.	-	Nein
T3	DCM	S	Ausgeben als Provider	Kann System mit gefälschten Streams füllen	R10	Nein
T4		R	Ausgeben als Provider	Kann System mit gefälschten Streams füllen	-	Nein
T5	EPG Provider	S	Ausgeben als EPG Proxy (IP)	Angreifer kann EPG Daten erhalten ohne zu bezahlen	R15	Nein
T6		R	Ausgeben als EPG Proxy (IP)	Angreifer kann EPG Daten erhalten ohne zu bezahlen	-	Nein
T7	Get Auth Token	T	Token ersetzen/kaputt machen	Kunde kann kein verschlüsseltes TV schauen (DoS)	R13, R14	Nein
T8		I	Angreifer beschafft sich ein Token.	Angreifer kann verschlüsseltes TV schauen.	R13, R14, R21, R30, R33	Ja
T9		D	DoS auf Auth Server Kommunikation	Kunden können sich nicht mehr authentifizieren.	R3, R4, R5, R6, R11,	Nein
T10	Get HLS Playlist	T	Anpassen der Pfade zu Chunk Files und Keys. Übernahme des Streams	Kunde bekommt falschen Stream. Kunde kann beeinflusst werden. Mediashop Attack	R13, R14	Nein
T11		I	Angreifer ergaunert sich eine oder mehrere Playlisten.	Er kann dadurch die Pfade zu den Chunk-Dateien ermitteln. Ausserdem erhält er Informationen über den Schlüsselwechsel.	R10, R13, R14, R20, R21, R28	Ja
T12		D	DoS auf TV API-Kommunikation	Kunden können kein TV schauen	R5	Nein
T13	Get Chunk	T	Chunks können manipuliert werden.	Kunden können kein TV schauen	R13, R14	Nein
T14		I	Chunks könnten abgegriffen werden.	Dadurch könnte der Angreifer die Chunk Inhalte schauen.	R13, R14, R20,	Ja

					R21, R28	
T15		D	DoS auf Cache-Kommunikation	Kunden können keine Chunks mehr abrufen und können kein TV mehr schauen	R5	Nein
T16	Get Decryption Context	T	Modifizieren des Decryption Context	Kunden können kein verschlüsseltes TV schauen	R13, R14, R18	Nein
T17		I	Decryption Context könnte von einem Angreifer mitgehört werden.	Der Angreifer hätte so Zugriff auf einen oder mehrere Entschlüsselungsschlüssel.	R13, R14, R21, R28, R29, R31, R32	Ja
T18		D	DoS auf Key-Kommunikation	Kunden können keine Schlüssel mehr abrufen und können kein verschlüsseltes TV mehr schauen	R5	Nein
T19	Store/read chunks	T	Chunks können manipuliert werden.	Kunden können kein TV schauen, falsche Informationen bekommen	R13, R14, R18	Nein
T20		I	-			Nein
T21		D	DoS auf Storage	Für einige Sender können Kunden keine Chunks mehr abrufen und kein TV mehr schauen	R5, R18	Nein
T22	Fetch chunks	T	Chunks können manipuliert werden.	Kunden können kein TV schauen, falsche Informationen bekommen	R13, R14, R18	Nein
T23		I	Chunks werden bei der Übertragung mitgeschnitten.	Anreifer kommt an die Chunks und kann die Videostreams schauen	R13, R14, R18, R20, R28	Nein
T24		D	DoS auf Cache/Catch-Kommunikation	Für einige Sender können Kunden keine Chunks mehr abrufen und kein TV mehr schauen	R5, R18	Nein
T25	Inform for new Chunks	T	Chunks können manipuliert werden.	Kunden können kein TV schauen, falsche Informationen bekommen	R12, R13, R14, R28	Nein
T26		I	Angreifer kann sich Informationen über neue Chunks holen	Angreifer kann sich selbst eine Playlist zusammenbauen	R13, R14, R18, R20	Nein
T27		D	DoS auf TV-Spray, verhindern der Kommunikation von catch zu TV API	Für einige Sender können Kunden keine Chunks mehr abrufen und kein TV mehr schauen	R5, R18	Nein

T28	Get Information from EPG Proxy	T	EPG kann manipuliert werden.	Replay kann unmöglich werden, falsche Programminformationen beim Kunden	R13, R14, R10	Nein
T29		I	EPG könnte vom Angreifer abgegriffen werden.	EPG könnte ohne Bezahlung abgerufen werden.	R13, R14, R18, R28	Nein
T30		D	DoS auf EPG Proxy	Kein direkter Effekt	R5	Nein
T31	Send key and reference	T	Senden falscher Keys.	Kunden können kein verschlüsseltes TV schauen	R12, R13, R14, R18, R28	Nein
T32		I	Key und KeyID könnten von einem Angreifer mitgehört werden.	Der Angreifer hätte so Zugriff auf einen oder mehrere Entschlüsselungsschlüssel.	R13, R14, R18, R29, R31, R32	Ja
T33		D	Verhindern des senden der Key-Informationen	Für einige Sender können Kunden keine Keys beziehen und bis repariert kein TV schauen	R1, R5, R18	Nein
T34	Store/read Key	T	Senden falscher Keys.	Kunden können kein verschlüsseltes TV schauen	R12, R28	Nein
T35		I	-	-	-	Nein
T36		D	Verhindern des Zugriffs auf die Key-Datenbank	Kunden können keine Schlüssel mehr abrufen und können kein verschlüsseltes TV mehr schauen	R3, R5	Nein
T37	Get key for KeyID	T	Senden falscher Keys.	Kunden können kein verschlüsseltes TV schauen	R12, R13, R14, R18, R28	Nein
T38		I	Key und KeyID könnten von einem Angreifer mitgehört werden.	Der Angreifer hätte so Zugriff auf einen oder mehrere Entschlüsselungsschlüssel.	R13, R14, R18, R29, R31, R32	Ja
T39		D	Verhindern der Kommunikation mit dem Key Server	Kunden können keine Schlüssel mehr abrufen und können kein verschlüsseltes TV mehr schauen	R5, R18	Nein
T40	Fetch Multicast Stream	T	Sende gefälschten MC-Stream	Kunden erhalten falsche Informationen.	R12, R18	Nein
T41		I	TV-Multicast könnte von Angreifer abgegriffen werden.	Angreifer kann kostenlos TV schauen.	R10	Nein

