HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Frau
Kristina Tanja Koska

Web Application Angriffe – Zustandsbasierte Angriffe im Kontext von HTTPS-Verbindungen

Mittweida, 2022

Fakultät
Angewandte Computer- und Biowissenschaften

BACHELORARBEIT

Web Application Angriffe – Zustandsbasierte Angriffe im Kontext von HTTPS-Verbindungen

Autor:
Frau Kristina Tanja Koska

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO17w4-B

Erstprüfer:
Prof. Dipl.-Ing. (BA) Ronny Bodach

Zweitprüfer:
M. Sc. Stefan Schildbach

Einreichung:
Mittweida, 24.07.2022

Verteidigung/Bewertung:
Mittweida, 2022

Faculty
Applied Computer Sciences and Biosciences

BACHELOR THESIS

Web Application Attacks – Session Management Attacks in Context of HTTPS Connections

author:
Ms. Kristina Tanja Koska

course of studies:
General and Digital Forensic Science

seminar group:
FO17w4-B

first examiner:
Prof. Dipl.-Ing. (BA) Ronny Bodach

second examiner:
M. Sc. Stefan Schildbach

submission:
Mittweida, 24.07.2022

defence/ evaluation:
Mittweida, 2022

Bibliografische Beschreibung:

Kristina Tanja, Koska:
Web Application Angriffe – Zustandsbasierte Angriffe im Kontext von HTTPS-Verbindungen. - 2022. – 5, 41 S.
Mittweida, Hochschule Mittweida, Fakultät Angewandte Computer- und Biowissenschaften, Bachelorarbeit, 2022

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
TLS	Transport Layer Security
SSL	Secure Sockets Layer
HTML	Hypertext Markup Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
CSS	Cascading Style Sheets
POODLE	Padding Oracle On Downgraded Legacy Encryption
CBC	Cipher Block Chaining
CRIME	Compression Ratio Info-leak Made Easy
BREACH	Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext
ROBOT	Return of Bleichenbacher’s Oracle Threat
HEIST	HTTP Encrypted Information can be Stolen through TCP-windows
API	Application Programming Interface
MAC	Media Access Control
HSTS	HTTP Strict Transport Security
SMACK	State Machine Attack
CSRF	Cross-Site Request Forgery
CVSS	Common Vulnerability Scoring System
BEAST	Browser Exploit Against SSL/TLS
DROWN	Decrypting RSA using Obsolete and Weakened Encryption
RFC	Request For Comments

1 Einleitung

Sobald eine Webanwendung Sitzungsinformationen austauscht, ist Vorsicht geboten. Unzureichender Schutz bedeutet ein hohes Risiko für Anmeldeinformationen wie Passwörter oder eben auch Session-Tokens. Potentielle Angreifer können sich so Zugang zu einem Nutzeraccount verschaffen und dessen Identität annehmen. Infolgedessen bedarf es zahlreicher Sicherheitsmaßnahmen, um solche Informationen bestmöglich schützen zu können.

1.1 Motivation

Das Open Web Application Security Project – kurz OWASP – eine Organisation, welche sich grundsätzlich für eine Verbesserung der Software-Sicherheit einsetzt, erstellt auch frei verfügbare Materialien zur Sicherheit von Webanwendungen und veröffentlicht unter den OWASP Top 10 alle paar Jahre deren zehn kritischste Schwachstellen. Das eigentliche Ziel ist aber darüber aufzuklären, wie die Risiken minimiert werden können. [01]

Access Control beschreibt die Durchsetzung der Beschränkungen für (nicht) authentifizierte Nutzer – um zu verhindern, dass Aktionen außerhalb einer jeweiligen Berechtigungsstufe durchgeführt werden. Broken Access Control liegt also dann vor, wenn solche Beschränkungen nicht oder nur teilweise durchgesetzt werden. Unbefugter Zugriff auf sensible Informationen kann im schlimmsten Fall zu deren Änderung oder Löschung führen. [01] Da das Ausmaß nach dem Ausnutzen einer solchen Schwachstelle enorm sein kann, wurde diese Sicherheitslücke im Vergleich von 2017 zu 2021 vom fünften auf den ersten Rang hochgestuft und führt damit die Liste der kritischsten sogar an (Abbildung 1) – immerhin 94% aller getesteten Webanwendungen sind anfällig für eine Form von Broken Access Control. [02]

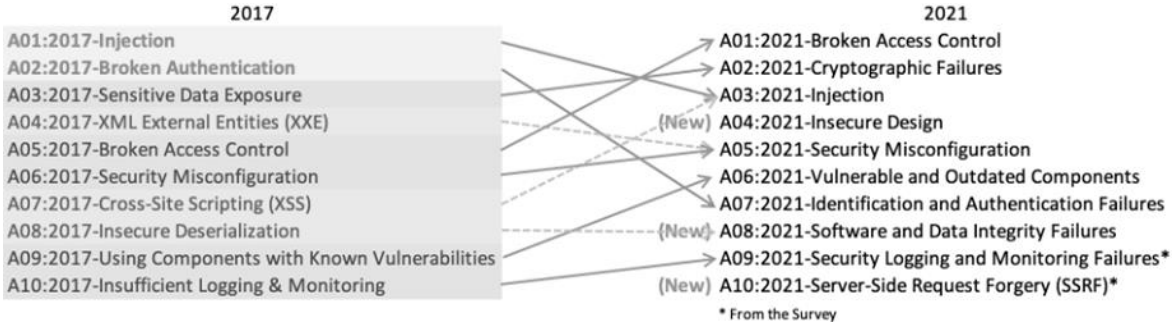

Abbildung 1: OWASP Top 10 (2017 versus 2021)

Quelle: [02]

Zu den möglichen Formen von Broken Access Control im Zusammenhang mit dem Session Management zählen unter anderem die Manipulation von Parametern beziehungsweise

das erzwungene Browsen durch eine Änderung der URL, die Manipulation von Metadaten – wie Cookies oder versteckter Felder – oder das forcierte Browsen, um auf eigentlich authentisierte Bereiche zuzugreifen. [01]

Gemäß dem entsprechenden OWASP-Risk-Rating-Modell lässt sich ein Fehler im Session Management also folgendermaßen klassifizieren (Tabelle 1):

Tabelle 1: OWASP Risk Rating Modell

Quelle: [03]

Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Deductibility	Technical Impact	Business Impact
APP Specific	Easy	Widespread	Easy	Severe	/App Business Specific
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

Die Bedrohungsquelle stellen nicht authentifizierte Angreifer sowie bereits authentifizierte Nutzer – also Innentäter – dar, welche versuchen könnten die Zugangsdaten anderer zu stehlen respektive deren Handlungen zu verschleiern. [03]

Beim Angriffsvektor handelt es sich also um Lücken im Session Management – vor allem entsprechende IDs – mit dem Ziel eine fremde Identität zu übernehmen. Die Ausnutzbarkeit wird als durchschnittlich beschrieben. [03]

Trotz dessen, dass es sehr schwer ist, ein sicheres Session Management zu implementieren, setzen viele Entwickler auf eigene Lösungen. Dementsprechend schleichen sich auch zahlreiche Fehler im Zusammenhang mit der Abmeldung, der Wiedererkennung des Nutzers oder Timeouts, ein – diese sind also recht verbreitet. Solche Schwachstellen zu finden kann demnach Probleme bereiten. [03]

Die technischen Auswirkungen sind insofern schwerwiegend, dass solche Fehler häufig zur Kompromittierung von Nutzerkonten führen. Ist eine Attacke erfolgreich, besitzt der Angreifer alle Rechte des Opfers – privilegierte Zugänge sind also oft das Ziel. [03]

Um die Auswirkungen auf das Unternehmen zu ermitteln, muss der Geschäftswert der betroffenen Daten und Funktionen beziehungsweise das Ausmaß bei Bekanntwerden der Schwachstelle ermittelt werden. [03]

1.2 Zielsetzung

Wirksame Sicherheitsmaßnahmen zu entwickeln und auch zu implementieren, bedeutet im Umkehrschluss, dass zuerst einmal die zugrundeliegenden Gefahren, Schwachstellen und Sicherheitslücken bekannt sein und verstanden werden müssen. Das beste System hat wenig Nutzen, wenn der Mensch, der es bedient, es nicht zu nutzen weiß.

Aus diesem Grund soll mit der vorliegenden Arbeit ein grundlegendes Verständnis zu möglichen Angriffen auf das Session Management beziehungsweise in diesem Fall auch die Verschlüsselung gelegt werden. Im Anschluss daran werden mögliche Schutzmaßnahmen dargelegt, welche als eine Art Leitfaden dienen sollen.

2 Grundlagen und Stand der Technik

Im Jahr 1969 gelang es Ingenieuren, zwei Computer miteinander zu vernetzen – mit dem Ziel Informationen auszutauschen. Aus der Verbindung von immer mehr Computern wuchsen ganze Netzwerke und das heutige Internet entstand. [04]

Das World Wide Web ist dabei ein angebotener Dienst im Internet – entwickelt vom Physiker Tim Berners-Lee und für die Öffentlichkeit zugänglich seit dem Jahr 1993. Es ermöglicht, multimediale Inhalte bereitzustellen und natürlich auch auszutauschen. Das Netzwerk beruht dabei auf Seiten, welche auf zahlreichen Servern weltweit vorgehalten werden. Ein Hyperlink – das Grundprinzip des World Wide Web – verweist dann auf den jeweiligen Ort, an dem die Seite aufbewahrt wird. [04]

Die tatsächliche Adresse eines Hyperlinks bezeichnet man als Uniform Resource Locator, kurz URL, oder Uniform Resource Identifier, kurz URI. Sie bestehen aus einem Schema und einem Schema-spezifischen Teil – durch einen Doppelpunkt voneinander getrennt. Der Schema-spezifische Teil besteht aus der Domain des Servers und einer genauen Pfadangabe. [04]

2.1 Internetprotokollfamilie

Den Informationsaustausch selbst übernehmen Netzwerkprotokolle – grundlegend handelt es sich hierbei um ein Regelwerk, welches die Übertragung der Daten zwischen verschiedenen Systemen definiert. [05]

Die Internetprotokollfamilie ist dabei eine Sammlung solcher Netzwerkprotokolle und realisiert, wie der Name schon sagt, die Kommunikation im Internet. Jedes dieser Protokolle übernimmt unterschiedliche Aufgaben. Um diese gruppieren zu können, werden sie unterschiedlichen Schichten zugeordnet. Das TCP/IP-Referenzmodell, welches in diesem Fall genutzt wird, besitzt vier solcher aufeinander aufbauenden Schichten (Tabelle 2). [05]

Tabelle 2: Relevante Protokolle im TCP/IP-Referenzmodell

Quelle: eigene Darstellung

<table>
<tr><td rowspan="2">Anwendung</td><td>HTTP</td><td colspan="2">HTTPS</td></tr>
<tr><td></td><td>TLS</td><td rowspan="2">QUIC</td></tr>
<tr><td rowspan="2">Transport</td><td></td><td></td></tr>
<tr><td colspan="2">TCP</td><td>UDP</td></tr>
<tr><td>Internet</td><td colspan="3">IP</td></tr>
<tr><td>Netzzugang</td><td colspan="3"></td></tr>
</table>

Die Protokolle werden in sogenannten Request For Comments, kurz RFCs, beschrieben und erst von der Internet Engineering Task Force zum Internetstandard erhoben. Diesen braucht es, da die zahlreichen dezentralen Netzwerke ansonsten nicht miteinander kommunizieren könnten – darüber wird also eine einheitliche Sprache definiert. [06]

2.1.1 Transmission Control Protocol

Das Transmission Control Protocol, kurz TCP, ist eine Implementierung der Transportschicht. Die standardisierte Vereinbarung, erstmals festgehalten in RFC 793, sieht also eine Informationsübertragung zwischen verschiedenen Teilnehmern eines Netzwerks vor. [07] Vereinfacht kann diese Kommunikation auch als Client-Server-Interaktion beschrieben werden. Der Client sendet eine Anfrage an den Server. Letzterer prüft dann wiederum, ob ersterer dazu berechtigt ist auf die angefragte Ressource zuzugreifen – falls ja, kann die Übertragung gestartet werden. [08]

Da TCP bidirektionale Kommunikation zulässt – zu gleicher Zeit also Daten gesendet und empfangen werden können – ist es nicht von Bedeutung, welcher Teilnehmer die Client- und welcher die Serverrolle einnimmt. Die jeweiligen Endpunkte (Sockets) müssen lediglich eindeutig identifizierbar sein – ermöglicht wird dies durch eine Kombination aus IP-Adresse und Portnummer, womit dann ein bestimmter Rechner respektive Dienst angesprochen werden kann. [07]

In diesem Fall handelt es sich um ein verbindungsorientiertes Protokoll – bevor also überhaupt eine Datenübertragung stattfinden kann, muss eine Verbindung aufgebaut und so lange aufrechterhalten werden, bis diese von beiden Kommunikationsteilnehmern auch wieder beendet wird. [09] Die entscheidenden Daten befinden sich hierbei im Header eines Pakets (Tabelle 3). Mit einer standardmäßigen Größe von 20 Byte – erweiterbar um 40 Byte – enthält dieser Steuerinformationen wie die Quell- und Zielportnummern, gesetzte Flags – diese aktivieren dann verschiedene TCP-Aktionen – oder die maximale Größe der nachstehenden inhaltlich relevanten Nutzdaten (Payload). [07]

Tabelle 3: TCP-Header

Quelle: [07]

Bits	0-15			16-31
0	Quell-Port			Ziel-Port
32	Sequenznummer			
64	Bestätigungsnummer			
96	Offset	Reserviert	Flags	Fenster-/Empfangsgröße
128	Prüfsumme			Urgent-Pointer
160	Optionen			

Der TCP-Verbindungsaufbau umfasst insgesamt drei Schritte – er wird deshalb auch als Drei-Wege-Handshake bezeichnet (Abbildung 2). Die initiale Anfrage des Clients enthält unter anderem ein gesetztes SYN-Flag (von englisch synchronize) und eine individuelle,

zufällig gewählte Sequenznummer – eine dahingehende Synchronisation mit dem Server stellt die vollständige Übertragung aller Datenpakete sowohl in der korrekten Reihenfolge als auch ohne Duplikate sicher. Im Falle, dass dieser dem Verbindungsaufbau zustimmt, enthält das zu sendende Segment gesetzte SYN- und ACK-Flags (von englisch acknowledge) – der Server bestätigt also die Sequenznummer des Clients, indem er diese um 1 erhöht, und übermittelt zusätzlich seine eigene. Ist dessen Port allerdings geschlossen oder der Zugriff anderweitig blockiert, enthält das Segment stattdessen ein RST-Flag (von englisch reset) – die Anfrage wird also zurückgesetzt. Den Erhalt des Bestätigungspakets quittiert der Client nun mit einem Segment, welches ebenfalls das gesetzte ACK-Flag und zusätzlich die um 1 erhöhte Sequenznummer des Servers enthält – gleichzeitig können bereits erste Daten übertragen werden. [07]

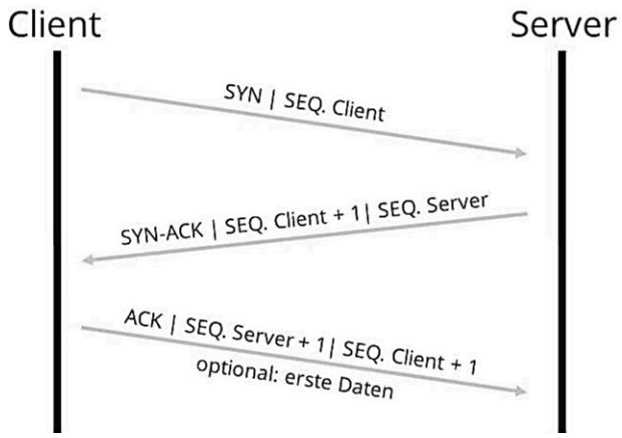

Abbildung 2: TCP-Verbindungsaufbau

Quelle: in Anlehnung an [07]

Standardmäßig beträgt die maximale Größe der zu sendenden Pakete 1500 Byte. Bedingt durch voranstehende Header bleiben 1460 Byte für reine Nutzdaten übrig. Damit nun auch ganze Webinhalte übertragen werden können, kommt es vor dem Transport zu einer Segmentierung. Die Daten werden also in mehrere Blöcke aufgeteilt und dann in zufälliger Abfolge versendet. Anhand der Sequenznummern kann die tatsächliche Reihenfolge allerdings rekonstruiert werden, sodass eine erneute Zusammensetzung im Anschluss der Übertragung kein Problem darstellt. Auch im Falle eines Übermittlungsfehlers oder einer fehlenden Rückmeldung seitens des Empfängers nach Ablauf eines Timers (Retransmission Timeout) wird automatisch der erneute Versand des betroffenen Pakets initiiert. Da eine vollständige Übertragung somit jederzeit gewährleistet werden kann, handelt es sich außerdem um ein zuverlässiges Protokoll. [07]

Der TCP-Verbindungsabbau, auch als TCP-Teardown bezeichnet, umfasst insgesamt vier Schritte (Abbildung 3). Sobald der Client keine Daten mehr übermitteln möchte, sendet er ein Segment mit gesetztem FIN-Flag (von englisch finish) und seiner Sequenznummer – damit wird die Verbindung von seiner Seite aus beendet. Der Server wiederum quittiert dessen Erhalt mit einem Paket, welches ein gesetztes ACK-Flag und die um 1 erhöhte Sequenznummer enthält. Letzterem ist es aber nach wie vor erlaubt Daten an die Gegenseite

zu übertragen – man spricht auch von einer halb geschlossenen Verbindung. Erst mit dem Versand eines Segments, welches ebenfalls ein FIN-Flag und dessen Sequenznummer enthält, teilt er dem Client mit, dass auch er keine weiteren Daten mehr übermitteln möchte. Dieser bestätigt dessen Erhalt noch ein letztes Mal mit einem Paket, welches ein gesetztes ACK-Flag und wiederum die um 1 erhöhte Sequenznummer des Servers enthält. Die Verbindung ist erst dann vollständig getrennt, wenn die maximale Laufzeit dieses Segments verstrichen ist – solange verweilt der Client in einem Wartemodus (Time-Wait-Zustand). [07]

