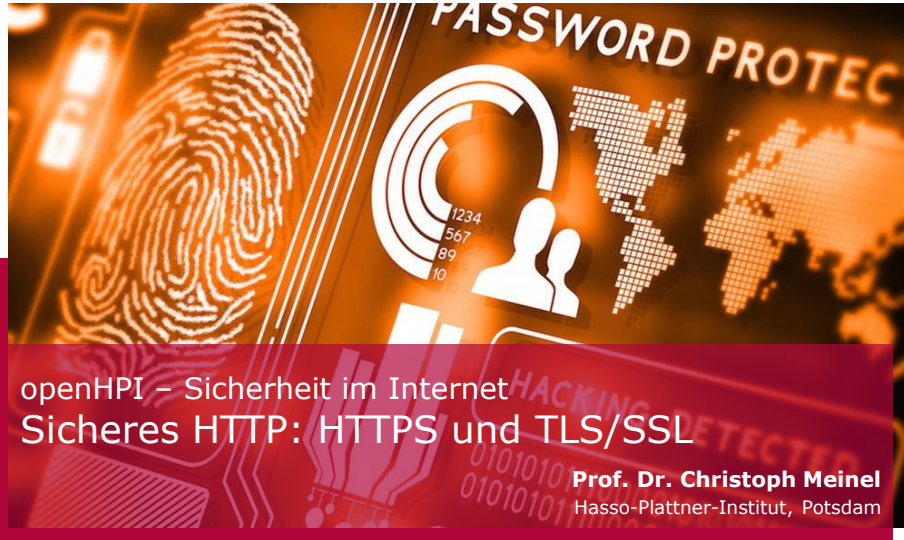




openHPI – Sicherheit im Internet Sicheres HTTP: HTTPS und TLS/SSL

Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut, Potsdam

HTTPS – Hyper Text Transfer Protocol Secure

HTTP ist im Internetprotokollstapel TCP/IP zuständig für die Anbindung des WWW an das Internet.

- HTTP regelt die Kommunikation zwischen Web-Browsern und den Web-Server der Internetdiensteanbieter
- HTTP bietet aber keine Sicherheit
 - Netzwerkpakete werden unverschlüsselt versandt und sind für jeden lesbar, der Zugriff auf das Netzwerk hat
 - keine starke Authentifizierung möglich

Deshalb anstelle von HTTP nach Möglichkeit immer die abgesicherte Variante HTTPS verwenden

HTTPS – Hyper Text Transfer Protocol Secure

HTTPS ist die abgesicherte Variante des HTTP Protokolls für das Web

- HTTPS nutzt Sicherungsschicht TLS/SSL im TCP/IP Stapel
 - TLS/SSL sorgt für die Verschlüsselung der Kommunikationsverbindung und stellt so Vertraulichkeit her
 - TLS/SSL bietet sichere gegenseitige Authentifizierung mittels von Zertifikaten
 - TLS/SSL nutzt effiziente hybride Verschlüsselung
- HTTPS kann so genutzt werden für eine abgesicherte vertrauliche Kommunikation mit starker Authentifizierung
 - Online-Banking
 - E-Shopping
 - Behördenkommunikation ...

HTTPS – Funktionsprinzip

Zusätzliche Sicherheitsfunktionalität von HTTPS gegenüber HTTP wird über (automatisch initiierten) TLS/SSL-Handshake erreicht

1. Browser (Client) sendet „Hello“ Nachricht
2. Web-Server antwortet mit „Hello“
 - In den „Hello“-Nachrichten werden Parameter festgelegt, wie z.B. zu verwendende Verschlüsselungsverfahren, Sitzungs-ID, angewandte TLS/SSL Version, ...
3. Web-Server schickt sein Zertifikat an den Client
4. Browser verifiziert Zertifikat und extrahiert öffentlichen Schlüssel des Web-Servers
5. Zur Verabredung eines Sitzungsschlüssels sendet Browser ein Pre-Master-Secret an Web-Server verschlüsselt mit dessen öffentlichen Schlüssel
6. Web-Server entschlüsselt Secret und beide berechnen daraus den Sitzungsschlüssel