T42		D	DoS auf Multicast Stream Kommunikation. Ausfall beim Provider	Es können keine Chunk Files mehr generiert werden. Dadurch gibt es kein Live TV und keine Aufzeichnung zu diesem Zeitpunkt	R2, R5	Ja
T43	Get program information	T	Bereitstellen von gefälschten EPG Daten	Replay kann unmöglich werden, falsche Programminformationen beim Kunden	R13, R14	Nein
T44		I	EPG könnte vom Angreifer abgegriffen werden.	EPG könnte ohne Bezahlung abgerufen werden.	R13, R14, R18, R28	Nein
T45		D	DoS auf Kommunikation zum EPG Provider oder Ausfall des EPG Providers	Es können keine Programminformationen mehr zur Verfügung gestellt werden. Keine grosse Auswirkung	R2, R5	Ja
T46	Chunk Files	T	Chunks können manipuliert werden.	Kunden können kein TV schauen, falsche Informationen bekommen	R12, R13, R14, R18, R16, R17, R35	Nein
T47		R	Chunks können manipuliert werden.	Kunden können kein TV schauen, falsche Informationen bekommen	R39, R40	Nein
T48		I	Angreifer kann Chunks auslesen.	Der Angreifer kann sich einen Stream zusammenbauen.	R11, R12, R15, R16, R17, R19, R20	Nein
T49		D	Ausfall der Disks	Es ist nicht möglich TV zu schauen.	R7	Nein
T50	Key Database	T	Anpassen der Keys oder KeyIDs.	Kunden können kein verschlüsseltes TV schauen	R12, R16, R17, R18	Nein
T51		R	Unbemerktes anpassen der Keys oder KeyIDs.	Angreifer kann TV-Wiedergabe verunmöglichen.	R39, R40, R41	Nein
T52		I	Key und Key ID können ausgelesen werden	Angreifer kann verschlüsseltes TV schauen	R11, R12, R15, R16, R17, R18, R19, R34, R35, R36	Ja

T53		D	Ausfall der Datenbank oder erhöhtes Anfrageaufkommen	Der Zugriff auf die Datenbank ist nicht mehr möglich. Die Schlüssel können nicht mehr gespeichert werden und es kommt zu einem Ausfall beim Recording.	R1, R7, R18	Nein
T54	Auth Server	S	Durch IP-Spoofing kann man ein Token erhalten	Der Angreifer hat dadurch Zugriff auf das TV System und kann die Inhalte konsumieren, obwohl er kein Kunde ist.	R10	Nein
T55		T	Code anpassen und Authentifizierung aushebeln oder verschärfen.	Jeder kann konsumieren oder keiner kann konsumieren.	R5, R8, R9, R11, R12, R16, R17, R18	Nein
T56		R	Angreifer kann unbemerkt ein Token fälschen oder den Code verändern.	Er kann auf alles zugreifen oder machen was er will.	R39, R40, R42	Nein
T57		I	Signierungsschlüssel kann geklaut werden.	Ein Angreifer kann sich selbst Tokens ausstellen.	R11, R12, R16, R17, R18, R19, R29, R34, R35, R36	Ja
T58		D	DoS auf Auth Server/Software	Kunden können sich nicht mehr authentifizieren.	R3, R4, R5, R6	Ja
T59		E	Broken Access Control oder fehlerhafte Algorithmen könnten dazu führen, dass sich ein Angreifer ein Access Token generieren kann, obwohl er keine Berechtigung dazu hat.	Der Angreifer hat dadurch Zugriff auf das TV System und kann die Inhalte konsumieren, obwohl er kein Kunde ist.	R5, R8, R9	Nein
T60	TV API	S	Gestohlenes Catch-Token IP Spoofing	Möglichkeit Chunk/Sender und EPG Daten zu modifizieren Zugriff auf Streams ohne Kunde zu sein	R10, R21, R20	Nein
T61		T	Modifizieren der Chunk-Informationen und Key Referenzen	Flasche Streams bereitstellen, Streams löschen, Verschlüsselung aufheben, Verschlüsselung erzwingen.	R5, R8, R9, R11, R12, R16, R17	Nein