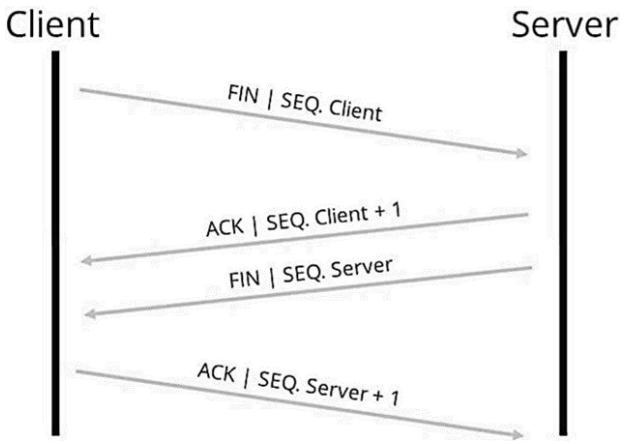

Abbildung 3: TCP-Verbindungsabbau

Quelle: in Anlehnung an [07]

2.1.2 Hypertext Transfer Protocol

Das Hypertext Transfer Protocol, kurz HTTP, ist eine Implementierung der Anwendungsschicht. Die standardisierte Vereinbarung, deren aktuell noch meistverbreitete Version in RFC 7540 festgehalten ist, sieht primär die Übertragung von Webseiten zwischen verschiedenen Teilnehmern des Internets vor. [10] Ursprünglich konnten lediglich mit Hypertext Markup Language, kurz HTML, verfasste Dokumente übermittelt werden. Heute können Browser alle Arten von Medien anfordern und sogar auf Datenbanken zugreifen, um Änderungen daran vorzunehmen. [11]

Um all das leisten und für eine vollständige Übertragung dieser Daten sorgen zu können, verwendet HTTP das verbindungsorientierte und zuverlässige TCP – letzteres nutzt für eine Adressierung dann die Portnummer 80. [10] Entsprechend der Client-Server-Architektur dieses Protokolls kommt hier das sogenannte Request-Response-Verfahren zum Einsatz (Abbildung 4). Ruft der Nutzer eine bestimmte Webseite auf, sendet der Client – in diesem Fall also der Browser – einen entsprechenden HTTP-Request an den Server. Dessen HTTP-Response wiederum setzt sich – je nachdem, ob die angeforderte Ressource gefunden werden kann – aus dem Resultat der Suche in Form eines Statuscodes und dem eigentlichen Inhalt zusammen. Für das Rendering, das heißt die endgültige Darstellung der Webseite, ist der Browser dann selbst verantwortlich. [11]

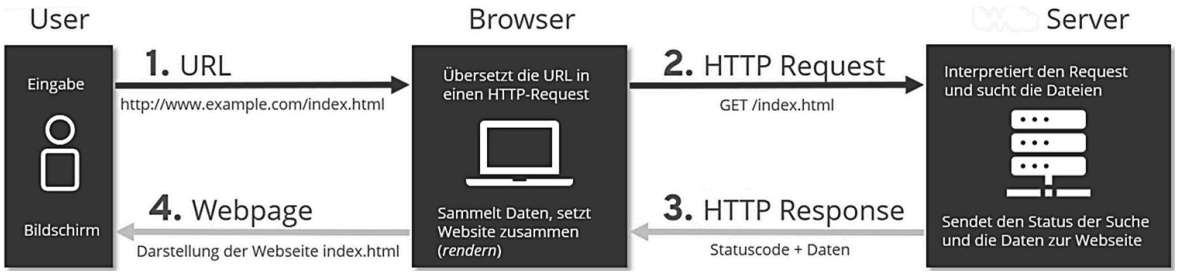

Abbildung 4: Kommunikationsprozess gemäß HTTP

Quelle: in Anlehnung an [12]

Ein HTTP-Request besteht grundsätzlich aus drei Komponenten: der Anfragezeile, den Kopfdaten und dem optionalen Nachrichtenrumpf. Erstere enthält eine Methode (Tabelle 4), eine URI beziehungsweise URL und die verwendete Protokollversion. [12]

Tabelle 4: HTTP-Anfragemethoden

Quelle: in Anlehnung an [12]

GET	ruft den angeforderten Inhalt ab
HEAD	weist den Server an, lediglich die Kopfdaten zu senden
POST	übermittelt Daten an den Server
PUT	führt Änderungen an einer Ressource durch
DELETE	löscht eine vorhandene Ressource
TRACE	liefert den Request so zurück, wie der Server ihn empfangen hat

Aus Sicherheitsgründen werden serverseitig allerdings meist nur die Methoden 1 bis 3 implementiert [13]. GET und POST ermöglichen dann sowohl den Abruf einer Webseite als auch die Übermittlung zusätzlicher Informationen – diese werden an entsprechende URI- beziehungsweise URL-Parameter oder Header respektive den Body übergeben [14]. POST wird also immer dann bevorzugt, wenn eine große Menge sensibler Daten unterschiedlichen Typs übertragen oder eine Änderung an der angeforderten Ressource durchgeführt werden soll. [12]

Eine HTTP-Response dagegen besteht grundsätzlich ebenfalls aus diesen drei Komponenten – analog handelt es sich hierbei um die Antwortzeile. Diese enthält die Protokollversion, das Resultat der Anfrage in Form eines Statuscodes (Tabelle 5) und dessen Beschreibung in Klartext [15]. Abhängig vom Erfolg des Requests beinhaltet der Header dann Informationen zur entsprechenden Ressource, die dem Client beispielsweise das Rendering oder Caching beziehungsweise im Falle eines Fehlers dessen Behebung erleichtern sollen. Nur im Falle, dass es sich um einen 2xx-Statuscode handelt, wird auch der tatsächliche Inhalt der Ressource übertragen. Ansonsten kann auch der Nachrichtenrumpf weitere Informationen liefern – entweder zu dem Fehler selbst oder dazu, welche Schritte seitens des Clients für eine erfolgreiche Bearbeitung der Anfrage erforderlich sind. [16]

Tabelle 5: HTTP-Statuscodes

Quelle: in Anlehnung an [15]

1xx	Information	die Anfrage dauert an – währenddessen werden allerdings Informationen zur Bearbeitung geliefert
2xx	Erfolg	die Anfrage wurde empfangen, verstanden und akzeptiert
3xx	Umleitung	die Anfrage wurde empfangen – zur erfolgreichen Bearbeitung sind weitere Schritte seitens des Clients erforderlich
4xx	Client-Fehler	die Anfrage wurde empfangen, kann aber aufgrund eines Fehlers seitens des Clients nicht ausgeführt werden
5xx	Server-Fehler	die Anfrage kann vorübergehend nicht ausgeführt werden oder ist seitens des Servers gar unmöglich

Da jeder Request-Response-Zyklus in sich abgeschlossen und vollkommen unabhängig von bisherigen Vorgängen stattfindet, müssen Informationen nicht notwendigerweise zwischengespeichert werden – jede Anfrage enthält also alle erforderlichen Informationen für deren Bearbeitung. Demnach handelt es sich hierbei um ein zustandsloses Protokoll. Trotz dessen bietet HTTP die Option einer Basisauthentifizierung – der Zugriff zu einer Datei wird nur gewährt, wenn Benutzername und Passwort korrekt sind [10]. Solche zustandsbehafteten Abläufe lassen sich jedoch auch auf einer höheren Kommunikationsebene implementieren. In diesem Fall antwortet der Server mit dem Statuscode 401 und dem Header WWW-Authenticate [10]. Im nächsten Request kann der Client dann die nötigen Daten unter Verwendung von POST übermitteln – HTTP überträgt diese Informationen allerdings unverschlüsselt im Klartext. [17]

2.1.3 Hypertext Transfer Protocol Secure

Die Problematik der unsicheren Datenübertragung im Internet wird mit der Einführung eines Verschlüsselungsprotokolls adressiert. Das Hypertext Transfer Protocol Secure, kurz HTTPS, ist dabei lediglich eine Erweiterung des ursprünglichen HTTP – ergänzt um den Secure Sockets Layer, kurz SSL, beziehungsweise dessen Nachfolger Transport Layer Security, kurz TLS, und damit erstmals festgehalten in RFC 2818. [18]

In diesem Fall nutzt TCP für eine Adressierung dann einfach die Portnummer 443 [18]. War der Verbindungsaufbau erfolgreich, müssen sich Client und Server nun auf einen einzigartigen Sitzungsschlüssel einigen, der anschließend für die Verschlüsselung der Daten verwendet werden kann – dieser Prozess wird als TLS-Handshake bezeichnet (Abbildung 5). Im ersten Schritt wird sowohl die aktuellste auf beiden Seiten verfügbare Protokollversion als auch die zu verwendende Cipher Suite – also eine Kombination verschiedener kryptografischer Verfahren – ausgehandelt. [19] Daraufhin stellt der Server sein Sicherheitszertifikat zur Verfügung. Nach dessen erfolgreicher Prüfung auf Gültigkeit, kann der Client dann eine zufällig generierte Zahl mithilfe des zusätzlich übermittelten Public Key verschlüsseln. Zur Dekodierung nutzt der Server wiederum seinen Private Key. Nun können beide

Kommunikationsteilnehmer die zuvor ausgehandelte Cipher Suite zur Erzeugung eines gemeinsamen Session Key verwenden – dieser allein dient dann der Verschlüsselung aller folgenden Nachrichten. Aus Sicherheits- und Ressourcengründen wird auf eine hybride Vorgehensweise gesetzt – die weniger anfällige, aber mathematisch komplexe asymmetrische Kryptografie findet also lediglich bei der Übertragung des Sitzungsschlüssels Anwendung. [20]

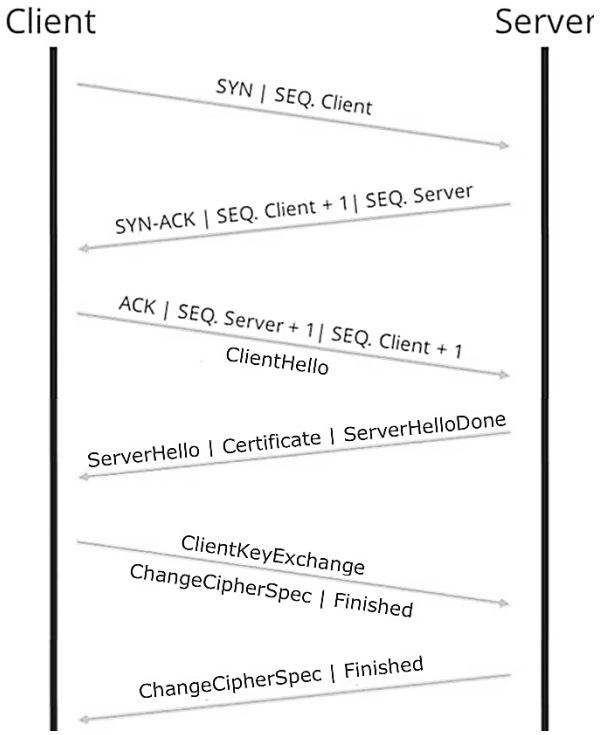

Abbildung 5: TLS-Handshake

Quelle: eigene Darstellung nach [71]

HTTPS fußt also auf dem Vertrauen des Clients zu einem Server. Dieses kann grundsätzlich durch drei Hauptkomponenten erfüllt werden: Verschlüsselung, Integrität und Authentifizierung. Die beiden ersteren werden dadurch garantiert, dass keine andere Partei – abgesehen von den Kommunikationsteilnehmern selbst – den Inhalt der Daten erfährt und damit auch nicht manipulieren kann. [19] Den Beweis der Authentizität erbringen die bereitgestellten Sicherheitszertifikate. Diese können in drei Gruppen unterteilt werden, welche sich im Hinblick auf Zweck und Funktion unterscheiden. Die erste und damit einfachste Ebene ist die der Domainvalidierung. Dabei wird lediglich die Kontrolle über eben diesen Domainnamen und nicht die Identität an sich geprüft. Solche Zertifikate sind demnach nur für Testserver, interne Links oder Non-Commerce-Seiten geeignet. Die zweite Ebene ist die der Unternehmensvalidierung. Zusätzlich zur Kontrolle über den Domainnamen muss ein Nachweis erbracht werden, dass die Organisation eingetragen und damit rechtlich haftbar ist. Damit eignet sich diese Art von Zertifikaten für öffentliche Webseiten, die auch persönliche Daten ihrer Nutzer einholen wollen. Die dritte Ebene ist die der erweiterten Validierung. Hinzu kommen zusätzliche Verifizierungsschritte, um diese Geschäftsinformationen auch dem Nutzer selbst zugänglich zu machen. Die beiden letzten Authentifizierungsgruppen

sind demnach Unternehmen und Organisationen vorbehalten – solche Zertifikate können also nicht von Einzelpersonen erworben werden. Ausgestellt werden diese letztlich von vertrauenswürdigen Zertifizierungsstellen. Falls es sich um keine primäre Certificate Authority handelt, muss zusätzlich das Zertifikat einer höhergestellten Zertifizierungsstelle vorliegen – der Nachweis für die Gültigkeit des Public Key. [21]

2.2 Web Application

Klassische Webseiten und Webanwendungen unterscheiden sich grundsätzlich im Grad der Interaktivität, Integration und Authentifizierung. Spricht man von einem interaktiven System, so werden dem Nutzer vielfältige Möglichkeiten geboten in Wechselbeziehung mit der Anwendung zu treten. Eine Webseite stellt in erster Linie Informationen bereit, wohingegen eine Webanwendung die Gelegenheit bietet den Inhalt auch zu bearbeiten – dadurch entsteht ein Dialog mit dem System selbst. In eine bestehende IT-Landschaft können sowohl Webseiten als auch Webanwendungen integriert werden – für letztere bietet es allerdings die Möglichkeit deren Funktionsumfang bedeutend zu erweitern oder Bearbeitungs- und Analyseprozesse zu optimieren. Im Fall von Webseiten ist eine Zugangsbeschränkung optional, im Fall von Webanwendungen hingegen obligatorisch. Nutzer werden also sowohl identifiziert als auch authentifiziert, um Unbefugten den Zugriff verwehren und sensible Daten optimal schützen zu können. [22]

2.2.1 Architektur und Funktionsweise

Eine Webanwendung ist nur deshalb so interaktiv nutzbar, da ihr Inhalt noch nicht oder nur teilweise festgelegt ist. Endgültig passiert dies erst zu dem Zeitpunkt, wenn der Nutzer eine Seite vom Server anfordert. Das lässt sich damit erklären, dass sich der Inhalt je nach den Aktionen eines Besuchers ändern kann – von Anforderung zu Anforderung. Solche dynamischen Seiten in Kombination mit statischen zeichnen also eine Webanwendung aus. Dabei können einerseits Informationen geliefert und aber andererseits auch von Nutzern, mithilfe von HTTP POST, übertragene Daten erfasst und gespeichert werden. [23]

Um das möglich zu machen und dynamische Seiten auch wirklich verarbeiten zu können, reicht eine einfache Client-Server-Architektur nicht mehr aus. Wird deren Inhalt über den Browser angefordert, muss dieser zuerst von einem sogenannten Applikationsserver zusammengesetzt werden – der wiederum greift dazu auf Hintergrundsysteme, wie zum Beispiel eine Datenbank, zu (Abbildung 6). Von diesen können weitere Informationen geholt werden – etwa solche, die von einem Nutzer erfasst oder von der Anwendung selbst, eventuell auch unter Berücksichtigung seiner Identität, abgelegt wurden. Daraus ergibt sich dann die sogenannte 3-Tier-Architektur. Moderne Webanwendungen können natürlich noch bedeutend vielschichtiger sein – dies ergibt sich grundsätzlich aber einfach aus einer differenzierteren Betrachtungsweise. [24]

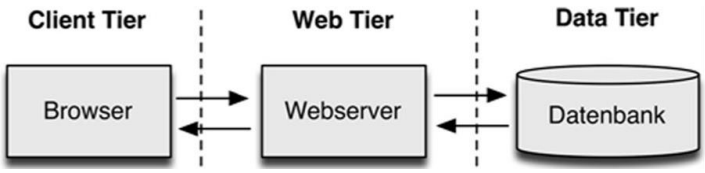

Abbildung 6: 3-Tier-Architektur einer Webanwendung

Quelle: [24]

Da Webanwendungen in der Regel aus mehreren solcher Seiten bestehen, muss sichergestellt werden, dass zwischen ihnen auch eine Verbindung besteht und sie untereinander auf irgendeine Art und Weise verknüpft sind. Dies geschieht in Form sogenannter Hyperlinks. Der letztendliche Aufbau einer Anwendung wird demnach durch die Gesamtheit solcher Verknüpfungen bestimmt. Die interne Verlinkung – also Verweise auf Seiten innerhalb der Onlinepräsenz – gibt damit eine grundlegende Navigation durch den Webauftritt vor. [25] Der Nutzerflow beschreibt also den Pfad, den ein Besucher zurücklegen muss, um die von der Webanwendung vorgesehene Aufgabe zu erfüllen – je nachdem, welche Rolle der einzelne User einnimmt, kann sich dieser Pfad allerdings ändern (Abbildung 7). Ein Administrator beispielsweise hat Zugriff auf Seiten, die allein ihm vorbehalten sind. Ähnlich wie ein authentifizierter Nutzer Zugang zum Rest der Anwendung erhält, während ein Gast lediglich die Startseite einsehen kann. [26]

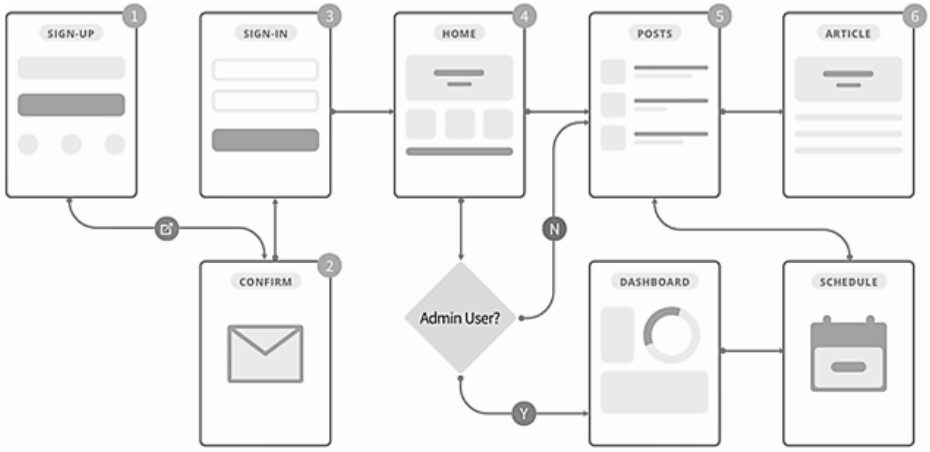

Abbildung 7: Nutzerflow-Diagramm

Quelle: [26]