HTTPS – Involvierte Verschlüsselungsverfahren

Die von HTTPS genutzten TLS/SSL Protokolle wenden ganze Palette verschiedener kryptographischer Verfahren an:

- 2-Schlüssel Verfahren zum Austausch des Sitzungsschlüssels:
 - RSA
 - RSA mit Diffie-Hellman Schlüsselaustausch
 - RSA mit Diffie-Hellman basierend auf elliptischen Kurven
 - Digital Signature Algorithm (DSA) mit Diffie-Hellman
- 1-Schlüssel Verschlüsselungsverfahren für Absicherung des Datenverkehrs:
 - AES, Camellia, IDEA, 3DES, ...
- Hash-Verfahren zur Gewährleistung der Datenintegrität:
 - HMAC (Hash based Message Authentication Code) basierend auf SHA-1, SHA-256, MD5

HTTPS – Vor- und Nachteile

Vorteile:

- Jedes Internetprotokoll der Anwendungsschicht kann auf TLS/SSL aufsetzen und dessen Sicherheitsfeatures nutzen.
- TLS/SSL bietet zusätzliche und unabhängige Schicht zwischen Transport- und Anwendungsschicht im TCP/IP-Protokollstapel und bleibt für Anwendungen transparent

Nachteile „Kosten“:

- Verbindungsaufbau braucht Server-seitig mehr Zeit
- Verschlüsselte Daten bieten kaum noch Möglichkeiten der Datenkomprimierung, um Durchsatz zu erhöhen
- TLS/SSL liefert Ende-zu-Ende-Verschlüsselung zwischen zwei Teilnehmern. Für Dienste mit mehreren Teilnehmern, die Pakete (teilweise) mitlesen müssen, ist TLS/SSL ungeeignet

- SSL 1.0 – 1994 von Netscape entwickelt und veröffentlicht
- SSL 2.0 – 1995 integriert in Netscape Navigator
 - Gleiche Schlüssel für Verschlüsselung und Integritätstests
 - Einsatz von MD5 für Integritätstests – gilt heute als unsicher
- SSL 3.0
 - Integration der Zertifikatsvalidierung
 - Nutzung von SHA-1 Algorithmus möglich
 - Hälfte des Master Keys ist abhängig von MD5 Hashfunktion, die als unsicher gilt, weil sie anfällig für Kollisionen ist
- TLS – Verbesserung von SSL 3.0
 - 1999 von IETF als Nachfolger von SSL standardisiert
 - Integritätsschutz mit zusätzlichem Schlüssel verbessert
 - ...

- **OpenSSL**
 - Weit verbreitete Version mit öffentlichem Quellcode
 - Sehr viel Aufsehen in Zusammenhang mit Heartbleed Bug
- **LibreSSL**
 - Aus OpenSSL Projekt hervorgegangen und nun von OpenBSD Projekt weiterentwickelt
- **Microsoft Secure Channel Package**
- **OS X Secure Transport Package**
- **GnuTLS**
 - Frei verfügbare Implementierung unter der GNU-Lizenz
- ...

BEAST – Browser Exploit Against TLS/SSL

- **Voraussetzung:** Verletzung der Same Origin Policy - Einschleusen von generiertem Inhalt in die Webseite möglich
- **Effekt:** Auslesen von Cookies und Stehlen von TLS-Sitzungen

Renegotiation

- Erlaubt einem Angreifer in einer MitM Position eigenen Datenverkehr mit dem eines Clients zu kombinieren, wobei Server fälschlich den Client als Sender identifiziert

Heartbleed

- Gestattet Auslesen des Anwendungsspeichers der TLS/SSL Anwendung, wodurch sensible Daten (Passwörter, Zertifikate,...) preisgegeben werden können
- Unbedingt alte openSSL-Versionen aktualisieren!