T62		R	Angreifer kann unbemerkt den Code/Daten verändern.	Er kann unbemerkt falsche Streams verbreiten. (Mediashop Attack)	R39, R40, R41	Nein
T63		I	Kann Meta-Information über Chunks und Keys auslesen.	Angreifer kann sich seine Playlist selbst zusammenstellen.	R8, R9, R10, R11, R12, R15, R16, R17, R19, R20, R21, R35, R36	
T64		D	DoS auf TV API Server/ Software	Kunden können keine Playlists mehr herunterladen und können nicht mehr auf die TV-Inhalte zugreifen.	R3, R4, R5, R6	Ja
T65		E	Broken Access Control SSH Zugriff mit Brute Force/Exploit im SSH Server	Möglichkeit Chunk/Sender und EPG Daten zu modifizieren oder das System zu übernehmen	R5, R8, R11, R12	Nein
T66	TV Cache	S	Mit einer Man-in-the-Middle Attacke könnte sich ein Angreifer sich als Catch Server ausgeben.	Der Angreifer kann so den Inhalt des Streams anpassen und so Werbung oder andere Inhalte verbreiten.	R5, R18	Nein
T67		T	Chunks können manipuliert werden.	Kunden können kein TV schauen, falsche Informationen bekommen	R5, R8, R9, R12, R16, R17	Nein
T68		R	Angreifer kann unbemerkt die Config/Chunks verändern.	Er kann auf alles zugreifen oder machen was er will.	R39, R40, R41	Nein
T69		I	Chunk Dateien können abgerufen werden.	Der Angreifer kann TV Streams schauen.	R11, R12, R15, R16, R17, R19, R20, R21	Ja
T70		D	DoS auf TV Cache Server/ Software	Es gibt mehrere Cache Server. Wenn es einen Ausfall gibt, hat dies keine Auswirkungen. Die Cacheserver laufen redundant. Falls jedoch zwei Server mit den gleichen Inhalten ausfallen, gibt es einen Ausfall bei einem Teil der Sender.	R3, R4, R5, R6	Ja

T71		E	Mit einer Brute-Force Attacke könnte sich ein Angreifer Zugriff auf den Server verschaffen. Da keine Zugriffssteuerung existiert, kann ein Angreifer auf alle Chunks zugreifen.	Er könnte die TV-Inhalte ohne Erlaubnis konsumieren, wenn die Namen der Chunks herausfindet. Er könnte sich Adminrechte verschaffen und den Server zweckentfremden. (Cryptomining, Datenzugriff)	R5, R11, R12, R20, R21	Nein
T72	TV Catch	S	Gestohlener SSH Zugriff Man-in-the-Middle zu Key Server	Er könnte sich Adminrechte verschaffen und den Server zweckentfremden. (Cryptomining, Datenzugriff) und kann die aufgezeichneten Chunks bearbeiten. Der Angreifer kommt an das Key-Material.	R10, R16, R17, R18, R22	Nein
T73		T	Manipulation des Key-Generators, damit dieser einfache oder bekannte Keys generiert.	Der Angreifer könnte so einfach an die Schlüssel kommen. Kunden können je nach Anpassung kein TV schauen	R5, R8, R9, R11, R12, R16, R17, R18	Nein
T74		R	Angreifer kann unbemerkt die Config/Chunks verändern. Angreifer kann unbemerkt Key stehlen.	Er kann auf alles zugreifen oder machen was er will.	R39, R40, R41	Nein
T75		I	Die Schlüssel des Key Generators könnten abgegriffen werden. Chunk Dateien können abgerufen werden.	Mit den Schlüsseln und den Referenzen kann der Angreifer die Streams entschlüsseln. Der Angreifer kann an die Chunks kommen.	R11, R12, R15, R16, R17, R18, R19, R20, R21, R22, R32, R33	Nein
T76		D	DoS auf TV Catch Server/ Software	Die Catch Server laufen nicht redundant, deshalb kommt es zu einem Teilausfall bei den Sendern. Es gibt ausserdem beim Recording eine Lücke.	R3, R4, R5, R6, R18	Nein
T77		E	Mit einer Brute-Force Attacke könnte sich ein Angreifer Zugriff auf den Server verschaffen.	Er könnte sich Adminrechte verschaffen und den Server zweckentfremden. (Cryptomining, Datenzugriff, Datenlöschung)	R5, R11, R12, R18	Nein
T78	EPG Proxy	S	Gestohlener SSH Zugriff	Er könnte sich Adminrechte verschaffen und den Server	R5, R11, R15,	Nein

				zweckentfremden. (Cryptomining, Datenzugriff)	R16, R17	
T79		T	Bereitstellen von gefälschten EPG Daten oder keinen Daten	Replay kann unmöglich werden, falsche Programminformationen beim Kunden	R5, R8, R9, R11, R12, R16, R17, R18	Nein
T80		R	Angreifer kann unbemerkt die Config/EPG Daten verändern.	Er kann auf die Programm Informationen zugreifen.	R39, R40, R41	Nein
T81		I	EPG könnte vom Angreifer abgegriffen werden.	EPG könnte ohne Bezahlung abgerufen werden.	R8, R9, R11, R12, R15, R16, R17, R35	Nein
T82		D	EPG Proxy nicht verfügbar	TV API kann EPG nicht laden. Keine direkte Auswirkung.	R5	Nein
T83		E	Mit einer Brute-Force Attacke könnte sich ein Angreifer Zugriff auf den Server verschaffen.	Er könnte sich Adminrechte verschaffen und den Server zweckentfremden. (Cryptomining, Datenzugriff)	R5, R11, R12	Ja
T84	Key Management System	S	Gestohlenes Token	Manipulation Key-Material möglich	R18	
T85		T	Löschung oder Verteilung der Keys kann manipuliert werden.	Ein Angreifer könnte sich so die Keys beschaffen oder auch die Löschung verhindern. Damit könnte er alle Chunks entschlüsseln. Er kann aber auch Wiedergabe verschlüsselter Kanäle verhindern	R5, R8, R9, R11, R12, R16, R17, R18	Nein
T86		R	Angreifer kann unbemerkt den Code/Config anpassen.	Er kann den Betrieb beeinträchtigen.	R39, R40, R41	Nein
T87		I	Schlüssel und Key Referenzen könnten abgegriffen werden	Der Angreifer könnte alle Streams entschlüsseln.	R8, R9, R11 R12, R15, R16, R17, R18, R19, R32, R33, R34,	Ja

