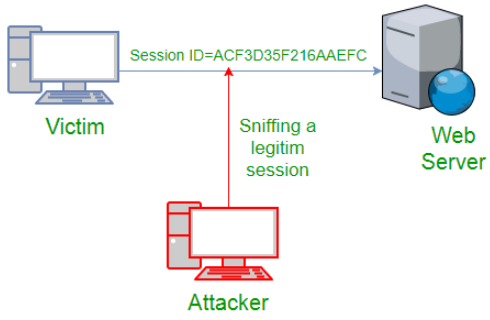
Session Hijacking

Beschreibung

Es bedeutet, dass eine gültige Computersitzung von einem anderen User ausgenutzt wird. Das heisst, der Angreifer kann sich mit der Session als ein anderer User ausgeben. Ziel, um unbefugten Zugang zu Informationen oder Diensten in einem Computersystem zu erlangen. Normalerweise wird dafür ein Session-Cookie von einem User ausgelesen, dieses Cookie dient der Authentifizierung eines Benutzers (Client - Server). Grund warum diese Cookies verwendet werden: HTTP ist zustandslos, jede Anfrage wird einzeln betrachtet. Somit weiss der Server nie, ob es der gleiche Benutzer ist. Mit den Cookies kann eine Sitzung erstellt werden.

Solange man keine verschlüsselte Verbindung zum Server herstellt, können die Cookies ganz einfach via eines ***man-in-the-middle Angriffes (sniffing)*** ausgelesen werden. Dazu müssen die «Opfer» und Angreifer im selben Netzwerk sein. Um die Cookies zu erhalten gibt es noch andere Angriffsarten: ***DNS cache poisoning*** (DNS-Einträge, die auf eine IP-Adresse vom Angreifer leiten), ***XSS (Cross-site scripting)***.



man-in-the-middle / sniffing 1

Damit der Angreifer das Cookie auf der Website verwenden kann, gibt es verschiedene Programme oder man benutzt die Developer Konsole im Browser. In der Konsole kann man mittels JS ein kleines Code-Snippet oder die UI unter «Application» > «Storage» > «Cookies» benutzen, um das Cookie zu setzen. Nach einem Refresh sollte man mit der Sitzung des «Opfers» eingeloggt sein.

Schutzmassnahmen

Zu den Methoden zur Verhinderung von Session Hijacking gehören:

SSL/TLS

Verschlüsselung des zwischen den Parteien übertragenen Datenverkehrs mittels SSL/TLS. Diese Technik wird von webbasierten Banken und anderen E-Commerce-Diensten weitgehend abgelöst, da sie Sniffing Angriffe vollständig verhindert.

VPN und Sicheres WLAN

Falls eine Website kein HTTP über SSL/TLS verwendet kann der Benutzer auf folgende zwei Methoden zurückgreifen

- Vermeidung von ungesicherten WLANs (Beispielsweise im Starbucks)
- Verwendung von VPN-Verbindungen

Session-Cookies

Sehr wichtig ist, dass die Session-Cookies eine lange Zufallszeichenfolge sind, damit ein

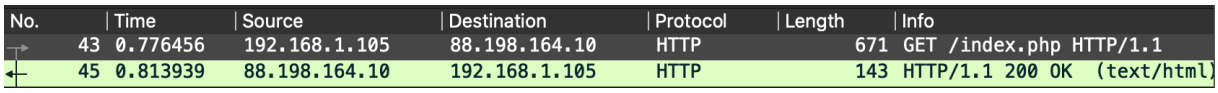
Angreifer nicht durch «trial and error» oder Brute-Force-Angriffe einfach einen gültigen Session-Key erraten könnte. Niemals sollte der gleiche Key für mehrere Sitzungen verwendet werden (z.B. Username und Passwort gehasht).

Alternativ dazu ändern einige Dienste den Wert des Cookies bei jeder einzelnen Anfrage. Dadurch wird das Fenster, in dem ein Angreifer operieren kann, drastisch verkleinert.