2.2.2 Zustandsinformationen

Die Zustandslosigkeit von HTTP und die damit einhergehende Problematik, dass ein Nutzer nicht dauerhaft identifizierbar ist, widerspricht dem Gedanken, dass Webanwendungen eben eine Sammlung zahlreicher Webseiten sind, die erst zusammen einen bestimmten Zweck erfüllen. Die Lösung ist also, dass mehrere Request-Response-Zyklen zu einer einzigen Sitzung zusammengefasst werden. Zustandsinformationen – wie Anmeldedaten oder vorherige Aktionen – werden zwischengespeichert und ermöglichen so eine echte

Interaktion. [27] Abgelegt werden diese dann entweder auf dem Client selbst oder – zusammen mit einer eindeutigen Session-ID, um die Zuordnung zum jeweiligen Nutzer zu erleichtern – auf dem Server. [28]

Eine Möglichkeit, solche Informationen an den Server zu übermitteln, ist das Hidden Form Field. Wie sich dem Namen entnehmen lässt, handelt es sich um verborgene Formularfelder, welche vom Nutzer selbst nicht ohne Weiteres gesehen oder verändert werden können. [28] Prinzipiell handelt es sich eher um eine rein optische Lösung und garantiert keinerlei Form von Sicherheit. [29] So erlauben es beispielsweise die Entwicklertools eines Browsers Änderungen am HTML-Quellcode und damit auch den versteckten Feldern vorzunehmen. Ebenso funktioniert dies bei Cascading Style Sheets, kurz CSS – dabei handelt es sich dann um optische Modifikationen. [30]

Will ein Server seine Daten an den Browser übermitteln, kann er diese auch als Parameteranhang zur URL übergeben. Die Webanwendung ist somit in der Lage, eine Sitzung eindeutig wiederzuerkennen und damit verknüpfte Informationen erneut bereitzustellen. [28] Diese Schlüssel-Wert-Paare können jedoch ebenso einfach über die Adresszeile des Browsers manipuliert werden. Dabei ist es aber noch nicht einmal von Belang, ob sie zu diesem Zeitpunkt auch übertragen wurden – sobald ein Angreifer die Parameter kennt, kann er die Adresszeile entsprechend anpassen. [29]

Über einen speziellen HTTP-Response-Header – Set-Cookie – lassen sich solche Informationen ebenfalls an den Browser übermitteln [24]. Ein Cookie, wie er gemeinhin auch bezeichnet wird, existiert dabei grundsätzlich in zwei Auslieferungsmodi: persistent und nichtpersistent. Während Ersteres meint, dass er bis zum Ablauf der angegebenen Gültigkeitsdauer auf der Festplatte des Clients gespeichert wird, spricht man von Letzterem, wenn der Cookie nur im Speicher abgelegt und beim Schließen des Browsers wieder gelöscht wird. Da allerdings sowohl Speicherort als auch Dateiformat allgemein bekannt sind, lassen sich in diesem Fall nicht nur übermittelte Informationen einsehen und manipulieren, sondern ebenso die Gültigkeitsdauer eines Cookies. Da Webanwendungen dessen Daten ohne weitere Prüfung einfach übernehmen, lässt sich ein eigentlich beschränkter Zeitraum ohne Weiteres entweder verlängern oder verkürzen. [29] Sollte ein Nutzer sich dazu entscheiden, Cookies über die entsprechende Browser-Funktion zu deaktivieren, kann die Anwendung URL-Parameter als Fallback-Option nutzen. [28]

Wann immer also Zustandsinformationen selbst auf dem Client gespeichert werden, müssen diese zeitgleich auch auf dem Server vorliegen – mit dem Ziel die Daten abgleichen und so eine Manipulation von außen verhindern zu können. Um den Aufwand zu reduzieren, sollte also lediglich die entsprechende Session-ID auf dem Client vorgehalten werden. Zudem empfiehlt es sich, ein generelles Abgreifen solcher Daten grundsätzlich zu unterbinden – eine Ablage erfolgt nur in verschlüsselter oder gehashter Form. [29]

3 Zustandsbasierte Angriffe

Bei einem Risiko handelt es sich um "eine mit der Eintrittswahrscheinlichkeit und dem Schadenspotenzial bewertete Bedrohung, dass ein Sicherheitsproblem in einer Anwendung auftritt beziehungsweise ausgenutzt wird". Innerhalb eines Unternehmens lässt sich dieses nur dann identifizieren, wenn auch der Kontext einer Webanwendung bekannt ist. Es kann also durchaus vorkommen, dass eine Sicherheitslücke zwar aus technischer Sicht eine hohe Kritikalität besitzt, für das Unternehmen jedoch ein geringeres Risiko darstellt. [24]

Das Common Vulnerability Scoring System, kurz CVSS, hilft zumindest bei einer generellen Bewertung von Schwachstellen. Grundsätzlich werden dabei eben mögliche Schadensereignisse identifiziert und deren Eintrittswahrscheinlichkeit beziehungsweise daraus resultierende Schäden abgeschätzt. Mithilfe eines solchen Systems und klar vordefinierten Faktoren, kann dies auch möglichst objektiv beziffert werden. Die Bewertung von Sicherheitslücken erfolgt hierbei anhand sogenannter Metriken. Für jede davon gibt es festgelegte Wahlmöglichkeiten, aus denen sich ein Schweregrad von 0.0 bis 10.0 errechnen lässt – letzteres entspricht dabei dem höchstmöglichen. Diesen Werten können anschließend qualitative Kategorien – None, Low, Medium, High und Critical – zugeordnet werden. Die Metriken sind dabei in drei Gruppen unterteilt. Basis-Metriken beschreiben wesentliche unveränderliche Merkmale einer solchen Schwachstelle, woraus sich ein sogenannter Base Score errechnen lässt, der dann für den technischen Schweregrad steht. Indem dieser an zeitliche Veränderungen oder die jeweilige Umgebung des betroffenen Systems angepasst wird, kann er auch später noch nachjustiert werden. [31]

Auch im Zusammenhang mit dem Session Management existieren verschiedene Arten von Sicherheitslücken. Session-IDs behalten beispielsweise auch nach dem Logout ihre Gültigkeit und können für die Wiederherstellung einer Session genutzt werden. Sie sind nicht zufällig genug oder ändern sich auch mit erneuter Anmeldung nicht und lassen sich dadurch leicht erraten. Werden sie im Klartext übertragen – statt verschlüsselt über HTTPS – kann auch die Kommunikation abgehört und die Sitzung des gerade angemeldeten Nutzers mithilfe der gültigen Session-ID übernommen werden. [32]

3.1 Man-in-the-Middle-Angriffe gegen HTTPS-Verbindungen

Findet die Übertragung der Daten allerdings über eine geschützte Verbindung statt, muss diese Verschlüsselung zuerst gebrochen werden. Ein Man-in-the-Middle-Angriff erlaubt es, sich unbemerkt in eine solche Kommunikation einzuschleichen – um Informationen mitzulesen oder sie sogar zu manipulieren. Der Angreifer gibt sich dafür gegenüber dem Sender als Empfänger und auch gegenüber dem Empfänger als Sender aus. [33]

Ein derartiger Angriff manifestiert sich in vielen verschiedenen Szenarien – je älter die eingesetzte TLS-Version ist, desto anfälliger ist eine Webanwendung auch für solche Attacken. Jedoch ist selbst die neueste – also 1.3 – nicht vor allen Angriffen gefeit. [34] Nichtsdestoweniger senkt die fortwährende Unterstützung vor allem von Version 1.1 – immerhin 98,7% aller Nutzer verwenden derartige Browser – die Sicherheit, welche potentiell erreichbar wäre. [35]

3.1.1 SSL-Stripping

Selbst beim Wechsel von einer verschlüsselten Seite auf eine ungeschützte, sind die Daten, welche zwischen Client und Server ausgetauscht werden, normalerweise kodiert. Das liegt daran, dass die Nachrichten verschlüsselt werden, noch bevor sie verschickt werden. Spezielle Tools sind allerdings dazu in der Lage, Zugriff darauf zu bekommen – und zwar vor der Kodierung. Ein solches Vorgehen wird als SSL-Stripping bezeichnet. [36]

Moxie Marlinspike programmierte bereits 2002 ein entsprechendes Tool – sslsniff. Dabei handelt es sich um eine Proxy-Software, welche die Infiltration von SSL-Datenströmen und den Austausch des Serverzertifikats gegen ein beliebiges eigenes erlaubt. Diese positioniert sich zwischen Client und Server, um ausgelieferte Webseiten gezielt nach eingebetteten Links – mit einem Verweis auf einen SSL-geschützten Log-in – zu durchsuchen. Findet die Proxy-Software eine solche Weiterleitung, modifiziert sie diese zu einem äquivalenten HTTP-Link. Der Nutzer verschickt also anstelle der vermeintlich verschlüsselten Daten ganz gewöhnliche im Klartext. Dank sslstrip als Zwischenstation kann ein Angreifer problemlos mitlesen und so eventuell auch an vertrauliche Informationen gelangen. Der User bekommt das in der Regel gar nicht erst mit, da das SSL-Stripping keine ungültige Verbindung erzeugt und damit auch keine Warnmeldung erscheint. [36]

Zunächst ist es also notwendig, dass ein Proxy zwischen Browser und Webserver geschaltet wird. Nur, wenn die Software Datenströme abfangen kann, besteht auch die Möglichkeit, die per SSL-Stripping modifizierten URLs einzustreuen. Möglich ist dies unter anderem durch einen unbemerkten Eintrag in den Browser-Optionen. Malware beispielsweise sorgt dafür, dass ein externer Proxy-Server automatisch dort eingetragen wird – ohne dass der Nutzer darüber informiert wird. Innerhalb eines Subnetzes kann auch auf das sogenannte Spoofing zurückgegriffen werden, um einen Proxy-Server ins Spiel zu bringen. IP-Adressen müssen für die Identifikation von Netzwerkgeräten in entsprechende Hardwareadressen – Media-Access-Control-Adressen, kurz MACs – aufgelöst werden. Der Angreifer kann diese manipulieren und durch die seines eigenen Systems ersetzen, um anschließend die übertragenen Daten abzufangen. Die dritte und letzte Möglichkeit setzt voraus, dass das Gerät, von dem aus der Angriff gestartet wird, auch als Router fungieren kann. Damit können unter anderem auch IP-Adressen an Nutzer vergeben oder Datenpakete mitgelesen und weitergeleitet werden – sogar über die Grenzen des Subnetzes hinaus. [36]

Nachdem der Proxy-Server in Position gebracht wurde, lässt der Angreifer das Tool, welches in den passenden Situationen modifizierte Links ausgibt und im besten Fall unverschlüsselte Informationen liefert, einfach laufen. Weder Client noch Server haben nämlich die Möglichkeit, SSL-Stripping zu erkennen. Stattdessen gehen beide davon aus, dass sie mit dem eigentlich kontaktierten Partner kommunizieren – die Integrität der übertragenen Daten wird deshalb auch nicht angezweifelt. Auch für Nutzer verhält sich die Situation recht ähnlich. Präsentieren manipulierte Webseiten nicht gerade ein auffallend fehlerhaftes Layout oder treten erhebliche Verzögerungen bei deren Abruf auf, gibt es kaum Anhaltspunkte dafür, dass zu diesem Zeitpunkt keine Verschlüsselung stattfindet. Auch optische Hinweise – etwa die farbige Hinterlegung der Adresszeile eines Browsers, welche eine sichere Verbindung kennzeichnen oder gängige Symbole wie das Sicherheitsschloss – können manipuliert sein und garantieren keine absolute Sicherheit. [36]

Die einzige Möglichkeit eines Nutzers, um sich vor SSL-Stripping zu schützen, ist, den Aufbau verschlüsselter HTTPS-Verbindungen zu forcieren. Äußerst mühsam, aber erfolgreich ist die vollständige Eingabe der gesicherten URL per Hand. Wenn eine SSL-geschützte Webseite häufiger in Anspruch genommen wird, kann diese auch als Lesezeichen gespeichert und darüber aufgerufen werden. Dies setzt jedoch voraus, dass der Nutzer sich in einem sicheren Netzwerk befindet, wenn er das Lesezeichen setzt. Ansonsten besteht die Möglichkeit, dass eine bereits manipulierte URL in die Liste aufgenommen wird. Es gibt aber auch verschiedene Browser-Erweiterungen, welche automatisch verschlüsselte Versionen von Webseiten aufrufen – sofern diese vorhanden sind. Dazu wird beispielsweise auf Domain- und Regellisten zurückgegriffen, um jegliche Seitenaufrufe über gesicherte Verbindungen abzuwickeln. [36]

Auch Betreiber einer Webanwendung können SSL-Stripping aktiv bekämpfen. Grundsätzlich sollte die Verschlüsselung für sämtliche Seiten aktiviert und die Umleitung eingehender ungeschützter Verbindungen auf verschlüsselte forciert werden – das Gleiche gilt für gesetzte Cookies. Eine weitere Maßnahme ist der Einsatz des Sicherheits-Standards HTTP Strict Transport Security – kurz HSTS – welcher im RFC 6798 definiert ist. Dieser ermöglicht es Servern, den Clients mitzuteilen, dass sie die aufgerufene Website ausschließlich über eine HTTPS-Verbindung erreichen – zumindest für einen bestimmten Zeitraum. Im Response-Header wird dafür das Feld Strict-Transport-Security gesetzt – zuzüglich der Direktive max-age, mit welcher eine Gültigkeitsdauer festgelegt werden kann. Der Parameter includeSubDomains weitet den Befehl, wie der Name schon sagt, außerdem auf alle Subdomains der Webpräsenz aus. Erhält der Client vom kontaktierten Server also eine Nachricht mit Strict-Transport-Security-Anweisung, werden automatisch auch alle zukünftigen Verbindungen zu der entsprechenden Domain in verschlüsselte umgewandelt. Ansonsten erscheint eine Fehlermeldung und die angeforderte Seite kann nicht abgerufen werden. Auch wenn HSTS eine eigentlich dauerhafte Lösung darstellt, um Webanwendungen vor SSL-Stripping schützen zu können, benötigt selbst dieser Mechanismus einen allerersten Verbindungsaufbau – der ist allerdings ebenso manipulierbar. [36]

Um diesem Problem zu begegnen, existieren sogenannte Preload-Listen. Diese enthalten lediglich solche Webanwendungen, welche ausschließlich über HTTPS abrufbar sind. Für eine Aufnahme müssen allerdings bestimmte Voraussetzungen erfüllt sein: logischerweise ist ein gültiges Zertifikat erforderlich – über jegliche Subdomains. Darüber hinaus muss das HSTS-Feld folgende Informationen enthalten: die max-age-Direktive muss eine Gültigkeitsdauer von mindestens 18 Wochen aufweisen. Außerdem sind sowohl die Direktiven includeSubDomains als auch preload zu spezifizieren. [36]

3.1.2 Downgrade-Attacke

Ein Fallback erfolgt immer dann, wenn der Client ein älteres Gerät nutzt und die Interoperabilität zwischen ihm und dem Server damit verbessert werden kann. Das kann ein Angreifer ausnutzen – und zwar mit dem sogenannten Downgrade-Angriff. Grundsätzlich werden dadurch alle starken Algorithmen aus der Liste der vorgeschlagenen gelöscht und nur solche, welche gebrochen werden können, bleiben bestehen – die gesamte Sicherheit des Protokolls wird damit also herabgesetzt. Zur grundlegenden Abwehr dieser Attacke wird nach Abschluss der Aushandlung eine Authentizitätsprüfung über die gesamte Kommunikation durchgeführt – mithilfe der Berechnung eines MACs über alle bisher gesendeten und empfangenen Nachrichten. Ist dessen Verifikation nicht erfolgreich, so muss die Verhandlung als attackiert betrachtet und abgebrochen werden. [37]

Padding Oracle On Downgraded Legacy Encryption, kurz POODLE, ist eine solche Downgrade-Attacke – entdeckt und veröffentlicht von Bodo Möller, Thai Duong und Krzysztof Kotowicz im Jahr 2014. Eine erfolgreiche Durchführung ist auch gegen die TLS-Version 1.2 noch möglich, womit die Herabstufung auf SSL 3.0 erzwungen werden kann. Ausgenutzt wird dabei die Aushandlung des Protokolls während des Handshakes. Indem der Angreifer wiederholt versucht eine sichere Verbindung herzustellen und diese aber immer wieder abbricht, bringt er den Server automatisch dazu eine ältere Version anzubieten, um den Client entgegenzukommen. Im zweiten Schritt folgt ein Angriff auf die Verschlüsselung selbst. Die verwendeten Cipher Suites enthalten verschiedene Algorithmen dafür – unter anderem auch Blockchiffren. Dabei werden die Daten, wie der Name schon sagt, in Blöcken fester Größe verschlüsselt. POODLE macht sich dabei eine Schwachstelle im sogenannten Cipher-Block-Chaining-Modus, kurz CBC, zunutze. Zuerst wird der Klartext in Blöcke zerlegt – am Ende werden dann mehrere Bytes aufgefüllt, um sicherzustellen, dass auch alle Blöcke gleich groß sind. Da bei SSL 3.0 das MAC-then-encrypt-Verfahren verwendet, der Klartext-Nachrichtenauthentifizierungscode also vor der Auffüllung und der Verschlüsselung berechnet wird, sind diese Bytes ungeschützt. Dadurch, dass der Server die Anfrage akzeptiert, sobald das letzte Byte des vorhergehenden Blocks die korrekte Auffüllänge angibt, kann der Angreifer diesen Wert erraten. Mithilfe von XOR kann dann das entschlüsselte Byte mit diesem Block kombiniert und so das eigentlich letzte Byte des Klartexts aufgedeckt werden – eine Wiederholung dieses Vorgangs ermöglicht letztendlich das Aufdecken des gesamten Session-Cookies. Um POODLE zu verhindern, sollte die Unterstützung für SSL 3.0 auf beiden Seiten deaktiviert und stattdessen TLS_FALLBACK_SCSV

aktiviert werden. Dabei handelt es sich um eine Protokollerweiterung, welche garantiert, dass während einer Verhandlung niemals auf frühere Protokollversionen – zumindest keine, die unterhalb der höchsten vom Server unterstützten Version liegen – zurückgegriffen wird. SSL wird also nur verwendet, wenn Altsysteme beteiligt sind und der Fallback somit nicht auf einen Downgrade-Angriff – dieser zwingt den Server dazu, Versionen einfach zu überspringen – zurückzuführen ist. Diese einfach zu implementierenden Maßnahmen führen dazu, dass das Risiko dieses Angriffs nur noch gering eingeschätzt wird. [38]