					R35, R36	
T88		D	DoS auf Key Management Server/ Software	Wenn das Key Management System ausfällt, können keine Schlüssel mehr abgerufen werden. Dadurch können keine verschlüsselte TV-Inhalte mehr konsumiert werden.	R3, R4, R5, R6, R18	Nein
T89		E	Mit einer Brute-Force Attacke könnte sich ein Angreifer Zugriff auf den Server verschaffen.	Ein Angreifer kann verschlüsselte Streams entschlüsseln. Er könnte sich Adminrechte verschaffen und den Server zweckentfremden. (Cryptomining, Datenzugriff)	R5, R8, R11, R12, R18	Nein
T90	FairPlay	S	Durch die Fälschung des Authentifizierungstokens kann sich jemand die Inhalte anschauen, ohne dass er dazu berechtigt ist.	Er kann sich die TV-Inhalte anschauen, ohne dass er dafür ein Kunde sein muss.	R5, R8	Ja
T91		T	Keys können beliebig ausgegeben oder die Ausgabe komplett verhindert werde	Ein Angreifer könnte sich so die Keys beschaffen oder auch die Löschung verhindern. Damit könnte er alle Chunks entschlüsseln. Er kann aber auch Wiedergabe verschlüsselter Kanäle verhindern.	R5, R8, R9, R11, R12, R16, R17, R18	Nein
T92		R	Angreifer kann unbemerkt die Config/Code verändern.	Er kann den Betrieb beeinträchtigen.	R39, R40, R41	Nein
T93		I	Angreifer kann sich die Authentifizierung für den Keyserver holen.	Der Angreifer könnte alle Streams entschlüsseln.	R8, R9, R11, R12, R15, R16, R17, R19, R32, R33, R34, R35, R36	Ja
T94		D	FairPlay Server nicht verfügbar	Kunden, welche auf FairPlay angewiesen sind (Apple) können kein verschlüsseltes TV schauen.	R3, R4, R5, R6	Ja
T95		E	Broken Access Control Mit einer Brute-Force Attacke könnte sich ein Angreifer Zugriff auf den Server verschaffen.	Ein Angreifer kann verschlüsselte Streams entschlüsseln.	R5, R8, R11, R12	Nein

				Er könnte sich Adminrechte verschaffen und den Server zweckentfremden. (Cryptomining, Datenzugriff)		
T96	Administratoren	S	Man-in-the-Middle / Social Hacking / Phishing	Die Kennwörter für die Server können so herausgefunden werden. Mit diesen Informationen ist der Zugriff auf das gesamte System möglich. Ausserdem könnte sich jemand als Chef oder System ausgeben und beim Administrator Informationen holen. (Phising, Social Hacking)	R13, R14, R16, R17, R19	Ja
T97		R	Social Hacking/ Phishing	Kann mit den Zugriffsdaten Streams stehlen und den Betrieb beeinträchtigen.	R39, R40, R41, R43	Nein
T98	App	S	App mit ähnlichen Namen im App Store aufschalten	Gewisse Benutzer könnten das falsche App herunterladen.	R23, R26	Ja
T99		T	Manipulation des Apps durch Updates	Defektes App oder Kontrolle übernehmen.	R23, R25	Nein
T100		R	Ein Entwickler kann bösartigen Code veröffentlichen.	Er kann dadurch den Betrieb beeinträchtigen oder Kunden angreifen.	R9, R42	Nein
T101		I	Angreifer kann an Schlüssel und Streams kommen.	Angreifer kann Streams aufzeichnen.	R3, R27, R37	Ja
T102		D	App store dazu bringen, App zu löschen (Copyright claim?)	Kunden können App nicht erhalten	R24	Nein
T103		E	Root Rechte auf dem Gerät holen.	Dadurch kann der Angreifer auf Daten vom App und DRM System zugreifen. Mögliche Entschlüsselung	R27	Ja
T104	Show Stream	T	Man-in-the-Middle	Kann App mit gefälschten Streams füllen (Schnauz auf Fernseher)	R13, R14	Nein
T105		I	Abfilmen des Displays	Angreifer kann Stream aufzeichnen.	-	Ja
T106	Send Auth Token	T	Der Angreifer könnte sich so ein unendliches Auth Token generieren. Oder eine andere IP-Adresse angeben.	Nutzer können Streams schauen, die sie nicht dürfen oder können gewisse verschlüsselte Streams nicht schauen.	R13, R14, R29, R30	Ja
T107		I	Der Angreifer kann sich ein Token ergaunern.	Dadurch kann der Angreifer unerlaubt Streams abrufen.	R13, R14, R30	Ja
T108		D	Kommunikation mit Tokens kann überlastet werden.	Kein Kunde kann sich mehr authentifizieren.	R3, R4, R5,	Nein