Angriffbeispiel

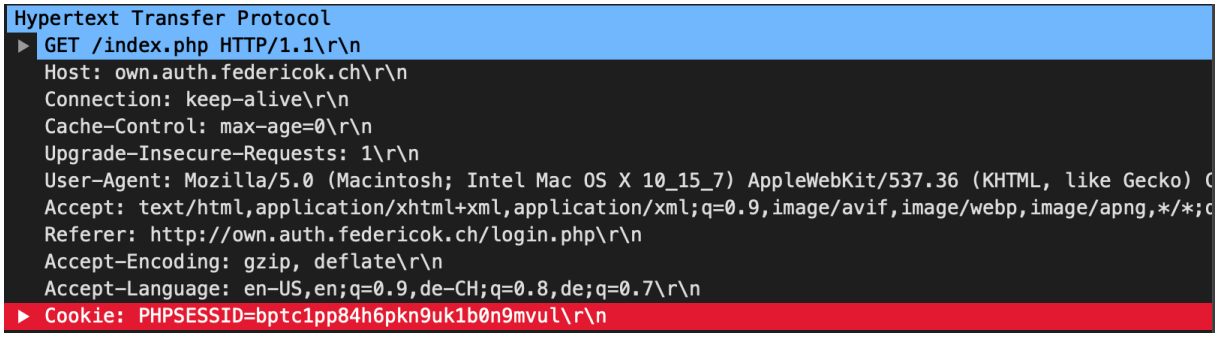
Der Angreifer betreibt Netzwerk Sniffing mit einem Programm.
Vulnerabilität: Unsichere Website über HTTP.
In Wireshark auf HTTP und der IP-Adresse von der Website filtern.

Sobald der User die Website öffnet oder darin navigiert, werden von Wireshark packets abgefangen.



No.	Time	Source	Destination	Protocol	Length	Info
43	0.776456	192.168.1.105	88.198.164.10	HTTP	671	GET /index.php HTTP/1.1
45	0.813939	88.198.164.10	192.168.1.105	HTTP	143	HTTP/1.1 200 OK (text/html)

Darin sieht man jetzt zwei Einträge, der erste ist die HTTP-GET Anfrage an den Server. Die zweite besteht dann aus der Rückgabe von Server, also in diesem Fall das index.php (HTML) file. Bei dem ersten Eintrag kann man nun überprüfen, ob vom Client ein Session-Cookie mitgesendet wurde.



```
Hypertext Transfer Protocol
▶ GET /index.php HTTP/1.1\r\n
  Host: own.auth.federicok.ch\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) C
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;c
  Referer: http://own.auth.federicok.ch/login.php\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9,de-CH;q=0.8,de;q=0.7\r\n
▶ Cookie: PHPSESSID=bptc1pp84h6pkn9uk1b0n9mvul\r\n
```

Unter Hypertext Transfer Protocol sieht man jetzt die request headers. Und siehe da, ein Cookie namens «PHPSESSID» wurde dem Server übermittelt.

Der Angreifer kann nun ganz einfach seinen Browser mit derselben Website öffnen und die Entwickler Tools öffnen. Darin hat er die Möglichkeit unter «Application» > «Storage» > «Cookies» bestehende Cookies anzuschauen, sowie neue hinzuzufügen. Er kann nun ein Cookie mit dem Key «PHPSESSID» und dem Value «bptc1pp84h6pkn9uk1b0n9mvul» erstellen. Cookies kann man auch über die JS Konsole erstellen und auslesen. Das JS Snippet dafür ist: `document.cookie = 'KEY=VALUE';`

Nach Neuladen der Website ist der Angreifer mit dem Benutzeraccount des Opfers eingeloggt. Sobald sich jedoch der Benutzer (Opfer) ausloggt, ist auch die Session vom Angreifer ungültig und somit wird er auch ausgeloggt.

Die Schutzmassnahme gegen diesen Angriff kann nicht in Code angegeben werden, da es eine Serverkonfiguration ist (SSL/TLS Zertifikat). Aber heutzutage gibt es kostenlose SSL Zertifizierungsstelle, wie Let’s Encrypt.