Forscher des französischen Instituts Inria haben im Rahmen des Projekts State Machine Attack, kurz SMACK, außerdem verschiedene TLS-Implementierungen untersucht und eine Reihe von Sicherheitslücken zu Tage gefördert – unter anderem eine, welche sie FREAK, abgeleitet von Factoring RSA Export Keys, tauften. [39] Hierbei wird eine Funktion ausgenutzt, welche auch in der TLS-Version 1.0 noch Verwendung findet – bei Bedarf kann auf sogenannte Export-Cipher-Suites zurückgegriffen werden. Diese beschränken allerdings die mögliche Schlüssellänge von 2048 auf lediglich 512 Bits. Dadurch wird der Rechenaufwand für das Dekodieren verschlüsselter Nachrichten deutlich reduziert. Diese Unterstützung ist zwar standardmäßig deaktiviert, die anfängliche Verhandlung über das zu verwendende Verschlüsselungsverfahren kann jedoch manipuliert werden, indem dem Server der Wunsch nach einer solchen Export-Verschlüsselung vorgegaukelt wird. Dieser liefert dann einen schwachen, maximal 512 Bit langen RSA-Schlüssel zurück, welcher relativ schnell gebrochen werden kann. [40]

Auch Logjam nutzt diese Schwachstelle. Dabei wird allerdings nicht die RSA-Verschlüsselung angegriffen, sondern der Schlüsselaustausch zu Beginn einer Kommunikation. Bei der Zufallszahl, die der Client anfänglich erzeugt, handelt es sich um eine große Primzahl. Mithilfe bestimmter Operationen berechnen beide Kommunikationspartner daraus dann den geheimen Schlüssel. Auch hier existiert eine schwache Export-Variante – die Länge der Primzahl wird hierbei auf ebenfalls maximal 512 Bit heruntergesetzt, statt der eigentlich empfohlenen 1024. Betreiber von Webservern sollten also die Export-Optionen in ihren Cipher Suites deaktivieren und sicherstellen, dass letztere auch auf dem aktuellsten Stand sind. Vor dem Hintergrund steigender Rechenkapazitäten wird außerdem die Verwendung von Primzahlen mit einer Länge von 2048 Bit empfohlen. [40]

3.1.3 Cross-Protokoll-Attacke

Bei einer Cross-Protokoll-Attacke besteht ein Teil der Strategie darin, dass Unterschiede zwischen Protokollen ausgenutzt werden, um eine bestimmte Schwachstelle in Verschlüsselungsprotokollen wie SSL und TLS offenzulegen. [41]

Browser Exploit Against SSL/TLS – kurz BEAST– nutzt ebenfalls eine Schwachstelle in den TLS-Versionen 1.0 und älter, welche den CBC-Modus verwenden, aus und hat damit Ähnlichkeiten zu POODLE. Erstmals beschrieben wurde diese Attacke von Phillip Rogaway im Jahr 2002, jedoch erst 2011 von den beiden Sicherheitsforschern Thai Duong und Juliano

Rizzo auch tatsächlich demonstriert. BEAST beruht dabei auf der Vorhersagbarkeit der Initialisierungsvektoren, welche als Teil des Verschlüsselungsprozesses erzeugt werden. Aufgrund dessen und der festen Blockgröße können die Grenzen dieser Chiffrenblöcke manipuliert und der Klartext langsam enthüllt werden – und das, ohne ihn durch den Erhalt des Schlüssels dekodieren zu müssen. Aufgrund der besonderen Bedingungen – unter anderem muss eine JavaScript-Injektion möglich sein, sodass die Same-Origin-Policy einer Webseite außer Kraft gesetzt werden kann – ist es unwahrscheinlich, dass der Angriff überhaupt gelingt. Und solange diese Richtlinie nicht irgendwie umgangen wird, ist eine Injektion praktisch unmöglich. [42]

Compression Ratio Info-leak Made Easy, kurz CRIME, macht sich – wie der Name schon sagt – eine Schwachstelle in der Kompression von TLS zunutze. Nachdem Adam Langley, ein Software-Ingenieur bei Google, erstmals die Hypothese dazu aufstellte, präsentierten Juliano Rizzo und Thai Duong dann im Jahr 2012 auch eine offizielle Demo eines solchen Angriffs. Anfällig ist hier vor allem die TLS-Version 1.0 – diese verwendet das Komprimierungsverfahren DEFLATE, welches sich in diesem Fall als problematisch erwiesen hat. Dessen Algorithmus eliminiert nämlich doppelte Zeichenketten. Informationen über die ursprüngliche Größe des Chiffriertextes können genutzt werden, um diese mit denen über die komprimierten Nutzdaten – das geheime Session-Cookie, welches es zu ermitteln gilt, und ein injizierter bösartiger Inhalt sind darin ebenfalls enthalten – zu vergleichen. Durch Beobachtung, ob sich die Größe der komprimierten Daten dabei verringert hat – denn das bedeutet, dass der eingeschleuste Inhalt dem geheimen zumindest in Teilen gleicht – kann der Wert des Cookies möglicherweise bestimmt werden. Um den Angriff zu verhindern, reicht es also die TLS-Kompression zu deaktivieren. Auch wenn die Auswirkungen von mittlerer Stärke sind, kann das Risiko insgesamt als gering erachtet werden. [43]

Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext – kurz BREACH – basiert zwar grundsätzlich auf der Logik von CRIME, kann aber nicht durch das einfache Ausschalten der TLS-Komprimierung verhindert werden. Die Kompression findet in diesem Fall nämlich bereits auf der HTTP-Ebene statt. Deren Deaktivierung würde also lediglich zu größeren Seiten führen, die übertragen werden müssen – dies stellt demnach keine praktikable Lösung dar. Stattdessen sollte die Länge des Inhalts durch das Einfügen einer zufälligen Anzahl von Bytes verborgen werden. Damit der Angriff erfolgreich ist, müssen hierbei zusätzlich zu einem Geheimnis Benutzereingaben in der HTTP-Response wiedergegeben werden – dies sollte also voneinander getrennt werden. Um in diesem Zusammenhang einfache Brute-Force-Attacken zu unterbinden, kann außerdem die Anfragerate begrenzt werden. Das Risiko ist also deutlich höher einzuschätzen. [44]

Decrypting RSA using Obsolete and Weakened Encryption, kurz DROWN, nutzt eine Sicherheitslücke in der SSL-Version 2.0 aus. Wird dieses veraltete Protokoll noch unterstützt, kann der Angriff auch gegen moderne Server ausgeführt werden – es besteht also ein mittleres Risiko, dass die Schwachstelle ausgenutzt wird. Offiziell vorgestellt wurde er im Jahr 2016 – es handelt sich dabei ebenfalls um eine Version des POODLE-Angriffs, also eine Attacke auf den Chiffretext. Voraussetzung ist, dass RSA-Chiffre-Suites verwendet werden.

Zunächst müssen Sitzungen zwischen Client und Server aufgezeichnet werden. Eine davon kann irgendwann entschlüsselt werden. In der zweiten Phase fängt der Angreifer dann den Handshake ab und stellt dabei mehrere Verbindungen, welche SSLv2 unterstützen, zum Server her. Dabei handelt es sich um Nachrichten, die auf den Chiffretext abzielen – denn der in diesem Fall verwendete ungepolsterte RSA kann verändert werden. Durch gezieltes Ausprobieren gelangt der Angreifer an das Hauptgeheimnis und entschlüsselt damit den Session Key und auch die zuvor aufgezeichneten Sitzungen. Um einen solchen Angriff zu verhindern, muss also sichergestellt sein, dass die Verwendung von SSLv2-Chiffre-Suites nicht unterstützt wird. Außerdem darf der Private Key des Servers nirgendwo anders verwendet werden. [41]

Erneut aufgetaucht ist diese Sicherheitslücke unter dem Namen Return of Bleichenbacher's Oracle Threat, kurz ROBOT, im Jahr 2017. Hierbei werden allerdings Operationen mit dem Private Key eines Servers autorisiert, um so den Datenverkehr aufzeichnen und ihn anschließend entschlüsseln zu können. Server, die trotz Unterstützung der RSA-Verschlüsselungsmodi Forward Secrecy verwenden, sind nicht so stark gefährdet. Die Durchführung an sich ist zwar plausibel, müsste aber schnell ausgeführt werden. [45]

Heartbleed wurde im Jahr 2014 als schwerwiegende Schwachstelle in der Verschlüsselungssoftware OpenSSL öffentlich bekannt und als eine der kritischsten des letzten Jahrzehnts eingestuft. Sie ist leicht auszunutzen und schwer zu erkennen. Es handelt sich hierbei nämlich um einen Pufferüberlauf – ein System lässt den Zugriff auf Daten zu, obwohl dieser eigentlich beschränkt sein sollte. So wird es Angreifern ermöglicht, den Private Key des Servers zu stehlen. OpenSSL bietet die Option, einen Heartbeat zu senden – um zu überprüfen, ob ein anderer Computer online ist, kann eine Nachricht verschickt und auf einen Beat gewartet werden. Angreifer können diese Funktion nutzen, um eine bösartige Nachricht statt einer normalen zu senden. Der Empfänger kann so dazu gebracht werden, geheime Daten zu akzeptieren und auch zu übertragen – einschließlich des Speichers. Bei Verwendung von OpenSSL genügt es aber auf die Version zu achten und diese bei Bedarf zu updaten. [46]

Application Layer Protocols Allowing Cross-Protocol Attacks – kurz ALPACA – ist ein eher theoretisches Problem, dennoch wird davor gewarnt, dass dieser oder ähnliche Angriffe in Zukunft für Probleme sorgen könnten. Hintergrund ist, dass TLS nicht zwischen Protokollen unterscheidet – zumindest nicht bei der Datenübertragung. Eine Confusion-Attacke könnte so dazu führen, dass unterschiedliche Protokolle miteinander reden. Eine Voraussetzung ist allerdings, dass der Domainname einer Webseite mit beispielsweise dem eines E-Mail-Servers identisch ist. Jedoch gibt es bereits eine Sicherheitsmaßnahme: die Aktivierung der TLS-Erweiterung Application Layer Protocol Negitiation. Diese sorgt dafür, dass sich die Kommunikationspartner bei den Protokollen abstimmen. Kommt also ein nicht erlaubtes Protokoll zum Einsatz, kann so eine Verbindung abgebrochen werden. [47]

3.1.4 Kollisionsattacke

TLS verwendet Hash-Funktionen, um signierte Zertifikate zu erstellen. Hashes sind eindeutige Werte, welche auf der Eingabe von Daten basieren. Selbst kurze, zufällige Strings, die in eine solche Funktion eingegeben werden, geben eine feste Anzahl von Zeichen zurück. Das ist vor allem deshalb nützlich, weil sich leicht erkennen lässt, ob eine Eingabe sich geändert hat. Damit soll also lediglich die Datenintegrität gewährleistet werden. [48]

Theoretisch sollten Hash-Funktionen eine eindeutige Zeichenfolge für alle eingegebenen Daten erstellen. Mit zunehmender Anzahl solcher Hashes steigt jedoch auch die Wahrscheinlichkeit, dass verschiedene Eingaben dieselbe Ausgabe generieren. So lässt sich beispielsweise ein nicht vertrauenswürdiges Zertifikat erstellen, welches einen identischen Hash wie das entsprechend vertrauenswürdige besitzt. Das Finden solcher übereinstimmenden Ausgabewerte wird dann als Kollisionsangriff bezeichnet. [48]

Karthikeyan Bhargavan und Gaëtan Leurent haben mehrere Schwachstellen veröffentlicht, die sich auf TLS beziehungsweise die Nutzung alter Hashfunktionen – in diesem Fall MD5 und SHA1, welche inzwischen als überholt gelten – beziehen. Security Losses from Obsolete and truncated Transcript Hashes, kurz SLOTH ist dabei also ein Verweis darauf, dass veraltete Algorithmen zu langsam aus den verwendeten Protokollen entfernt werden. Einer der relevantesten solcher Angriffe betrifft die Authentifizierung mit Clientzertifikaten in der TLS-Version 1.2. Denn auch ein Nutzer hat die Möglichkeit sich mit einem Zertifikat beim Server auszuweisen, wenn dieser das verlangt. In diesem Fall handelt es sich bei letzterem allerdings um den des Angreifers. Der kann sich dann gegenüber einem anderen Server mit dem Zertifikat dieses Clients anmelden – vorausgesetzt der unterstützt das Signaturverfahren mit MD5. Die älteren Versionen bleiben davon nur deshalb verschont, weil sie dafür eine Kombination aus MD5 und SHA1 nutzen. Später wurde dies dann aber durch einen Mechanismus ersetzt, bei dem der verwendete Algorithmus zwischen Server und Client erst noch ausgehandelt werden muss. Auch wenn MD5 und SHA1 bereits zu diesem Zeitpunkt durch etliche Angriffe untergraben worden waren, sind sie auch heute immer noch zulässig. Und selbst, wenn ein Server angibt, solche Signaturen nicht zu unterstützen, gibt es zahlreiche TLS-Implementierungen, die sie dennoch akzeptieren. [49]

Der SWEET32-Angriff wiederum nutzt eine Sicherheitslücke aus, wonach die in kryptografischen Protokollen verwendeten Blockchiffren – vor allem mit einer Länge von 64 Bit – anfällig für Kollisionen sind. Algorithmen wie Triple-DES und Blowfish sind im Rahmen der Verschlüsselung gängiger Sicherheitsprotokolle zwar weit verbreitet, die Wahrscheinlichkeit eines solchen Angriffs ist allerdings eher gering einzuschätzen. Für dessen Erfolg müssen nämlich weitere Bedingungen – wie eine längerfristige Überwachung des Datenverkehrs und die Ausführung von JavaScript im Browser des Clients, um die Anfragen an eine vom Angreifer kontrollierte Webseite weiterzuleiten – erfüllt sein. Trotzdem machen kurze Blockgrößen einen Server dafür anfällig, denselben Hash für mehrere Eingabewerte zu verwenden. Das liegt daran, dass sich dessen Länge von der des verwendeten Schlüssels zwar unterscheidet, erstere dabei aber durch den Algorithmus selbst festgelegt wird und

letztere damit begrenzen kann. Allerdings hängt es nicht nur von der Schlüsselgröße ab, sondern auch von der Datenmenge. Eine solche Chiffre kann nämlich nur eine bestimmte Anzahl an Blöcken mit demselben Schlüssel kodieren, bevor es zu Kollisionen in der Ausgabe kommt – also ein identischer Chiffretext generiert wird. Bevor Karthikeyan Bhargavan und Gaëtan Leurent im Jahr 2016 diese Schwachstelle veröffentlichten, waren etliche Kryptographen schon darauf aufmerksam geworden. Ausmaß und Geschwindigkeit, mit der sie ausgenutzt werden kann – nämlich in weniger als zwei Tagen – wurden aber erst durch die beiden enthüllt. Trotz dessen, dass Webserver so konfiguriert sein müssten, dass sie stärkeren Verschlüsselungssuiten den Vorzug geben, waren auch etliche hochkarätige Seiten unter den anfälligen dabei – das Risiko war und ist also gegeben. [50]

3.1.5 Seitenkanalattacke

Eine Seitenkanalattacke – erstmals im Jahr 1996 dokumentiert von dem amerikanischen Kryptologen Paul C. Kocher – bezeichnet einen Angriff, bei dem Algorithmen oder Daten nicht direkt angegriffen werden. Stattdessen werden physikalische und logische Nebeneffekte eines Systems ausgenutzt, um erst durch deren Analyse Informationen zu erhalten, die das tatsächliche Angriffsziel betreffen. Beispielsweise existieren Methoden, mit denen die Speichernutzung einzelner Prozesse analysiert oder die Zeit, die ein System benötigt, um eine bestimmte Aktion auszuführen, gemessen wird. Dabei kann also sowohl aktiv als auch passiv vorgegangen werden – entweder kann in den Ablauf eines Systems eingegriffen werden, um die Ausführung einer Funktion zu veranlassen, oder dessen Reaktion lediglich beobachtet werden. Das wiederum lässt dann aber Rückschlüsse auf die Funktionsweise und den verwendeten Algorithmus zu. Eines haben die Methoden jedoch in der Regel gemeinsam: um verwertbare Informationen zu gewinnen, benötigen sie viele Einzeldurchläufe. Dadurch, dass diese Attacken häufig auf der Analyse mehrerer Kanäle basieren, wirken Gegenmaßnahmen jeweils nur gegen eine Angriffsmethode. [51]

LUCKY13 beispielsweise ist ein solcher Timing-Angriff, welcher die TLS-Versionen 1.2 und älter betrifft. Es handelt sich hierbei um ein Problem der Protokoll-Spezifikation und nicht einer bestimmten Implementierung. Voraussetzung ist allerdings, dass der CBC-Operationsmodus verwendet wird – der Angriff kann also als eine fortgeschrittene Art von POODLE betrachtet werden. Entdeckt wurde er von den beiden Forschern Nadhem AlFardan und Kenny Paterson und verdankt seinen Namen den 13 Bytes der Header-Informationen zur TLS-MAC-Berechnung. Je nachdem, ob der Klartext also um ein oder zwei Bytes aufgefüllt wurde beziehungsweise ein fehlerhaft formatiertes Padding enthält, kann ein Unterschied in der Verarbeitungszeit zwischen den Nachrichten festgestellt werden. Mithilfe der Timing-Daten kann dann ein Klartext-Wiederherstellungsangriff auf die MAC-Prüfung durchgeführt werden, während ein fehlerhaftes CBC-Padding verarbeitet wird. Da für den Angriff einige Bedingungen erfüllt sein müssen – es ist beispielsweise erforderlich, dass sich der Angreifer im selben LAN befindet und genügend Sitzungen generiert werden, um die Zeitsignale der Session vom Netzwerkrauschen unterscheiden zu können – stellt LUCKY13 für die meisten Nutzer keine ernsthafte Bedrohung dar. Dennoch können Sicherheitsmaßnahmen – wie

beispielsweise das Hinzufügen zufälliger Zeitverzögerungen zum Entschlüsselungsprozess oder die grundlegende Umstellung auf Advanced-Encryption-Standard-Chiffren – etabliert werden, um eine statistische Analyse zu erschweren. [52]