					R6, R7	
T109	Chunk Info Database	T	Chunk Metadaten könnten verändert werden.	Dadurch ist die Konsumation von TV Streams nicht mehr möglich oder kann beeinflusst werden. (Mediashop Attack)	R11, R12, R15, R16, R17, R18, R19, R35, R36	Nein
T110		R	Unbemerktes anpassen der Chunk- oder EPG-Informationen.	Angreifer kann falsche Informationen verbreiten.	R39, R40, R41	Nein
T111		I	Chunk Metadaten können abgerufen werden.	Der Angreifer kann sich selbst eine Playlist zusammenstellen.	R8, R9, R11, R12, R15, R16, R17, R19, R20, R21, R35, R36	Nein
T112		D	Datenbank nicht verfügbar	Kunden können keine Playlists mehr herunterladen und können nicht mehr auf die TV-Inhalte zugreifen.	R3, R4, R5, R6	Ja

MITIGATION (SCHADENSBEGRENZUNG)

Nachfolgend werden die Massnahmen beschrieben, die die obengenannten Bedrohungen abschwächen bzw. die Ausnutzung von diesen schwierig machen.

Nr.	Beschreibung
R1	Der Key Generator muss die Keys so lange behalten, bis sie erfolgreich an den Key Server gesendet wurden.
R2	Redundanz schaffen, indem mehrere Anbieter für den gleichen Dienst verwendet werden.
R3	Mehrere Instanzen des Systems für Hochverfügbarkeit/Redundanz aufschalten.
R4	Einsatz von selbstheilenden Systemen mit Orchestrierung. (Benötigt ausserdem den Einsatz von Health Checks)
R5	Verwendung von Monitoring und Alarmierung.
R6	Einsatz von Metric collection und Autoscaling.
R7	Einsatz von redundanten Speichersystemen. (RAID, Ceph, Object-Storage, Datenbank-Cluster)
R8	Für Algorithmen und bekannte Implementierung wird auf bewährte Bibliotheken zurückgegriffen. (Keine Fehler durch eigenen Code)
R9	Durchführen von Code- und Konfig-Reviews. (automatisiert und manuell)
R10	Kontrolle des kompletten Kommunikationswegs, um Netzwerk-Spoofing zu verhindern.
R11	Limitierung der Anzahl Loginversuche pro Zeiteinheit.
R12	Einsatz einer Firewall (White- und Blacklisting, Portsperrung) Nur Ports und Verbindungen freigeben, welche auch verwendet werden.
R13	Einsatz von http Public Key Pinning, um Man-in-the-Middle Attacken zu verhindern.
R14	Kommunikation findet nur per TLS statt.
R15	Aktivierung verstärkter Authentifizierung (Benutzername/Passwort, Kennwortrichtlinien, Token, 2FA)
R16	Sensibilisierung der Administratoren auf Phishing und Social Hacking.
R17	Administratoren gezielt auswählen. Soviel wie nötig, so wenig wie möglich.
R18	Abgeschottetes Netzwerk ohne Zugriff von aussen (nur per Jump Host)
R19	Externer Zugriff von Administratoren nur mit 2-Faktor Authentifizierung zulassen.
R20	Chunk Dateien müssen verschlüsselt werden.
R21	Zugriff auf die Ressourcen und Server nur im Init7 Kundennetzwerk zulassen.
R22	Gespeicherte Keys werden nach Rotation gelöscht.
R23	Apps signieren.
R24	App auch auf Website anbieten.
R25	Apps nur aus offiziellem Store unterstützen.
R26	Apps regelmässig im Store überprüfen und Nachahmer den Storebetreibern melden.
R27	Root-Rechte/Jailbreak auf den Endgeräten erkennen und Betrieb der App verweigern.
R28	Datenübertragung muss authentifiziert sein.
R29	Schlüssel werden periodisch rotiert.
R30	Quelladresse im Request Header mit Inhalt des JWT Tokens abgleichen.
R31	Schlüsselaustausch findet nur verschlüsselt statt.
R32	Es werden pro TV-Sender verschiedene Keys verwendet.
R33	Key oder Token hat nur eine begrenzte Lebensdauer.

R34	Keys verschlüsselt in der Datenbank speichern.
R35	Für die Dienste und die Datenzugriffe werden nur Serviceuser mit begrenzten Zugriffsrechten verwendet.
R36	Verhindern von SQL-Injection durch die Verwendung von Prepared Statements.
R37	Einsatz von DRM-Lösungen, welche für die jeweiligen Endgeräte vorgesehen sind.
R38	Einschränken der Anzahl gleichzeitige Sessions/Logins.
R39	Es gibt keine geteilten Administratoren Logins, sondern nur personalisierte Logins.
R40	Zentralisierte (Applikations-)Log-Collection, kein bearbeiten der Admins.
R41	Datenbankadministratoren dürfen keinen Admin-Zugriff auf dem System haben.
R42	Arbeit mit Versionsverwaltung (Bsp. Git) und Continous Deployment.
R43	Ablauf der Administratoren Kennwörter nach einer gewissen Dauer.

7.3.2 Analyse Security Requirements

Tabelle 7.3: Analyse der Security Requirements.

Nr.	Beschreibung	Erfüllt	Weshalb
R1	Der Key Generator muss die Keys so lange behalten, bis sie erfolgreich an den Key Server gesendet wurden.	Ja	Der Key Generator schreibt die Keys auf das Filesystem, wenn die Übertragung zum Key Server nicht erfolgreich ist. Diese werden später nochmals übertragen.
R2	Redundanz schaffen, indem mehrere Anbieter für den gleichen Dienst verwendet werden.	Nein	Die Testinfrastruktur lies die Umsetzung dieser Anforderung nicht zu.
R3	Mehrere Instanzen des Systems für Hochverfügbarkeit/Redundanz aufschalten.	Teilweise	Im Code wurde darauf geachtet, dass die Systeme mit mehreren Instanzen redundant laufen können. Wurde aber aufgrund des Prototyps nicht umgesetzt.
R4	Einsatz von selbstheilenden Systemen mit Orchestrierung. (Benötigt ausserdem den Einsatz von Health Checks)	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R5	Verwendung von Monitoring und Alarmierung.	Nein	Die Testinfrastruktur lies die Umsetzung dieser Anforderung nicht zu.
R6	Einsatz von Metric collection und Autoscaling.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R7	Einsatz von redundanten Speichersystemen. (RAID, Ceph, Object-Storage, Datenbank-Cluster)	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R8	Für Algorithmen und bekannte Implementierung wird auf bewährte Bibliotheken zurück gegriffen. (Keine Fehler durch eigenen Code)	Ja	Code wurde bei der Implementierung von Algorithmen immer die Referenzimplementation verwendet.
R9	Durchführen von Code- und Konfig-Reviews. (automatisiert und manuell)	Teilweise	Es wurden manuelle Reviews durchgeführt. Auf automatisierte Reviews wurde verzichtet.
R10	Kontrolle des kompletten Kommunikationswegs, um Netzwerk-Spoofing zu verhindern.	Ja	Das System läuft in einem eigenem Netzwerk, welches unter eigener Kontrolle befindet.
R11	Limitierung der Anzahl Loginversuche pro Zeiteinheit.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R12	Einsatz einer Firewall (White- und Blacklisting, Portsperrung) Nur Ports und Verbindungen freigeben, welche auch verwendet werden.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.

Tabelle 7.3: Analyse der Security Requirements.

Nr.	Beschreibung	Erfüllt	Weshalb
R13	Einsatz von http Public Key Pinning, um Man-in-the-Middle Attacken zu verhindern.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R14	Kommunikation findet nur per TLS statt.	Ja	Kommunikation wurde mit https umgesetzt. Es werden gültige Zertifikate verwendet.
R15	Aktivierung verstärkter Authentifizierung (Benutzername/Passwort, Kennwortrichtlinien, Token, 2FA)	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R16	Sensibilisierung der Administratoren auf Phishing und Social Hacking.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R17	Administratoren gezielt auswählen. Soviel wie nötig, so wenig wie möglich.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R18	Abgeschottetes Netzwerk ohne Zugriff von aussen (nur per Jump Host)	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R19	Externer Zugriff von Administratoren nur mit 2-Faktor Authentifizierung zulassen.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R20	Chunk Dateien müssen verschlüsselt werden.	Ja	Dies war das Hauptziel der Arbeit.
R21	Zugriff auf die Ressourcen und Server nur im Init7 Kundennetzwerk zulassen.	Ja	Der Zugriff auf die Ressourcen ist über IP Filterung geregelt.
R22	Gespeicherte Keys werden nach Rotation gelöscht.	Ja	Das Verfallsdatum kann vom Systembetreiber gewählt werden. Die Keys werden gelöscht, sobald sie nicht mehr gebraucht werden.
R23	Apps signieren.	Ja	Apps sind durch Apple signiert.
R24	App auch auf Website anbieten.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R25	Apps nur aus offiziellem Store unterstützen.	Ja	Nur offizielle Apps mit dem richtigen Zertifikat können auf die Streams zugreifen.
R26	Apps regelmässig im Store überprüfen und Nachahmer den Storebetreibern melden.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt. Dies liegt am Deployment, kann individuel umgesetzt werden.
R27	Root-Rechte/Jailbreak auf den Endgeräten erkennen und Betrieb der App verweigern.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.

Tabelle 7.3: Analyse der Security Requirements.