HTTP Encrypted Information can be Stolen through TCP-windows, kurz HEIST, beruht grundsätzlich auf BREACH. Allerdings muss in diesem Fall der verschlüsselte Datenverkehr gar nicht mitgelesen werden – stattdessen genügt dem Angreifer beispielsweise eine bösartige Anzeige in der attackierten Webseite. Diese enthält JavaScript-Code, mit dem die Zeit für eine TCP-Antwort gemessen werden kann. Die Grundlage dieses Timing-Angriffs sind die Application Programming Interfaces, kurz APIs, Fetch und Resource Timing. Damit kann JavaScript die Ladezeiten einzelner Bestandteile einer Webseite beinahe exakt ermitteln. Diese ändern sich nämlich allein durch eine minimale Zunahme der übertragenen Daten, wenn die Größe eines TCP-Windows damit überschritten wird. Eine Änderung wird dadurch erreicht, dass sich wiederholende Daten, wie beschrieben, besser komprimieren lassen. Indem der Angreifer einfach eine Zeichenfolge rät und dafür sorgt, dass diese in der Webseite auftaucht, kann er testen, ob sie einen bestimmten Wert hat – das komprimierte Resultat ist bei einem richtigen Test schließlich kleiner als bei einem falschen. Ob HEIST auch in der Praxis wirklich relevant wird, lässt sich noch nicht sagen. Dennoch empfehlen die Forscher als Schutz die Unterstützung von Third Party Cookies abzuschalten, auch wenn es bei einigen Seiten zu Problemen führen könnte. [53]

Eine, wenn auch bis jetzt für die Praxis eher irrelevante, Angriffsmethode wurde unter dem Namen Raccoon veröffentlicht. Betroffen sind dabei ebenfalls die TLS-Versionen 1.2 und älter. Während der Erzeugung des Premaster-Secrets – also der beim Handshake vom Client zufällig generierten Zahl – werden die Ergebnisse jeweils gehasht. Bevor das geschieht, werden führende Nullen entfernt – dies wiederum spiegelt sich aber in der Eingabelänge des Hashwerts wider, was dann abermals zu einem messbaren Zeitunterschied führt. Ein einmal abgefangener Schlüsselaustausch lässt sich modifiziert mehrfach an den Server zurückschicken, woraus sich nach und nach Informationen ableiten lassen und zuletzt der Schlüssel selbst berechnet werden kann. Der Angriff kann also nur gelingen, wenn dabei immer derselbe Public Key genutzt wird – diese Vorgehensweise ist mittlerweile jedoch vergleichsweise selten. Die Schwachstelle wird daher lediglich mit einem niedrigen Schweregrad bemessen [54]. Da sie allerdings zeigt, dass Seitenkanalattacken bei der Entwicklung eines Protokolls bisher nicht ausreichend berücksichtigt wurden, ist die Forschung grundsätzlich trotzdem relevant. [55]

3.2 Session Hijacking

Ein Session-Hijacking-Angriff beschreibt grundsätzlich den erfolgreichen Versuch eines Angreifers, eine Session zu übernehmen. Dazu kann er sich als autorisierter Nutzer ausgeben, um Zugang zu einer Webanwendung zu bekommen. Eine solche Entführung kann also zu jedem Zeitpunkt einer noch andauernden Sitzung erfolgen – durch die Übernahme einer aktiven Kommunikation zweier Geräte. [56]

Durch einen Man-in-the-Browser-Angriff beispielsweise kann ein Nutzer dazu gebracht werden, Malware auf seinem System zu installieren – durch Aufrufen einer infizierten URL. Cross-Site-Scripting zum Beispiel erlaubt es, clientseitige Skripte in Webseiten einzuschleusen. Sobald dieser Link angeklickt wird, wird die Malware installiert. Diese erhält dadurch die entsprechende Session-ID des Nutzers, um den eigentlich unbefugten Zugriff auf die Webanwendung zu erhalten. [56]

Session Fixation funktioniert ähnlich – allerdings wird der Link genutzt, um einen Nutzer aktiv dazu aufzufordern, sich bei einer Webanwendung anzumelden. Sobald ihm der authentifizierte Zugang gewährt wurde, übergibt er den Zugriff unbeabsichtigt an den Angreifer, welcher damit die Kontrolle erhält. [56]

Beim Session-Side-Jacking wird der Netzwerkverkehr des Nutzers mithilfe von Packet Sniffing überwacht. Wenn eine HTTPS-Verschlüsselung also einmal gebrochen ist, kann eine Session-ID auch auf diese Weise gefunden und damit die Kontrolle über eine Sitzung übernommen werden. [56]

Verhindert werden kann Session Hijacking hauptsächlich dadurch, dass Transaktionen oder die Eingabe persönlicher Informationen nur über mit HTTPS verschlüsselte Webseiten getätigt und keine Links aufgerufen werden, deren Ursprung nicht gesichert ist. Auch öffentliche Netzwerke sollten vermieden oder anderenfalls ein virtuelles privates Netzwerk verwendet werden – dieses sorgt für eine zusätzliche Verschlüsselung der Verbindung. Außerdem können zusätzlich Tools, welche die Erkennung und Entfernung von Viren unterstützen beziehungsweise Schutz gegen Malware bieten, eingesetzt werden. [56] Stand Juli 2022 existieren dennoch 499 bekannte Schwachstellen im Zusammenhang mit Session Hijacking, welche sogar kritische Auswirkungen haben können. [57]

3.3 Cross-Site-Request-Forgery

Eine ähnliche Schwachstelle ist Cross-Site Request Forgery, kurz CSRF – auch Session Riding genannt. Erstmals erwähnt wurde sie bereits 2001, dies blieb allerdings ohne nennenswerte Folgen. Ein erhebliches Sicherheitsrisiko geht von ihr dennoch aus. Stand Juli 2022 gibt es immerhin 2293 Sicherheitslücken, die damit in Verbindung stehen [58]. Das liegt daran, dass CSRF häufig explizit verhindert werden muss. [24]

Sobald ein an der Webanwendung angemeldeter Nutzer eine zugehörige URL aufruft, wird ein HTTP-Request an den entsprechenden Webserver gesendet. Allein anhand der übertragenen Session-ID kann diesem dann eine bereits bestehende Sitzung zugeordnet werden. Selbst bei einer Trennung der Netzwerkverbindung und einer erneuten Anfrage, bleibt die ID erhalten und wird automatisch mitgesendet, bis sie invalidiert oder erneuert wird. Die verwendete URL ist allerdings generisch, also für jeden Nutzer identisch. CSRF nutzt diesen Sachverhalt aus – einfach, indem einem angemeldeten User eine URL untergeschoben wird (Abbildung 8). Sobald dieser dann auf den Link klickt, wird das bereits gültige Session-

Cookie erkannt. Die Aktion wird also über das Profil des jeweiligen Nutzers durchgeführt. Solche Angriffe sind demnach überall dort durchführbar, wo der Zustand einer Anwendung über eine authentifizierte Session geändert wird. [24]

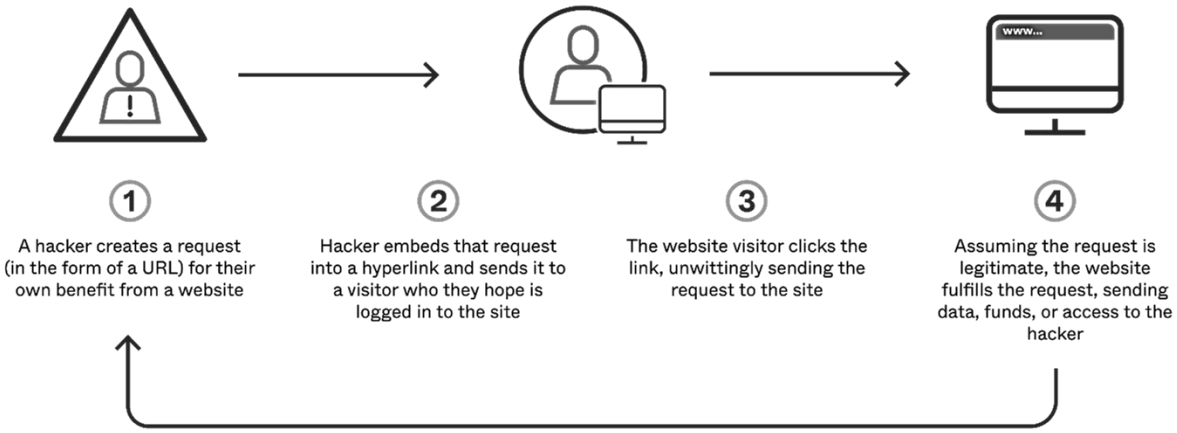

Abbildung 8: Ablauf einer Cross-Site-Request-Forgery-Attacke

Quelle: in Anlehnung an [24]

Um einen Nutzer also dazu zu verleiten, eine bestimmte URL aufzurufen, existieren verschiedene Wege – etwa das Posten eines solchen Links in einem Forum, ob als Kommentar oder in einem Kontaktformular. Dieses Vorgehen erhöht nämlich die Wahrscheinlichkeit, dass derjenige User auch tatsächlich an der Anwendung angemeldet ist. Kann ein Angreifer die URL eines eingebundenen Bildes bestimmen oder sogar HTML-Markup selbst in die Seite einbauen, so kann die Attacke automatisch ausgeführt werden – allein dadurch, dass einem Nutzer die entsprechende Webseite angezeigt wird. Die Anfrage wird also einfach ausgeführt – dass es sich bei der aufgerufenen Ressource um kein Bild handelt, wird schließlich nicht geprüft. Auch, wenn eine Anwendung Änderungen nur über POST erlaubt, ist ein solcher Angriff möglich. Hierfür muss der Angreifer sein Opfer allerdings zunächst entweder auf eine speziell präparierte Webseite mit eingebettetem Formular locken oder, wenn möglich, eine Cross-Site-Scripting-Schwachstelle ausnutzen – der Angriff kann dann per JavaScript in die Seite selbst eingebaut und auch dort ausgeführt werden. Durch diese Kombination gewinnt CSRF entscheidend an Kritikalität. [24]

Eine grundlegende Maßnahme zur Verhinderung besteht darin, keine generischen URLs für solch HTTP-basierte Änderungen zu verwenden. Es braucht also nur ein zufälliges Request-spezifisches Token – häufig CSRF-Token genannt – welches in den Aufruf eingebaut wird. Einem Benutzer kann somit kein Request mehr untergeschoben werden. Änderungen sollten aber nach wie vor ausschließlich mittels POST möglich sein. [24]

3.4 URL-Jumping

Die eigentlich vorgesehene Reihenfolge, in der eine Webanwendung durchlaufen werden soll, kann per URL-Jumping verlassen werden. Funktionen, die also noch gar nicht

zugänglich sein sollten, können damit bereits genutzt werden. Schließlich kann eine einzelne Seite besucht werden, indem schlichtweg die zugehörige URL in die Adresszeile des Browsers eingegeben wird. Genau davon profitiert aber ein potentieller Angreifer – bestimmte Schritte wie eine Anmeldung können damit einfach übersprungen werden. Ist die gewünschte Reihenfolge also nirgends festgelegt, muss damit gerechnet werden, dass diese Schwachstelle ausgenutzt wird. [29]

Die letzte tatsächlich besuchte Seite muss demnach mit der verglichen werden, die auch an entsprechender Stelle einer vorgesehenen Sequenz sein sollte. Für die Übertragung von ersterer gibt es grundsätzlich vier Optionen: versteckte Felder, Parameter, Cookies und Eintragungen im Referer-Feld des HTTP-Headers. Bis auf letztere haben sich alle anderen bereits als mehr oder weniger unsicher erwiesen – die Möglichkeit einer Manipulation besteht jedenfalls durchaus. [59] Das Referer-Feld allerdings wird vom Browser automatisch gesetzt. Jedoch ist es nicht in jedem Fall vorhanden – so zum Beispiel, wenn die entsprechende Seite über ein Lesezeichen beziehungsweise eben direkt über die Adresszeile aufgerufen oder dies aus Datenschutzgründen untersagt wurde. Existiert der Wert, handelt es sich dabei um diejenige Seite, von wo aus auf die aufgerufene verlinkt wurde. Allerdings kann dieser ebenfalls verändert werden, wenn auch nicht direkt im Browser – da der Header unverschlüsselt übertragen wird, ermöglicht stattdessen ein Proxy eine ebenso problemlose Manipulation. [29]

Die sicherste Art eine Reihenfolge festzulegen, ohne dass sie umgangen werden kann, ist also auch in diesem Fall sie auf dem Server zu speichern und lediglich eine Session-ID auf dem Client abzulegen. Ist dies nicht möglich, sollte trotzdem das Referer-Feld bevorzugt werden – da es automatisch befüllt wird, ist die Gefahr geringer, dass ein Angreifer sofort darauf aufmerksam wird. [59]

3.5 Clickjacking

Clickjacking wurde im Jahr 2008 von Jeremiah Grossman und Robert Hansen entdeckt und auch benannt. Der zugrundeliegende Vorgang wird als Framing bezeichnet – basierend auf dem Einsatz von sogenannten Iframes (Abbildung 9). Das Ziel ist es den Nutzer zu täuschen, indem dieser auf einen Button in der Webanwendung des Angreifers klickt – die tatsächliche Aktion findet aber in der eigentlichen statt. Dieses Iframe ist nämlich eigentlich das tatsächlich vorgelagerte und wurde nur durch eine CSS-Anweisung transparent gemacht. [24] Wie man dem Namen entnehmen kann, dient das Vorgehen also dem Abfangen von Klicks, um Aktionen ausführen zu können, auf die der Angreifer normalerweise keinen Zugriff hätte. [29]

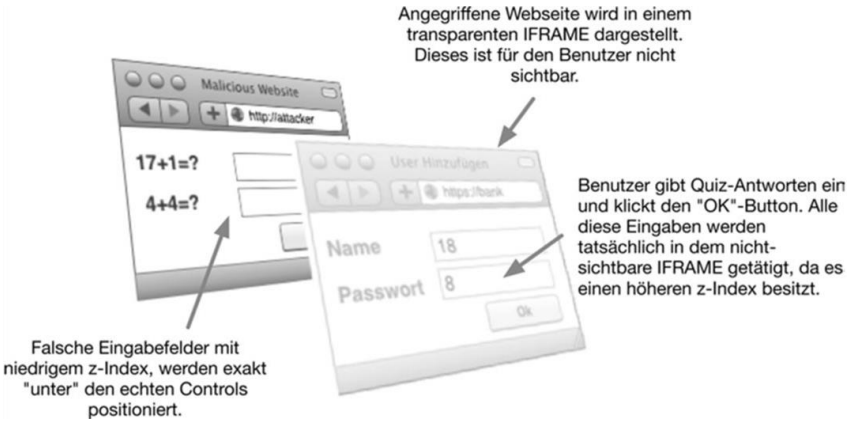

Abbildung 9: Ablauf einer Clickjacking-Attacke

Quelle: [24]

Der erste bekannte Angriff dieser Art richtete sich gegen Facebook und bekam den Namen Likejacking. In diesem Fall wurde es ausgenutzt, dass nicht nur Kommentare, sondern auch Profile gelikt werden können. So auch das der Angreifer, indem der entsprechende Button einfach vor den eigentlichen gelagert und unsichtbar gemacht wurde. Damit kann nicht nur das Profil selbst verbreitet werden – das Vertrauen von Facebook-Freunden dem gegenüber steigt aufgrund des Likes – davon kann im Nachhinein auch weiter Schadsoftware ausgehen. [29]

Stand Juli 2022 existieren insgesamt 177 veröffentlichte Schwachstellen, welche Clickjacking ermöglichen. Wie bereits beschrieben, geht die erstmalige Entdeckung einer solchen Vulnerabilität auf das Jahr 2008 zurück. Das Angriffsziel war in diesem Fall der Adobe Flash Player (CVE-2008-4503). Unter Vorspiegelung falscher, für das Laufen der Applikation notwendiger, Kontrollelemente, konnte die tatsächliche Freigabe von Webcam und Mikrofon verschleiert werden. [60]

Um eine entsprechende Schwachstelle ausfindig machen zu können, muss eruiert werden, ob eine Webanwendung überhaupt Seiten besitzt, die sich dafür eignen. Das sind solche, auf die ein Angreifer eben normalerweise keinen Zugriff hat und deren implementierte Funktionen nur von authentifizierten Nutzern ausgelöst werden können. Natürlich ist es in diesem Fall auch möglich alle Seiten, nicht nur die gefährdeten, zu schützen. [29]

Dies erreicht man damit, dass man das Einbinden der betroffenen Webseite in einem Frame unterbindet. Dazu existieren grundsätzlich drei Möglichkeiten. Der Einsatz eines Framebusters ist eine Option. Wird die Webanwendung in einem Frame geladen, befreit dieser sie daraus und erzwingt ihr Öffnen unmittelbar im Browserfenster. Iframes verfügen mittlerweile über ein sandbox-Attribut, welches den Framebuster aushebelt, indem der Zugriff auf die Top-Level-Navigation verboten wird. JavaScript-Code kann jedoch ansonsten weiterhin ungehindert ausgeführt werden. In Kombination mit dem X-Frame-Options-Header lässt sich ein Clickjacking-Angriff allerdings nach wie vor verhindern. Dieser informiert den Browser

darüber, ob und falls ja von wem eine Webanwendung in einen Frame eingebunden werden darf (Tabelle 6). [29]