Nr.	Beschreibung	Erfüllt	Weshalb
R28	Datenübertragung muss authentifiziert sein.	Ja	Für die Kommunikation zwischen den Microservices werden Tokens verwendet.
R29	Schlüssel werden periodisch rotiert.	Ja	Key Generator erstellt periodisch neue Schlüssel. Der Ablauf wird durch den Key Server geregelt.
R30	Quelladresse im Request Header mit Inhalt des JWT Tokens abgleichen.	Nein	Wurde aus Zeitgründen nicht mehr umgesetzt.
R31	Schlüsselaustausch findet nur verschlüsselt statt.	Ja	Der Schlüsselaustausch findet über CEK Nachrichten statt. Diese sind durch FairPlay Streaming verschlüsselt.
R32	Es werden pro TV-Sender verschiedene Keys verwendet.	Ja	Pro Sender wird ein seperater Key Generator eingesetzt.
R33	Key oder Token hat nur eine begrenzte Lebensdauer.	Teilweise	Token Lebensdauer wurde noch nicht festgelegt
R34	Keys verschlüsselt in der Datenbank speichern.	Nein	Wurde aus Zeitgründen nicht mehr umgesetzt.
R35	Für die Dienste und die Datenzugriffe werden nur Serviceuser mit begrenzten Zugriffsrechten verwendet.	Ja	Pro Dienst gibt es einen Benutzer mit eingeschränkten Rechten. Dieser kann mit einem Token authentifiziert werden.
R36	Verhindern von SQL-Injection durch die Verwendung von Prepared Statements.	Ja	Für die Implementation der Datenbank wurde das Django Rest Framework verwendet.
R37	Einsatz von DRM-Lösungen, welche für die jeweiligen Endgeräte vorgesehen sind.	Ja	Es wird FairPlay Streaming für Apple Geräte eingesetzt.
R38	Einschränken der Anzahl gleichzeitige Sessions/Logins.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R39	Es gibt keine geteilten Administratoren Logins, sondern nur personalisierte Logins.	Nein	Es wurden perönliche Konten eingerichtet. Jedoch ist der sudo Befehl nicht gesperrt. Dies wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R40	Zentralisierte (Applikations-)Log-Collection, kein bearbeiten der Admins.	Nein	Momentan gibt es pro Microservice ein Log. Diese werden nicht mit einem Log Collector gesammelt.
R41	Datenbankadministratoren dürfen keinen Admin-Zugriff auf dem System haben.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.
R42	Arbeit mit Versionsverwaltung (Bsp. Git) und Continous Deployment.	Teilweise	Das System wurde mit Github entwickelt. Jedoch ist das Deployment auf die Testinfrastruktur manuell.

Tabelle 7.3: Analyse der Security Requirements.

Nr.	Beschreibung	Erfüllt	Weshalb
R43	Ablauf der Administratoren Kennwörter nach einer gewissen Dauer.	Nein	Wurde nicht umgesetzt, da es sich um einen Prototypen handelt.

7.3.3 Risiko Analyse

Der Prototyp und die Architektur wurden in der Evaluierung mit einer Risiko Analyse überprüft. Es wurde für jede Schwachstelle die Wahrscheinlichkeit einer Ausnutzung und der Einfluss festgelegt. Das genaue Vorgehen wird im Abschnitt 3.3 erklärt. Eine Auswertung der Analyse ist im Unterabschnitt 4.6.3. Auf der nächsten Seite sind die Schwachstellen und ihre jeweiligen Analysen aufgelistet.

Risikoanalyse TV-Architektur

Nr.	Element	Kat.	Realistisch	Mitigiert	Wahrsch.	Einfluss	Risko
T1	TV Konsument	S	Ja	Nein	Medium	Medium	Medium
T2		R	Nein	Nein	Medium	Medium	Medium
T3	DCM	S	Nein	Ja	Low	High	Low
T4		R	Nein	Nein	Low	High	Low
T5	EPG Provider	S	Nein	Nein	Low	Low	Low
T6		R	Nein	Nein	Low	Low	Low
T7	Get Auth Token	T	Nein	Ja	Low	Low	Low
T8		I	Ja	Ja	Medium	High	Medium
T9		D	Nein	Nein	Low	Medium	Low
T10	Get HLS Playlist	T	Nein	Ja	Low	Medium	Low
T11		I	Ja	Ja	Low	Low	Low
T12		D	Nein	Nein	Low	Low	Low
T13	Get Chunk	T	Nein	Ja	Low	Medium	Low
T14		I	Ja	Ja	Low	Low	Low
T15		D	Nein	Nein	Low	Low	Low
T16	Get Decryption Context	T	Nein	Ja	Low	Low	Low
T17		I	Ja	Ja	Medium	Low	Low
T18		D	Nein	Nein	Low	Low	Low
T19	Store/read chunks	T	Nein	Ja	Low	Medium	Low
T21		D	Nein	Nein	Low	Medium	Low
T22	Fetch chunks	T	Nein	Ja	Low	Medium	Low
T23		I	Nein	Ja	Low	Low	Low
T24		D	Nein	Ja	Low	Medium	Low
T25	Inform for new Chunks	T	Nein	Ja	Low	Medium	Low
T26		I	Nein	Ja	Low	Low	Low
T27		D	Nein	Nein	Low	Medium	Low
T28	Get Information from EPG Proxy	T	Nein	Ja	Low	Low	Low
T29		I	Nein	Ja	Medium	Low	Low
T30		D	Nein	Nein	Low	Low	Low
T31	Send key and reference	T	Nein	Ja	Low	Medium	Low
T32		I	Ja	Ja	Low	Medium	Low
T33		D	Nein	Ja	Low	Medium	Low
T34	Store/read Key	T	Nein	Ja	Low	High	Low
T36		D	Nein	Nein	Low	High	Low
T37	Get key for KeyID	T	Nein	Ja	Low	Medium	Low
T38		I	Ja	Ja	Low	High	Low
T39		D	Nein	Nein	Low	High	Low
T40	Fetch Multicast Stream	T	Nein	Ja	Low	Medium	Low
T41		I	Nein	Ja	Low	High	Low
T42		D	Ja	Nein	Medium	Medium	Medium
T43	Get program information	T	Nein	Ja	Low	Low	Low
T44		I	Nein	Ja	Low	Low	Low
T45		D	Ja	Nein	Low	Low	Low