Tabelle 6: X-Frame-Options eines HTTP-Headers

Quelle: in Anlehnung an [29]

deny	der Browser stellt die Seite grundsätzlich nicht dar, wenn sie in einem Frame geladen wird
sameorigin	die Seite wird nur dann in Frames dargestellt, wenn der Top-Level-Kontext von der gleichen Origin wie die Seite stammt
allow-from origin	die Seite wird dann in Frames dargestellt, wenn der Top-Level-Kontext von der Origin origin stammt

4 Web Application Security

Sogenannte Assets – also Güter eines Unternehmens sowohl materiellen als auch immateriellen Werts – stehen im Zentrum der IT-Sicherheit. Diesen kann immer ein konkreter Schutzbedarf zugeordnet werden, welcher sich dann an den verschiedenen Zielen bemisst (Tabelle 7). Der individuelle Aufwand richtet sich also immer nach dem Wert, der Risikohöhe und der Geschäftskritikalität eines bestimmten Assets. Insbesondere dann, wenn eines der ersten beiden Schutzziele betroffen ist und bereits von einer Kompromittierung der Daten oder Systeme gesprochen wird, muss eine Priorisierung stattfinden. [24]

Tabelle 7: Primäre Schutzziele der IT-Sicherheit

Quelle: in Anlehnung an [24]

Vertraulichkeit	Schutz vor unbefugter Preisgabe sensibler Informationen
Integrität	Korrektheit beziehungsweise Unversehrtheit der Daten
Verfügbarkeit	Geschäftsprozesse stehen stets wie gewünscht zur Verfügung

Web Application Security stellt dabei eine Unterdisziplin der IT-Sicherheit dar und beschäftigt sich – wie der Name schon sagt – mit eben solchen Aspekten, die den Schutz webbasierter Anwendungen beziehungsweise derer Komponenten und Dienste betreffen. Dabei kann jedoch immer nur von einem Sicherheitsniveau gesprochen werden – dieses kann anhand unterschiedlicher Kriterien beurteilt und gemessen werden. Eine Webanwendung kann also nicht als sicher eingestuft werden, nur weil es eine Vielzahl unterschiedlicher Schutzmaßnahmen implementiert oder keine bekannten Schwachstellen besitzt – damit erreicht es lediglich ein angemessenes Niveau. [24]

Sicherheit sollte in erster Linie nach den nicht-funktionalen Eigenschaften beurteilt werden – diese machen die Qualität einer Anwendung aus. In einer Veröffentlichung des US-Verteidigungsministeriums werden solche sicherheitsrelevanten Qualitätskriterien beschrieben (Tabelle 8). Im Fall der Zurechenbarkeit und Resistenz wird häufig auch von der Robustheit einer Anwendung gesprochen. [24]

Tabelle 8: Sicherheitsrelevante Qualitätskriterien

Quelle: in Anlehnung an [24]

Vertrauenswürdigkeit	die Webanwendung ist frei von Sicherheitsmängeln
Zurechenbarkeit	die Anwendung arbeitet korrekt und wie vorgesehen
Resistenz	der Schaden ist, im Falle einer Kompromittierung, minimal und die Anwendung in einer tolerierbaren Zeit wieder betriebsbereit
Konformität	die Webanwendung sowie deren Erstellungsprozess sind im Einklang mit geltenden Standards oder Gesetzen

Dabei kann zwischen drei Arten von Sicherheitsanforderungen unterschieden werden: funktional, nicht-funktional und Assurance. Nicht-funktional beschreibt hierbei eine eindeutig formulierte Eigenschaft, die ein System erfüllen muss – wie die Unterbindung eines nicht autorisierten Zugriffs. Bei der funktionalen Anforderung handelt es sich dann um die Erweiterung eben dieses Systems um eine konkret spezifizierbare Sicherheitsfunktion – also beispielsweise die Übertragung eines verschlüssel-ten Session-Cookies über eine HTTPS-Verbindung. Nachdem diese implementiert wurde, kann die Webanwendung als Ganzes auf ihre Sicherheit geprüft werden – etwa durch erforderliche Tests. Findet sich dann eine weitere Schwachstelle, wird der Prozess wiederholt – eben so lange, bis ein angemessenes Sicherheitsniveau erreicht wurde. [24]

4.1 IT-Grundschutz

In Deutschland ist für Empfehlungen bezüglich der IT-Sicherheit das Bundesamt für Sicherheit in der Informationstechnik zuständig. Der veröffentlichte Grundschutzkatalog definiert dabei eine Art Basisschutz für die gesamte IT-Infrastruktur. Seit dem Jahr 2012 existiert auch ein Baustein, der Webanwendungssicherheit abdeckt. Zusätzlich wurden der Best Practice Guide für Sichere Webanwendungen und eine weitere Leitlinie zu deren IT-Sicherheit veröffentlicht. Diese beschreiben zwar nur allgemeine Best Practices, bieten Entwicklern aber eine wichtige Hilfestellung bei der Implementierung verschiedener Anwendungsfunktionen. Einen verpflichtenden Standard stellen die Dokumente jedoch nicht dar. [24]

Sicherheitsuntersuchungen haben zum Ziel, die Umsetzung konkreter Sicherheitsanforderungen beziehungsweise deren Wirksamkeit zu verifizieren. Hierbei muss allerdings die Verifikation von der Validierung unterschieden werden: im Rahmen von ersterer wird lediglich die Umsetzung konkreter Sicherheitsvorgaben geprüft, nicht deren Sinnhaftigkeit. Diese erfolgt im Rahmen der Validierung. Viele funktionale Anforderungen lassen sich auch durch Anwendungstests verifizieren – meistens sogar automatisiert.

Penetration Testing stellt hierbei eines der beliebtesten Verfahren für die Sicherheitsanalyse von Webanwendungen dar. Ein Tester versetzt sich hierbei in die Rolle des Angreifers, um Schwachstellen zu identifizieren. Im Vergleich zu anderen Methoden sind solche Tests vor allem durch eine hohe Effektivität und Nachvollziehbarkeit der Ergebnisse gekennzeichnet. Ein Problem besteht allerdings darin, dass häufig der Business-Kontext der getesteten Anwendung nur unzureichend verstanden wird, da es sich meist um externe Tester handelt. Deshalb liegt der Fokus von Pentests oftmals auf der technischen Seite. Unterschieden wird also hauptsächlich zwischen Black- und White-Box. Idealerweise sollten nämlich möglichst viele Informationen über die Anwendung vorliegen. [24]

Eine Web Application Firewall schützt die Anwendungsebene. In der Regel ist sie sowohl benutzer- als auch sitzungs- und anwendungsorientiert, kennt die dahinter liegende Webanwendung und weiß, welche Dienste angeboten werden. Sie fungiert also als Vermittler zwischen einem Nutzer und der Anwendung selbst und stellt sicher, dass nur autorisierte

Aktionen ausgeführt werden können. Dies geschieht durch das Filtern, Überwachen und Blockieren jeglichen HTTP/S-Datenverkehrs. Welcher dabei als böswillig und welcher als sicher gilt, wird über eine Reihe von Richtlinien bestimmt, welche an individuelle Anforderungen angepasst werden können. Obwohl sie regelmäßig aktualisiert werden müssen, ermöglicht maschinelles Lernen bereits einen Teil davon automatisch. [61]

Im Bereich der Anwendungssicherheit kommt der TLS-Konfiguration eine große Bedeutung zu. Vor allem für den Produktivbetrieb ist es wichtig, dort gesetzte Einstellungen weitgehend automatisiert testen zu können. Neben den angebotenen Cipher Suites betrifft das auch verschiedene Aspekte der verwendeten Zertifikate, insbesondere aber deren Gültigkeit. Für diesen Zweck existieren frei verfügbare Tools wie SSL Labs – ein webbasierter Dienst, dessen Bewertung den De-Facto-Standard darstellt. Entsprechend der Sicherheit wird eine Webseite auf Basis des US-Schulnotensystems in A bis F bewertet. Dies funktioniert jedoch natürlich ausschließlich für extern erreichbare Systeme. Eine Automatisierung wird aber über das entsprechend frei verfügbare Kommandozeilentool geboten. [24]

4.2 Anforderungen an das Session Management

"Die Session-ID einer authentifizierten Session entspricht einem temporären Benutzerpasswort und erfordert daher auch denselben Schutz wie dieses." [24] Demnach gilt es wirksame Maßnahmen zu ergreifen, deren Beachtung eine Art Mindestschutz darstellt und die Gefahr eines Angriffs aber bereits deutlich verringert. [62]

Ein Ansatz bezieht sich auf die Session-ID selbst beziehungsweise deren Zufälligkeit – ist letztere nicht sichergestellt, kann erstere schlichtweg erraten oder durch eine Brute-Force-Attacke geknackt werden. Das Session Management wird standardmäßig zwar über den Applikationsserver abgewickelt, weswegen von eigenen Implementierungen generell abgesehen werden sollte, es unterscheidet sich allerdings in der Bildung der Session-IDs – diese sind in der Regel technologiespezifisch und teilen sich lediglich die gemeinsame Mindestlänge: 120 Bit. Dennoch lässt sich die Zufälligkeit zumindest rudimentär mithilfe von Tools wie WebScarab oder Burp testen. [24]

Werden Session-IDs mit Hilfe von Cookies übermittelt, sollte der Aktionsradius auf ein Minimum beschränkt werden. Da es sich hierbei um nichts anderes als eine Informationsübertragung über HTTP-Header handelt, gibt es die Möglichkeit verschiedene Flags zu setzen (Tabelle 9) [24]. Um eine optimale Sicherheit gewährleisten zu können, ist unter anderem der Pfad lediglich auf die Webanwendung selbst zu beschränken – damit dies möglich ist, müssen alle beteiligten Ressourcen unterhalb dieses angegebenen Pfads platziert werden. Cookies, welche an jeden Host in einer Domain geschickt werden, sollten also grundsätzlich vermieden werden. Macht die Webanwendung außerdem Gebrauch von einer TLS-Verschlüsselung, wird zusätzlich das secure-Flag gesetzt. Damit kann gewährleistet werden, dass Cookies niemals über eine unverschlüsselte Verbindung übertragen werden. Zuletzt ist das httpOnly-Flag zu setzen – jedoch nur unter der Voraussetzung, dass der verwendete

Browser diese Funktionalität unterstützt. Ansonsten verhält sich der Cookie wie im Normalzustand – kann also nach wie vor von JavaScript ausgelesen werden. Im schlimmsten Fall führt die Angabe des Flags allerdings zu einem Syntaxfehler und damit auch zur Ablehnung des Cookies. [62]

Tabelle 9: Auswirkung von gesetzten Flags auf das Session-Cookie

Quelle: in Anlehnung an [24]

domain	Ausweitung der Gültigkeit des Cookies vom Host- auf den Domain-Bereich – der Host, der das Cookie setzt, muss sich in der angegebenen Domain befinden, sonst wird die Einstellung vom Browser ignoriert	nicht setzen
expires	Erzeugung eines persistenten Cookies, das für den angegebenen Zeitraum auf der Festplatte des Benutzers gespeichert wird und dort auch nach Schließen des Browsers erhalten bleibt	nicht setzen
path	Einschränken der Gültigkeit des Cookies auf einen bestimmten Pfad des betreffenden Hosts	auf Pfad der Anwendung setzen
secure	Übertragung nur mittels HTTPS	setzen und HTTPS verwenden
httpOnly	kein clientseitiges Auslesen des Cookies mittels JavaScript möglich	setzen

Wird URL-Rewriting zur Übertragung der Session-ID eingesetzt – diese wird dabei automatisch in jeden Link der Webanwendung eingebaut – sollte der Nutzer in jedem Fall darüber in Kenntnis gesetzt werden. Versendet er eine gespeicherte Seite oder einen Ausschnitt daraus, führt das unweigerlich zu einer Steigerung des Angriffsrisikos. Um einem Über-die-Schulter-Schauen entgegenzuwirken, sollten entweder lange Session-IDs oder URLs verwendet werden – dies erschwert das einfache Abschreiben respektive ermöglicht es die ID aus dem sichtbaren Bereich des Bildschirms hinauszuschieben. Zusätzlich ist serverseitig sicherzustellen, dass lediglich interne Links mit einer Sitzungs-ID versehen werden. Soll zu externen Webseiten verlinkt werden, darf das niemals direkt, sondern immer über einen Redirect geschehen. Ansonsten ist die URL inklusive ID im Referrer einsehbar. Dies ist allerdings nicht immer vermeidbar. Benutzereingaben, die ebenso HTML-Tags enthalten und damit zur Auslieferung des Referrers seitens des Betreibers einer Webseite führen können, müssen demnach in gleichem Maße behandelt und gefiltert werden. [62] Dieses Verfahren sollte demzufolge als inhärent unsicher betrachtet und deaktiviert werden – vor allem wenn es nur als Fallback für Cookies dient. [62]

Lässt sich der Einsatz von URL-Rewriting trotz allem nicht vermeiden, ist er unbedingt in Verbindung mit Session-Binding-Techniken zu erfolgen. Beispielsweise kann dazu die IP-Adresse des Clients herangezogen werden. Diese wird bei jedem erneuten Zugriff mit jener,

bei der Instanziierung der Session hinterlegten, Adresse abgeglichen – stimmen die beiden nicht überein, wird die Sitzung invalidiert. In diesem Fall existieren allerdings deutliche Beschränkungen in der Wahl möglicher Einsatzumgebungen. Wird entweder eine Proxy-Architektur eingesetzt – der ausgehende Router und damit auch die IP wechselt während einer Session – oder die tatsächliche Adresse kommt aufgrund vorgeschalteter Netzkomponenten gar nicht erst bei der Webanwendung an, so stellt dies ein Hindernis dar. In Intranet-Bereichen hingegen gibt es in der Regel keine Probleme. Ein weiteres Merkmal für die Identifikation der Sitzung könnte die Typenbezeichnung des Browsers sein, welche im User-Agent-Header mitgeschickt wird. Diese Maßnahme stellt zwar keine erhebliche Verbesserung in Bezug auf den Schutz dar, ist aber aufgrund seiner leichten Durchführbarkeit dennoch zu empfehlen. Ähnlich verhält es sich mit dem Accept-, Accept-Language und Referrer-Header – sofern die Übermittlung von letzterem nicht im Browser deaktiviert oder durch Content-Filter blockiert wurde, würde bei jedem Request geprüft werden, ob die angegebene URL mit einem für alle möglichen Zugriffe festgelegten Teil des Pfads beginnt. [62] Kombiniert man all diese Header zusätzlich mit Eigenschaften des Browsers, welche sich ebenfalls mit JavaScript auslesen lassen, so lässt sich ein nahezu eindeutiger und stabiler Fingerabdruck generieren. [24]

Session-Binding-Techniken helfen allerdings nur so lange, bis einer bereits authentifizierten Session eine Anfrage untergeschoben werden soll – diese zielt dann darauf ab eine Änderung am Nutzerprofil oder der Sitzung selbst herbeizuführen. Da HTTP-GET deutlich anfälliger für CSRF-Angriffe ist, sollte ein Update per se mithilfe von POST durchgeführt werden – die Schwachstelle an sich ist damit jedoch nicht beseitigt. [24] So ist allerdings eine Kombination von URL-Rewriting und dem Einsatz von Cookies denkbar. Hierbei existieren grundsätzlich zwei Varianten. Eine Möglichkeit ist die Session-ID selbst in letzterem und einen Session-Code – also eine andere, nicht erratbare ID – über einen zusätzlichen Parameter, welcher in die URL eingefügt wird, zu transportieren. Beim Klick auf den Link wird die Übereinstimmung des in der Sitzung hinterlegten Session-Code mit dem übergebenen geprüft. Im Falle einer Inkongruenz, wird die Ausführung gestoppt und die ID invalidiert. Die zweite Option ist die des Dialogtrackings – gleichzeitig stellt sie auch die sicherste Form des Session Managements dar. Dabei ist der eingeführte Session-Code bei jedem Zugriff neu zu vergeben und somit nicht mehr statisch – er wird dann als Token bezeichnet. Dieser wird in jeden möglichen Dialogschritt eingebaut, welcher damit eindeutig identifiziert werden kann. Erhält die Webanwendung einen Request, wird geprüft, ob dieser gültig ist, indem das entsprechende Token mit einer Liste valider Tokens verglichen und bei dessen Vorkommen daraus gestrichen wird. Ein erneuter Aufruf ist demnach nicht mehr möglich – auch weil eine Aktion ganz ohne Token ebenfalls nicht ausgeführt werden kann. Es reicht also nicht mehr aus die Session-ID und ein, sich auf dem Übertragungsweg befindliches, Token zu kennen – letzteres muss aus der Liste gültiger Codes und damit von einem noch nicht aufgerufenen Link stammen. [62]

Ein weiterer Ansatz ist das grundlegende Erneuern der Session-ID – hierbei sind diverse Szenarien vorstellbar, in denen dies erforderlich ist (Abbildung 10). Einem Nutzer ist es

beispielsweise gewöhnlich möglich, sich parallel mehrfach an einer Webanwendung anzumel-den – für ihn werden dann verschiedene authentifizierte Sitzungen (Concurrent Sessions) verwaltet. Diese Funktion sollte jedoch unterbunden werden, da sowohl Probleme durch Nebenläufigkeit (Race Conditions) auftreten können als auch die Angriffsfläche nur unnötig vergrößert wird. Folglich sollte immer geprüft werden, ob für den sich anmeldenden Nutzer bereits eine Session besteht – falls ja, wird deren ID invalidiert und durch eine neue ersetzt. [24] Außerdem ist jede Session aktiv zu beenden. Dies kann sowohl durch den Nutzer selbst – in Form eines Logout-Buttons – als auch durch die Webanwendung geschehen. Für letztere wiederum existieren verschiedene Möglichkeiten, die aber in jedem Fall greifen müssen, sollte das gewünschte Nutzerverhalten ausbleiben. Entsteht beispielsweise eine Fehlersituation, wird die Abmeldung in dem Moment des Eintritts erzwungen. [62] Lässt der Nutzer eine Webanwendung geöffnet, bleibt diese nach dem letzten Request noch für eine gewisse Zeitspanne (Idletime) aktiv und wird erst dann beendet. Allerdings ist es möglich, den User darüber in Kenntnis zu setzen – dieser kann dann die eventuell unfreiwillige Abmeldung verhindern. Zusätzlich ist es ratsam serverseitig eine maximale Lebensdauer (Timeout) zu implementieren – in diesem Fall wird die Session beendet, egal ob der Nutzer aktiv ist oder nicht. So kann es diesem unmöglich gemacht werden eine Sitzung beliebig lange offenzuhalten, selbst wenn ihm beispielsweise bereits die Zugriffsrechte entzogen wurden. Jede Abmeldung verlangt die Invalidation der Session-ID – nach abermalig erfolgreicher Authentifizierung wird diese neu gesetzt. [24]