Nr.	Element	Kat.	Realistisch	Mitigiert	Wahrsch.	Einfluss	Risko
T46	Chunk Files	T	Nein	Ja	Low	Medium	Low
T47		R	Nein	Nein	Low	Medium	Low
T48		I	Nein	Ja	Low	Low	Low
T49		D	Nein	Nein	Medium	Medium	Medium
T50	Key Database	T	Nein	Ja	Low	High	Low
T51		R	Nein	Nein	Low	High	Low
T52		I	Ja	Ja	Low	High	Low
T53		D	Nein	Ja	Low	High	Low
T54	Auth Server	S	Nein	Ja	Low	High	Low
T55		T	Nein	Ja	Low	High	Low
T56		R	Nein	Ja	Low	High	Low
T57		I	Ja	Ja	Low	High	Low
T58		D	Ja	Nein	Low	Medium	Low
T59		E	Nein	Ja	Low	High	Low
T60	TV API	S	Nein	Ja	Low	High	Low
T61		T	Nein	Ja	Low	High	Low
T62		R	Nein	Nein	Medium	High	Medium
T63		I		Ja	Low	Low	Low
T64		D	Ja	Nein	Medium	High	Medium
T65		E	Nein	Ja	Low	Medium	Low
T66	TV Cache	S	Nein	Nein	Low	Medium	Low
T67		T	Nein	Ja	Low	Medium	Low
T68		R	Nein	Nein	Low	Medium	Low
T69		I	Ja	Ja	Low	Low	Low
T70		D	Ja	Nein	Medium	Low	Low
T71		E	Nein	Ja	Low	Medium	Low
T72	TV Catch	S	Nein	Ja	Low	Medium	Low
T73		T	Nein	Ja	Low	Medium	Low
T74		R	Nein	Nein	Low	Medium	Low
T75		I	Nein	Ja	Low	Medium	Low
T76		D	Nein	Nein	Low	Medium	Low
T77		E	Nein	Ja	Low	Medium	Low
T78	EPG Proxy	S	Nein	Nein	Low	Low	Low
T79		T	Nein	Ja	Low	Low	Low
T80		R	Nein	Nein	Low	Low	Low
T81		I	Nein	Ja	Low	Low	Low
T82		D	Nein	Nein	Low	Low	Low
T83		E	Ja	Ja	Low	Medium	Low
T85	Key Management System	T	Nein	Ja	Low	High	Low
T86		R	Nein	Nein	Medium	High	Medium
T87		I	Ja	Ja	Low	High	Low
T88		D	Nein	Nein	Low	High	Low
T89		E	Nein	Ja	Low	High	Low
T90	FairPlay	S	Ja	Ja	Medium	High	Medium
T91		T	Nein	Ja	Low	High	Low
T92		R	Nein	Nein	Low	High	Low
T93		I	Ja	Ja	Low	High	Low

Nr.	Element	Kat.	Realistisch	Mitigiert	Wahrsch.	Einfluss	Risko
T94		D	Ja	Nein	Medium	High	Medium
T95		E	Nein	Ja	Medium	High	Medium
T96	Administratoren	S	Ja	Ja	Medium	High	Medium
T97		R	Nein	Nein	Medium	High	Medium
T98	App	S	Ja	Ja	Medium	Medium	Medium
T99		T	Nein	Ja	Low	High	Low
T100		R	Nein	Nein	Low	High	Low
T101		I	Ja	Ja	Low	High	Low
T102		D	Nein	Nein	Low	High	Low
T103		E	Ja	Nein	Low	High	Low
T104	Show Stream	T	Nein	Ja	Low	Low	Low
T105		I	Ja	Nein	Medium	Low	Low
T106	Send Auth Token	T	Ja	Ja	Medium	High	Medium
T107		I	Ja	Ja	Medium	High	Medium
T108		D	Nein	Nein	Low	Low	Low
T109	Chunk Info Database	T	Nein	Ja	Low	Medium	Low
T110		R	Nein	Nein	Low	Medium	Low
T111		I	Nein	Ja	Low	Low	Low
T112		D	Ja	Nein	Medium	High	Medium

7.3.4 Code für Simulation

Für die Simulation der Schlüsselanfragen wurde ein Programm in Matlab geschrieben. Damit konnte überprüft werden, welche Intervalle sich für die Key-Rotation eignen. Die Methodik für die Simulation wird im Unterabschnitt 3.3.1 beschrieben. Im Unterabschnitt 4.6.1 werden die Ergebnisse ausgewertet.

```
% variables
testTime=1440;
x = 1:180;

% Generate addedConsumers - They switch on the tv and never turn it off
maxCostumersFluctuation = 3;
addedConsumers = zeros(1,testTime);
for a = 1:testTime

        change = round(maxCostumersFluctuation * rand(1));

        if(change>= 0)
            addedConsumers(a)=change;
        else
            addedConsumers(a)= 0;
        end

end

% Simulate the requests
for c = 1:180
    w = zeros(1,testTime);
    for i = 1:testTime
        w(i)=addedConsumers(i);
        for t = 1:i
            if(mod(t,c)==0 && t~=i)
                w(t)=w(t)+addedConsumers(t);
            end
        end
    end
    y(c) = sum(w) ./testTime .*60
end

% Plotting
plot(x,y,'Color','r','LineWidth',2)
title("Key-Anfragen pro Stunde");
subtitle("Simulation über 24 Stunden");
xlabel("Key-Rotationen pro x min")
ylabel("Anfragen pro Stunde")
```