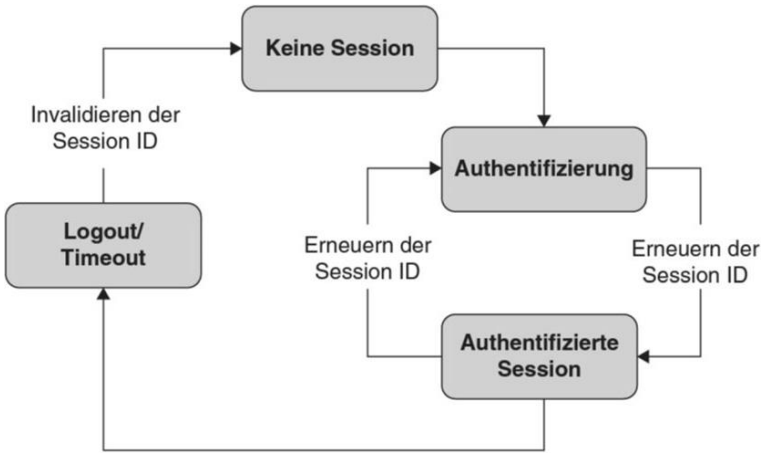

Abbildung 10: Session Lifecycle

Quelle: [24]

Es ist für Nutzer gegebenenfalls auch möglich, persistente Sessions zu erzeugen – durch eine entsprechende Auswahlmöglichkeit innerhalb des Anmeldedialogs wird das expires-Flag gesetzt und dem Browser damit mitgeteilt, dass die Session-ID vorgehalten werden soll. Wird ihnen dadurch der Zugriff auf sensible Daten gewährt, sollte dies allerdings unterbunden werden. Dennoch ist es legitim eine Art "Remember-Me"-Option – durch einen zusätzlichen Cookie wird der Nutzer selbst erkannt und muss nur noch sein Passwort eingeben – oder Persistenz auf niedrigeren Ebenen zu implementieren: über anonyme, identifizierte oder einfach authentifizierte Sessions. [24]

4.3 HTTP/3

Durch die Einführung eines neuen Standards wandelt sich die Datenübertragung im Web grundlegend. HTTP/3, festgehalten im RFC 9114, setzt unter Verwendung von QUIC – ursprünglich ein Akronym für Quick UDP Internet Connections – nicht mehr auf das Transportprotokoll TCP, sondern wie der Name schon sagt auf UDP – das User Datagram Protocol. [63] Mit einer auf allen Ebenen entsprechenden Implementierung und diesbezüglichen Unterstützung wurde zwar bereits begonnen – da viele Webseiten aber noch nicht einmal mit HTTP/2 arbeiten, wird eine komplette Umstellung auch nicht in naher Zukunft erfolgen. [64]

UDP, spezifiziert in RFC 768, unterscheidet sich insofern von TCP, dass es sich um ein verbindungsloses Protokoll handelt – eine Datenübertragung kann also sofort und ohne vorherigen Verbindungsaufbau beginnen. Dadurch fehlt jedoch ein Quittierungsmechanismus und es kann nicht gewährleistet werden, dass keines der Pakete verloren geht, dupliziert wurde oder außerhalb der Reihenfolge eintrifft. Mit einer Prüfsumme lässt sich allerdings immerhin feststellen, ob die Pakete fehlerfrei ankommen. Ist dies der Fall, erfolgt eine direkte Weiterleitung zur Anwendung selbst – die Adressierung erfolgt auch hierbei über Portnummern. [65]

QUIC selbst wurde bereits im Jahr 2013 erstmals vorgestellt, sollte dann allerdings als Grundlage für HTTP/3 und dessen Standardisierung dienen. Die Abhängigkeit von TCP, welche sich auch durch zahlreiche Weiterentwicklungen nicht auflösen lassen konnte, stellte die hauptsächliche Motivation dafür dar. Die anfängliche Schwäche einer durch den Handshake verzögerten Datenübertragung entwickelte sich bei steigender Komplexität der Webseiten zu einer großen Herausforderung. Das sogenannte Head-of-Line-Blocking – geht ein Paket verloren, müssen alle restlichen Pakete einer Warteschlange darauf warten, dass dieses erneut übertragen wird – lässt die Latenz weiter ansteigen. QUIC schafft mit einer Vorwärtsfehlerkorrektur Abhilfe. Damit lassen sich einzelne fehlerhafte Pakete rekonstruieren – und erst, wenn mehrere betroffen sind, werden Retransmissions notwendig. Lastspitzen können durch Packet Pacing – also die automatische Begrenzung der Datenübertragungsrate – vermieden werden. Durch Multiplex-Verbindungen ist der Client außerdem in der Lage mehrere Datenströme gleichzeitig und unabhängig voneinander zu senden und auch zu empfangen. Eine Verbindung wird nämlich nicht über die Kombination aus IP-Adresse und Portnummer adressiert, sondern über eigene Kennungen. Daraus ergibt sich zusätzlich der Vorteil, dass beim Roaming – dem Wechsel zwischen verschiedenen Netzwerken – auch keine Verbindungen mehr unterbrochen werden. QUIC setzt zudem auf eine bereits integrierte Verschlüsselung mit der geschwindigkeitsoptimierten TLS-Version 1.3. Da deren Parameter bereits in den ersten UDP-Paketen übermittelt werden und darüber hinaus sowohl Header als auch Payload verschlüsselt sind, lassen sich auch Man-in-the-Middle-Attacken und die Manipulation der Daten zuverlässiger verhindern. [66]

4.4 Incident-Response-Plan

Präventive IT-Forensik oder Forensic Readiness beschreibt die optimale Vorbereitung auf den Ernstfall. Denn dieser lässt sich auch trotz hohen Sicherheitsstandards nicht immer vermeiden. Dafür muss die gesamte Unternehmens- und IT-Landschaft einer IST-Analyse unterzogen werden – letztere ist vor allem dahingehend zu überprüfen, ob nach einem Sicherheitsvorfall auch genügend Spuren gesichert werden können, die für eine forensische Auswertung herangezogen werden können. Ein im Anschluss zu erstellender Incident-Response-Plan stellt dann eine Art Schritt-für-Schritt-Anleitung dar – das Vorgehen soll eine schnelle und sichere Wiederinbetriebnahme der Systeme gewährleisten und damit Ausfallzeiten beziehungsweise wirtschaftliche Einbußen minimieren. [67]

Grundsätzlich durchläuft ein solcher Incident-Response-Lebenszyklus sechs Phasen. Eine intensive Vorbereitung bildet dabei den Grundstein für den restlichen Prozess. Kommt es dann tatsächlich zu einem Ernstfall, muss dieser identifiziert werden – es müssen also Kennzeichen und Symptome festgelegt werden, die einen potentiellen Cyberangriff und den Zeitpunkt eines Eingreifens des IT-Teams definieren. Bei der Eindämmung kommt es darauf an, alle Bedrohungen zunächst nur an der weiteren Ausbreitung zu hindern – betroffene Dateien und Systeme sollten im Hinblick auf eine spätere forensische Untersuchung also nicht gelöscht oder bereinigt werden. Daraufhin gilt es die Ursache ausfindig zu machen. Ist dies geschehen, muss eine vollständige Beseitigung aller ausfindig gemachten Sicherheitslücken erfolgen. Andernfalls bedeutet dies eine Wiederholung des Vorfalls. Sobald die Sicherheit vollends wiederhergestellt wurde, können diese Systeme in ihre Geschäftsumgebung rückgeführt werden. Zuletzt sollten die Erkenntnisse, die aus dem Vorfall gewonnen werden konnten, analysiert werden – mit dem Ziel, noch eventuell bestehende Mängel des Incident-Response-Plans zu beseitigen und entsprechende Optimierungen durchzuführen. Es handelt sich also um einen sich stetig weiterentwickelnden Prozess. [68]

Zuerst sollten also die wichtigsten Stakeholder bestimmt werden – eine ordnungsgemäße Vorbereitung erfordert schließlich nicht nur das Sicherheitsteam. Stattdessen wirkt sich ein solcher Vorfall auf jegliche Abteilungen eines Unternehmens aus. Für eine effektive Koordination müssen also Vertreter der Geschäftsleitung sowie Experten aus den Gebieten für IT, Recht und Public-Relations hinzugezogen werden. Zusätzlich sollte eine entsprechende Kommunikationsmethode eingerichtet werden – vor allem für den Fall, dass die für gewöhnlich genutzten Kanäle von dem Angriff betroffen sind. [69]

Um das Ausmaß und die Folgen eines potentiellen Sicherheitsvorfalls bestimmen und die optimale Schutzstrategie erarbeiten zu können, müssen unternehmenskritische Ressourcen bestimmt werden. Diese sollten im Ernstfall oberste Priorität haben, um Unterbrechungen des normalen Geschäftsbetriebs auf ein Minimum zu reduzieren. [69]

Zusätzliche Theorieübungen – oft im Rahmen von RedTeam-Übungen – sorgen dafür, dass ein Unternehmen koordinierter und effektiver reagieren kann. Bestehende Reaktionsmaßnahmen sollten auf alle möglichen Incident-Response-Szenarien durchgespielt werden.

Dies betrifft auch die Stakeholder, welche über das unmittelbar technische Team hinausgehen. Im Voraus sollte also festgelegt werden, wer bei einem Angriff informiert werden muss. Wird ein aktiver Angreifer in einem Netzwerk erkannt, ist entscheidend, dass das Sicherheitsteam ermittelt, wie dieser die Umgebung überhaupt infiltrieren konnte, welche Tools oder Techniken dabei verwendet und welche Ressourcen anvisiert wurden. Solche Informationen helfen dabei, die richtige Vorgehensweise zu bestimmen und den Angriff schnellstmöglich zu neutralisieren. Der Angreifer kann zur Sammlung wichtiger Informationen über dessen Ziele und Methoden zunächst auch nur beobachtet werden. Sollte dabei eine Datenpanne festgestellt werden können, muss ermittelt werden können, was und wie es exfiltriert wurde. Wurde ein System mit hoher Priorität kompromittiert, kann der Geschäftsbetrieb nicht in jedem Fall wie gewohnt aufrechterhalten werden. Die Erstellung eines Business-Recovery-Plans sollte also ebenfalls in Betracht gezogen werden – damit Unterbrechungen im Ernstfall auf ein Minimum reduziert werden können. [69]

Damit es gar nicht erst zu einem Sicherheitsvorfall kommt, sollten bereits im Vorfeld Schutzmaßnahmen getroffen werden. Security-Tools für Netzwerk, Server, Cloud, Mobilgeräte und die Anwendungen selbst müssen also bereits während des normalen Geschäftsbetriebs bereitgestellt und implementiert werden. [69]

Da auch schwache Zugriffskontrollen von Angreifern ausgenutzt werden, um Berechtigungen ausweiten und so die Abwehr eines Unternehmens unterwandern zu können, sollte der Zugang reguliert werden. Als wirksame Maßnahmen haben sich hierbei die mehrstufige Authentifizierung, die Reduzierung der Anzahl der zu überwachenden Zugriffspunkte und der Konten mit Administrator-Rechten auf ein Minimum – also das Principle of Least Privilege – und ein robustes Session Management erwiesen. [69]

Und selbst wenn Angriffe nicht nur dadurch verhindert werden können, tragen Schulungsprogramme dennoch dazu bei, deren Risiko zu verringern und vor allem die Anzahl der Meldungen zu begrenzen, auf die das Sicherheitsteam reagieren muss. Mithilfe einer Angriffssimulation kann ohne Sicherheitsrisiko ein realer Vorfall gestartet werden. Diejenigen Mitarbeiter, welche sich davon täuschen lassen, müssen ein spezielles Awareness-Training absolvieren. Damit lassen sich also gezielt Benutzergruppen identifizieren, die ebenfalls einen Schwachpunkt darstellen können. [69]

Eine angemessene Reaktion auf einen tatsächlichen Angriff erfordert Transparenz über jegliche Vorgänge innerhalb einer IT-Landschaft. Das Sicherheitsteam muss also sicherstellen, dass ihnen das nötige Handwerkszeug zur Verfügung steht, auf welches sie einerseits für die Ermittlung von Eintritts- und Persistenzpunkten der Angreifer und andererseits für die Bestimmung des Ausmaßes und der Folgen eines Vorfalls angewiesen sind. Da viele Angriffe allerdings erst nach Tagen oder Wochen entdeckt werden, müssen zusätzlich Protokolldaten – der Schwerpunkt liegt hier auf Endpoint- und Netzwerkdaten – gesammelt und ebenfalls über mehrere Wochen oder gegebenenfalls Monate gespeichert werden. Bedarf es einer Vorfallsanalyse, kann dann darauf zurückgegriffen werden. [69]

Neben der Sicherstellung dieser Transparenz sollte auch in Tools investiert werden, welche während der nachfolgenden Untersuchung den nötigen Kontext liefern. Endpoint beziehungsweise Extended Detection and Response sind dabei die am häufigsten verwendeten – damit kann in der gesamten Umgebung nach Indicators of Compromise respektive Attack gesucht werden. Dadurch können Ressourcen ermittelt werden, welche kompromittiert wurden. Infolgedessen lassen sich Ausmaß und Folgen eines Angriffs bemessen. Je mehr Daten also erhoben werden, desto mehr Kontext steht grundsätzlich auch für die Analyse zur Verfügung. Die Fragen, welche Ressourcen die Angreifer im Visier hatten, wie sie sich Zugang zur Umgebung verschafft haben und ob die Möglichkeit besteht, dass sie erneut auf die Umgebung zugreifen, sollten mit diesem Kontext beantwortet werden können.

Einen Angriff zu erkennen, ist allerdings nur ein Teil des Prozesses. Um angemessen darauf reagieren zu können, muss das Sicherheitsteam auch in der Lage sein, diverse Reaktionsmaßnahmen zum Stoppen und Beseitigen solcher Attacken einzuleiten. Dazu zählen unter anderem das Isolieren betroffener Hosts, die Blockade von Command and Control beziehungsweise schädlichen Aktivitäten in betroffenen Webanwendungen, das Einfrieren kompromittierter Konten und damit auch die Zugriffssperrung für den Angreifer, das Beseitigen von dessen Artefakten und Werkzeugen, das Schließen von Eintrittspunkten respektive Persistenzbereichen und letztendlich die Wiederherstellung betroffener Ressourcen und Anpassung existierender Konfigurationen. [69]

Auch wenn ein solcher Vorfall eine schnelle und effektive Reaktion erfordert, sind viele Unternehmen nicht in der Lage, angemessen darauf zu reagieren – zumindest nicht ohne fremde Hilfe. Um also gewährleisten zu können, dass trotzdem die richtigen Maßnahmen ergriffen werden, sollte gegebenenfalls ein externer Dienstleister hinzugezogen werden. Solche Managed-Detection-and-Response-Provider bieten zu jeder Zeit Analysen von Sicherheitsvorfällen an. Sie helfen also nicht nur, darauf zu reagieren, bevor sie sich zu weitreichenden Sicherheitspannen entwickeln, sie senken auch deren Wahrscheinlichkeit. Gelegentlich werden Data-Forensic-Incident-Response-Services auch nach einem Vorfall noch weiter genutzt – zum Sammeln von Beweisen und anschließendem Geltendmachen eines Rechts- oder Versicherungsanspruchs. [69]

5 Fazit

In diesem Kapitel wird eine mögliche Gestaltung eines Leitfadens mit den Best Practices der Entwicklung eines sicheren Session Managements vorgestellt. Es folgt eine kritische Betrachtung des beleuchteten Themas und ein Ausblick zu möglichen Entwicklungen.

5.1 Ergebnisse

In jedem Fall sollte eine Webanwendung ein angemessenes Sicherheitsniveau bereits durch Maßnahmen den Basisschutz betreffend erreichen. Durch automatisches Schwachstellenmanagement und regelmäßiges Penetration Testing sollten die kritischsten Sicherheitslücken behoben werden können. Eine Web Application Firewall filtert dann zusätzlich den Datenverkehr während des normalen Geschäftsbetriebs.

Auch wenn die Generierung der Session-ID einer ausgereiften API überlassen werden sollte, ist es von Vorteil, grundlegende Anforderungen an sie zu kennen. Aus einer hohen Zufälligkeit (Randomness) – das Muster, das für die Erzeugung verwendet wird, darf nicht aus einer Menge von Proben ableitbar sein – kann gefolgert werden, dass die ID auch nicht mit extern bekannten oder erratbaren Daten verknüpft werden darf. Außerdem muss sie ausreichend lang sein, um gegenüber Brute-Force-Angriffen zu bestehen.

Für die Übertragung der ID sollte immer eine geschützte Verbindung gewählt werden, sie darf also in keinem Fall unverschlüsselt übertragen werden. Daher empfiehlt sich als Transportmedium ein Cookie – hierbei sollten sowohl das secure- als auch das httpOnly-Flag gesetzt sein. Der angegebene Pfad sollte lediglich auf den der Anwendung beschränkt sein.

Die Session selbst sollte außerdem das kürzestmögliche Ablaufdatum haben – gerade noch angemessen für den jeweiligen Anwendungszweck. Wenn die Sitzung nicht manuell vom Nutzer beendet wird, müssen also Mechanismen greifen, die sie automatisch terminieren. Sobald dies geschieht, ist die Session-ID zu invalidieren und nach jeder Authentifizierung auch wieder neu zu vergeben.

Das Session Management sollte also nicht isoliert, sondern immer in Zusammenhang mit der verwendeten Verschlüsselung, betrachtet werden (Tabelle 10). Eine Session-ID kann schließlich noch so zufällig sein, wenn sie schlussendlich einfach ausgelesen wird und dem Angreifer so die Übernahme einer bereits authentifizierten Sitzung ermöglicht. [24]

Tabelle 10: Checkliste funktionaler Sicherheitsanforderungen

Quelle: in Anlehnung an [24]

HTTPS-Verschlüsselung	▪ der Zugriff auf sensible Bereiche ist ausschließlich über HTTPS möglich, ansonsten wird sofort auf den HTTPS-Kontext weitergeleitet ▪ ausschließlich sichere TLS-Protokolle und Cipher Suites mit hoher Sicherheit werden vom Server unterstützt
Sicherheitszertifikate	▪ verwendete Zertifikate besitzen ausreichende Gültigkeit ▪ ausschließlich robuste Algorithmen kommen zum Einsatz
Session Management	▪ die Session-ID wird nach jeder Authentifizierung erneuert ▪ korrekte Cookie-Flags werden gesetzt ▪ das Session Management lässt sich nur mittels Cookies durchführen (Fallback auf URL-Rewriting wird verhindert) ▪ das Session-Timeout wird korrekt durchgesetzt ▪ ändernde HTTP-Anfragen führen zu einem 403-Fehler, wenn kein gültiges CSRF-Token mitgesendet wurde
Datensicherheit	▪ sensible Daten lassen sich ausschließlich über HTTPS und mittels POST übertragen ▪ No-Caching-Header sind bei der Übermittlung von sensiblen Daten gesetzt
Security Header	▪ alle spezifizierten HTTP-Security-Header werden von der Anwendung versendet

Kommt es doch zu einem Sicherheitsvorfall, sollte ein Incident-Response-Plan bereits zur Verfügung stehen. Dadurch sollte immerhin verhindert werden können, dass der Geschäftsbetrieb eine Zeit lang eingestellt werden muss oder sich der Angriff sogar zu einer weitreichenden Datenpanne entwickelt.

5.2 Bewertung der Arbeit

OWASP führt das Session Management zusammen mit der Authentifizierung unter dem Punkt Broken Access Control. In diesem Fall wäre es also sinnvoll bei der Ausarbeitung eines Leitfadens auch beides gemeinsam zu betrachten.

In jedem Fall entwickeln sich die Methoden der Angreifer laufend weiter. Konzeption und Implementierung solcher Sicherheitsmaßnahmen können dieser Entwicklung demnach nur mit einer entsprechenden Verzögerung folgen – reagieren statt agieren. Ein jederzeit angemessenes Sicherheitsniveau kann also nur dadurch aufrecht erhalten werden, dass die Risikoabschätzung zyklisch wiederholt und Maßnahmen gegebenenfalls angepasst werden.

IT-Sicherheit stellt demnach auch keinen statischen Zustand dar, sondern muss als ein andauernder Prozess verstanden werden. Einen konkreten Leitfaden zu entwickeln, ist bei der Geschwindigkeit der Entwicklungen – vor allem im Bereich der Verschlüsselung – sogar fast unmöglich. Lediglich Best Practices können den Entwicklern mit an die Hand gegeben werden – und auch nur dann, wenn sie regelmäßig überarbeitet und aktualisiert werden.

5.3 Ausblick

Im Bereich des Session Managements gibt es eine Entwicklung vom Einsatz der Cookies hin zu Tokens. Statt die Informationen also sowohl auf dem Client als auch auf dem Server zu speichern, wird das bei einer Authentifizierung generierte Token nur bei ersterem hinterlegt und letzterer validiert es bei jeder neuen Anfrage. Das wiederum schafft dann die Möglichkeit, dass diese auch in unterschiedlichen Stufen eines Authentisierungsprozesses genutzt werden können. Werden zusätzliche Hilfsmittel wie Dongle, welche direkt im Gerät des Nutzers stecken, verwendet, wird das Token auf Basis dieses Private Key generiert. Der User ist also schon allein dadurch authentifiziert, dass er ein Hard Token besitzt. Soft Tokens hingegen werden auf Basis des genutzten Geräts generiert – handelt es sich dabei um ein mobiles, kann eine Authentisierung auch in Verbindung mit dem Fingerabdruck stattfinden. [70]

Bezüglich der Verwendung von QUIC und HTTP/3 hegen Forscher noch Zweifel. Abgesehen von der Performance einzelner Implementierungen, welche sich im Vergleich zu HTTP/2 nicht zwingend verbessert – dies liegt auch an einer wesentlich höheren CPU-Last – werden Datenschutzprobleme im Zusammenhang mit Fingerprinting bemängelt. Beides liegt aber eher an der Art der Implementierung von QUIC und nicht unbedingt an der Protokollspezifikation selbst. [63]

Literatur

[01] Simic, Ivona: OWASP Top 10 2021 – Der ultimative Leitfaden, https://crashtest-security.com/de/owasp-top-10-2021, verfügbar am .2022

[02] OWASP Top 10 team: OWASP Top 10:2021, https://owasp.org/Top10, verfügbar am .2022

[03] Kuketz, Mike: Angriffstechniken – Sicherheit von Webanwendungen Teil 3, https://www.kuketz-blog.de/angriffstechniken-sicherheit-von-webanwendungen-teil3, verfügbar am .2022

[04] Suchhelden GmbH: Was genau verbirgt sich eigentlich hinter dem World Wide Web?, https://www.suchhelden.de/lexikon/world-wide-web.php, verfügbar am .2022

[05] gebhard-webdesign: Internetprotokolle, https://www.webtechnologien.com/wissen/das-internet/internetprotokolle, verfügbar am .2022

[06] Meinel, Christoph: Wie die technischen Normen des Internets entstehen, https://www.spektrum.de/kolumne/wie-die-technischen-normen-des-internets-entstehen/1848853, verfügbar am .2022

[07] KnowHow: TCP (Transmission Control Protocol) – das Transportprotokoll im Porträt, https://www.ionos.de/digitalguide/server/knowhow/tcp-vorgestellt, verfügbar am .2022

[08] Meinel, Christoph: Wie funktioniert ein Computernetzwerk?, https://www.spektrum.de/kolumne/wie-funktioniert-ein-computernetzwerk/1638398, verfügbar am .2022

[09] Lutkevich, Ben: TCP (Transmission Control Protocol), https://www.computerweekly.com/de/definition/TCP-Transmission-Control-Protocol, verfügbar am .2022

[10] Luber, Stefan; Donner, Andreas: Was ist HTTP (Hypertext Transfer Protocol)?, https://www.ip-insider.de/was-ist-http-hypertext-transfer-protocol-a-691181, verfügbar am .2022

[11] Hosting-Technik: Was ist HTTP?, https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-http, verfügbar am .2022

[12] Kaushik, Gunjan; Singh, Ravinder: What is HTTP Request?, https://www.toolsqa.com/client-server/http-request, verfügbar am .2022

[13] Häßler, Ulrike: Der HTTP-Request: Daten vom Server holen, https://www.mediaevent.de/tutorial/http-request.html, verfügbar am .2022

[14] Sematext Group: HTTP Requests, https://sematext.com/glossary/http-requests, verfügbar am .2022

[15] Gitonga, Davis: How the Web Works, HTTP Request/Response Cycle, https://davisgitonga.dev/blog/request-response-cycle, verfügbar am .2022

[16] IBM Corporation: HTTP responses, https://www.ibm.com/docs/en/cics-ts/5.2?topic=protocol-http-responses, verfügbar am .2022

[17] MirkoK; Augsten, Stephan: Was bedeutet „stateless“?, https://www.dev-insider.de/was-bedeutet-stateless-a-974510, verfügbar am .2022

[18] SSL.com Support Team: What is HTTPS?, https://www.ssl.com/faqs/what-is-https, verfügbar am .2022

[19] Nestmann, Georg; Nöll, Hendrik; Woeschka, Alexander: TLS: Die Technologie für sichere Verschlüsselung und Kommunikation im Netz, https://www.comcrypto.de/wissensecke/einfach-erklaert/was-ist-tls-verschluesselung.html, verfügbar am .2022

[20] Sicherheit: TLS: Wie das Internet verschlüsselt wird, https://www.ionos.de/digitalguide/server/sicherheit/tls-transport-layer-security, verfügbar am .2022

[21] DigiCert, Inc: Was sind SSL, TLS und HTTPS?, https://www.websecurity.digicert.com/de/de/security-topics/what-is-ssl-tls-https, verfügbar am .2022

[22] Belsky, Vadim: Webanwendungen vs. klassische Websites: wo liegt der Unterschied?, https://www.scnsoft.de/blog/webanwendung-vs-website, verfügbar am .2022

[23] Adobe: Grundlegendes zu Webanwendungen, https://helpx.adobe.com/de/dreamweaver/using/web-applications.html, verfügbar am .2022

[24] Rohr, Matthias: Sicherheit von Webanwendungen in der Praxis – Wie sich Unternehmen schützen können – Hintergründe, Maßnahmen, Prüfverfahren und Prozesse, Wiesbaden, Springer Vieweg, 2018

[25] Suchmaschinenmarketing: SEO-Basics: Über interne Verlinkung zur richtigen Website-Struktur, https://www.ionos.de/digitalguide/online-marketing/suchmaschinenmarketing/seo-basics-interne-verlinkung-und-website-struktur/, verfügbar am .2022

[26] Rabolini, Jacopo: Graphite: a user flow kit for Sketch, https://github.com/KingFelix/Graphite, verfügbar am .2022

[27] Packetlabs: Session Management in HTTP: How does it work?, https://www.packetlabs.net/posts/session-management, verfügbar am .2022

[28] Gareis, Thomas: Session-ID, https://www.seobility.net/de/wiki/Session-ID, verfügbar am .2022

[29] Eilers, Carsten: You've been hacked! – Alles über Exploits gegen Webanwendungen, Bonn, Rheinwerk Verlag, 2019

[30] Marz, Fabian: WebWissen: Entwicklertools im Browser, https://hallo.digital/blog/webwissen-entwicklertools-im-browser, verfügbar am .2022

[31] Kurtz, Andreas: Von niedrig bis kritisch: Schwachstellenbewertung mit CVSS, https://www.heise.de/hintergrund/Von-niedrig-bis-kritisch-Schwachstellenbewertung-mit-CVSS-5031983.html, verfügbar am .2022

[32] Klein-Hennig, Martin: Sicherheitslücken FREAK und Logjam – was tun?, https://www.datenschutz-notizen.de/sicherheitsluecken-freak-und-logjam-was-tun-1211418, verfügbar am .2022

[33] Bundesamt für Sicherheit in der Informationstechnik: Man-In-The-Middle-Angriff, https://www.bsi.bund.de/SharedDocs/Glossareintraege/DE/M/Man-In-The-Middle-Angriff.html, verfügbar am .2022

[34] Heise: Angriffe auf TLS, https://www.heise.de/select/ix/2018/8/1533435196337811/ix.0818.110-116.qxp_table_3639.html, verfügbar am .2022

[35] Deveria, Alexis: Browser support tables for modern web technologies, https://caniuse.com, verfügbar am .2022

[36] Hosting-Technik: SSL-Stripping: Grundlagen und Schutzmöglichkeiten, https://www.ionos.de/digitalguide/hosting/hosting-technik/ssl-stripping-so-schuetzen-sie-ihr-webprojekt, verfügbar am .2022

[37] Bless, Roland et al.: Sichere Netzwerkkommunikation – Grundlagen, Protokolle und Architekturen, Berlin Heidelberg, Springer-Verlag, 2005

[38] Kiprin, Borislav: Wie man einen SSL-POODLE-Angriff verhindert, https://crashtest-security.com/de/poodle-angriff, verfügbar am .2022

[39] Böck, Hanno: Uraltalgorithmen gefährden Millionen Webseiten, https://www.golem.de/news/verschluesselung-neue-tls-angriffe-smack-und-freak-1503-112737.html, verfügbar am .2022

[40] Klein-Hennig, Martin: OWASP Top Ten: A2 – Fehler in Authentifizierung und Session Management, https://www.datenschutz-notizen.de/owasp-top-ten-a2-fehler-in-authentifizierung-und-session-management-2713238, verfügbar am .2022

[41] Kiprin, Borislav: Was ist die DROWN-Schwachstelle und wie kann sie verhindert werden?, https://crashtest-security.com/de/drown-angriff, verfügbar am .2022

[42] Kiprin, Borislav: Was ist ein SSL-BEAST-Angriff und wie funktioniert er?, https://crashtest-security.com/de/ssl-beast-angriff, verfügbar am .2022

[43] Kiprin, Borislav: Was ist ein CRIME-Angriff und wie funktioniert er?, https://crashtest-security.com/de/crime-angriff, verfügbar am .2022

[44] Kiprin, Borislav: Wie man einen BREACH-Angriff verhindert, https://crashtest-security.com/de/breach-angriff, verfügbar am .2022

[45] Kiprin, Borislav: Was ist ein ROBOT-Angriff und wie kann er verhindert werden?, https://crashtest-security.com/de/robot-angriff, verfügbar am .2022

[46] Kiprin, Borislav: Was ist der Heartbleed-Bug und wie kann er verhindert werden?, https://crashtest-security.com/de/heartbleed-bug, verfügbar am .2022

[47] Schirrmacher, Dennis: Alpaca-Attacke: Angreifer könnten mit TLS gesicherte Verbindungen attackieren,

https://www.heise.de/news/ALPACA-Attacke-Angreifer-koennten-mit-TLS-gesicherte-Verbindungen-attackieren-6066915.html, verfügbar am .2022

[48] TheFastCode: What is shattered? SHA1 collision attacks explained, https://www.thefastcode.com/de-eur/article/what-is-shattered-sha-1-collision-attacks-explained, verfügbar am .2022

[49] Böck, Hanno: Sloth-Angriffe nutzen alte Hash-Algorithmen aus, https://www.golem.de/news/md5-sha1-sloth-angriffe-nutzen-alte-hash-algorithmen-aus-1601-118381.html, verfügbar am .2022

[50] Kiprin, Borislav: Was ist der SWEET32-Angriff und wie kann man ihn verhindern?, https://crashtest-security.com/de/sweet32-angriff, verfügbar am .2022

[51] Luber, Stefan; Schmitz, Peter: Was ist eine Seitenkanalattacke?, https://www.security-insider.de/was-ist-eine-seitenkanalattacke-a-893560, verfügbar am .2022

[52] Kiprin, Borislav: Was ist ein SSL LUCKY13-Angriff und wie kann man ihn verhindern?, https://crashtest-security.com/de/lucky13-angriff, verfügbar am .2022

[53] Schmidt, Jürgen: HEIST: Wiederbelebter Angriff auf HTTPS vorgestellt, https://www.heise.de/security/meldung/HEIST-Wiederbelebter-Angriff-auf-HTTPS-vorgestellt-3287786.html, verfügbar am .2022

[54] National Institute of Standards and Technology (NIST): CVE-2020-1968 Detail, https://nvd.nist.gov/vuln/detail/CVE-2020-1968, verfügbar am .2022

[55] Böck, Hanno: Raccoon-Angriff auf TLS betrifft nur Wenige, https://www.golem.de/news/diffie-hellman-seitenkanal-raccoon-angriff-auf-tls-betrifft-nur-wenige-2009-150735.html, verfügbar am .2022

[56] Papazyan, Syuzanna: Was ist ein Session-Hijacking-Angriff?, https://powerdmarc.com/de/what-is-a-session-hijacking-attack, verfügbar am .2022

[57] National Institute of Standards and Technology (NIST): Session Hijacking, https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=Session+hijack&search_type=all&isCpeNameSearch=false, verfügbar am .2022

[58] National Institute of Standards and Technology (NIST): Cross-Site Request Forgery, https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=Cross-Site+Request+Forgery&queryType=phrase&search_type=all&isCpeNameSearch=false, verfügbar am .2022

[59] Andrews, Mike; Whittaker, James A.: How to Break Web Software – Functional and Security Testing of Web Applications and Web Services, Boston, Addison-Wesley Professional, 2006

[60] National Institute of Standards and Technology (NIST): Clickjacking, https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=Clickjacking&search_type=all&isCpeNameSearch=false, verfügbar am .2022

[61] F5, Inc.: Was ist eine Web Application Firewall (WAF)?, https://www.f5.com/de_de/services/resources/glossary/web-application-firewall, verfügbar am .2022

[62] Schreiber, Thomas; Hoffmann, Achim: Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/WebSec/WebSec.pdf?__blob=publicationFile&v=1, verfügbar am .2022

[63] Pfister, Benjamin: IETF erhebt HTTP/3 zum RFC, https://www.heise.de/news/IETF-erhebt-HTTP-3-zum-RFC-7135411.html, verfügbar am .2022

[64] Jankov, Tonino: Was ist HTTP/3? – Hintergrundinformationen über das schnelle neue UDP-basierte Protokoll, https://kinsta.com/de/blog/http3, verfügbar am .2022

[65] Luber, Stefan; Donner, Andreas: Was ist UDP (User Datagram Protocol)?, https://www.ip-insider.de/was-ist-udp-user-datagram-protocol-a-789006, verfügbar am .2022

[66] Luber, Stefan; Donner, Andreas: Was ist QUIC (Quick UDP Internet Connections)?, https://www.ip-insider.de/was-ist-quic-quick-udp-internet-connections-a-1031914, verfügbar am .2022

[67] Logemann, Thorsten: Präventive IT-Forensik, https://www.intersoft-consulting.de/it-forensik/praeventive-it-forensik, verfügbar am .2022

[68] Kahmen, Markus: Incident Management – Mit Incident Response Plan vor Cyberattacken schützen, https://www.business-

wissen.de/artikel/incident-management-mit-incident-response-plan-vor-cyberattacken-schuetzen, verfügbar am .2022

[69] Sophos Ltd.: INCIDENT RESPONSE GUIDE, https://assets.sophos.com/X24WTUEQ/at/6h78cmh5rs35shbkw354ch/sophos-incident-response-guide-de.pdf, verfügbar am .2022

[70] Pollicove, Matt: Ultimate Guide to Token-based Authentication, https://www.pingidentity.com/en/resources/blog/post/ultimate-guide-token-based-authentication.html, verfügbar am .2022

[71] Fruhlinger, Josh; Maier, Florian: Was ist SSL/TLS?, https://www.computerwoche.de/a/was-ist-ssl-tls,3553233, verfügbar am .2022

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Mittweida, 24.07.2022	Kristina Tanja Koska	
Ort, Datum	Vollständiger Name	Unterschrift