



openHPI – Sicherheit im Internet Angriffe auf Dienste

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

Angriffe auf Dienste

Über das weltweite Internet bekommt man Zugang zu ganz verschiedenen, populären Diensten, z.B.:

- ☐ WWW – World Wide Web
- ☐ Email
- ☐ Dateitransfer (FTP)
- ☐ Rechner-Fernsteuerung – Remote Control –, z.B. TELNET/SSH
- ☐ ...
- Jeder dieser Dienste basiert auf einer eigens dafür entwickelten Software, die wie alle Software Schwachstellen haben kann
- Da Angreifer im Internet solche Schwachstellen aufgefunden haben und ausnutzen, ist es wichtig zu wissen, diese Schwachstellen und Angriffsmöglichkeiten zu kennen

Ziel des Angriffs:

- Einschränkung bzw. Abbruch der Funktionstüchtigkeit des Diensts: Arbeitsverweigerung des Dienstes – **Denial of Service – DoS**
- Dienst ist für interessierte Nutzer nicht mehr verfügbar

Verbreitete Angriffsmethode:

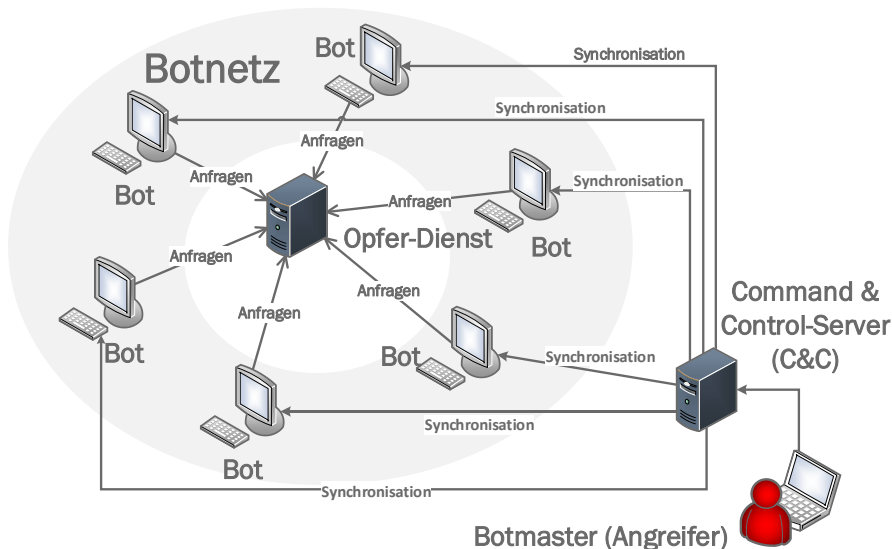
- **Bombardierung** eines Dienstes mit Anfragen
- Jede **Anfrage eines Clients** an einen Server **bindet** für die Zeit der Verarbeitung der Anfrage **Server-Ressourcen**
 - Speicher, Netzwerkbandbreite, Rechenleistung
- Vielzahl von gleichzeitigen Anfragen kann Server überlasten
- Anfragen der Angreifer sind so gestaltet, dass über eine möglichst lange Zeit möglichst viele Ressourcen gebunden sind

Serverleistung ist heute meist sehr groß → einzelner Angreifer kann keinen erfolgreichen DoS-Angriff ausführen, darum:

Verteilter (Distributed) Denial-of-Service-Angriff – DDoS:

Angreifer oder mehrere koordinierte Angreifer versenden Flut von Anfragen aus verschiedenen Quellen

- Vielzahl an Angriffs-Quellen werden synchronisiert über Absprachen in Foren oder kleine Helferprogramme bzw. Schadprogramme
- Dienst wird durch die Vielzahl der Anfragen überlastet
- DDoS werden sehr häufig von **Botnets** ausgeführt
 - Zentraler Kommandoserver (**C&C-Server**) gibt Startschuss für Angriff, also die Aussendung der Anfragen
 - **Trojaner** auf gekaperten Computern (teils mehrere Millionen) führen Angriff dann zeitsynchronisiert durch



Einige spektakuläre Beispiele:

Februar 2014:

- DDoS-Angriff auf die „CloudFlare“ – Infrastruktur eines Anbieters von DDoS-Schutz mit bisher höchster Last von 400 Gbit/s

März 2013:

- DDoS-Angriff auf „Spamhaus.org“, ein bekannter Betreiber einer schwarzen Liste für Spam-Emailadressen

Juli 2011:

- DDoS-Angriff auf VISA, Mastercard und Paypal, nachdem diese die Konten von Wikileaks eingefroren hatten

April 2011:

- DDoS-Angriff auf Sony-Playstation-Netzwerk, weil sich Sony gegen das Brechen von Schutzmaßnahmen (Jailbreak) der Playstation 3 gestemmt hat

Versand von Phishing- und Spam-Nachrichten

- Phishing und Spam machen Großteil (80%) des Email-Verkehrs aus !
- Enthalten meist Werbung oder betrügerische Nachrichten
- Absenderadresse ist meist gefälscht (Email-Spoofing)
- Versand erfolgt typischerweise über
 - Botnets (in diesem Fall meist von den Email-Konten der Nutzer der gekaperten Bots)
 - Open-Relay-Server (fehlerhaft konfigurierte Email-Server, die Versand für Absender ohne Authentifikation zulassen)

Versand von Schadcode

- Schadcode wird versteckt in gespoofter Email verschickt
- Schadcode ist versteckt z.B. in einem Anhang der Email
- Opfer aktiviert Schadcode unwissentlich manuell oder sein Email-Programm aktiviert Schadcode automatisch aufgrund einer Schwachstelle

Denial-of-Service-Angriffe auf Email-Server

- Server können nur beschränkte Zahl offener (TCP) Verbindungen bedienen
- Angreifer veranlassen massenhaften Verbindungsaufbau zu Email-Servern
- Daten werden dazu bewusst nur sehr langsam übertragen, um Verbindung möglichst lange offen zu halten
→ **Slowloris-Angriff**

Brute-Force-Angriff

- Angreifer probiert systematisch, sich mit allen möglichen Nutzer-Passwort-Kombinationen beim Email-Server anzumelden
- Erfolgreich, weil einige Dienste die Zahl der Login-Versuche nicht limitieren
- Angreifer kann dann mit geknackten Passwörtern die Nutzerkonten übernehmen ...

Viele der genannten Angriffe auf Email-Systeme können mit verlässlicher **Authentifizierung** verhindert werden

- **SMTP-AUTH** erlaubt Authentifizierung bei Email-Servern vor dem Versand mit Login/Passwort-Kombination (heute üblich bei Emailservern)
- Nutzung von **SSL/TLS** ermöglicht Authentifikation beim Email-Server mittels digitaler Zertifikate
- **PGP** (Pretty Good Privacy) oder **SMIME** erlauben Ende-zu-Ende-Verschlüsselung der Emails und zertifikatsbasierte Authentifikation beim Empfänger

→ Geeignete Authentifikation **reduziert Wirksamkeit** der genannten Angriffe erheblich

Remote Code Execution (dt. **entfernte Codeausführung**)

Bezeichnet Ausnutzung von Schwachstellen eines entfernten Dienstes mit dem Ziel, Schadcode auf dem Server des Diensteanbieters zur Ausführung zu bringen

Ziel:

- Angreifer strebt einen **vollständigen Zugriff** auf das dienst anbietende System an → Remote Code Execution **potenziell sehr gefährlicher Angriff**

Hauptursache

- Mangelhafte Filterung von Nutzereingaben
 - Betroffen sind also grundsätzlich alle Programme (oder Webseiten), die freie Nutzereingaben akzeptieren
 - In Texteingabe wird Schadcode versteckt, der sich wegen mangelnder Filterung in Programmcode einbetten kann

Buffer Overflow – Variante von Remote Code Execution

- Angreifer überschreitet bei Benutzereingabe Maximallänge
- Eingabe wird zur Verarbeitung vom Programm im Speicher abgelegt, genauer in einem sogenannten „Buffer“
- Programm(ierer) müsste vorm Speichern der Eingabe überprüfen, ob maximal zulässige Länge eingehalten wurde und sonst Eingabe abschneiden oder ganz verwerfen)
- Bei unabgeschnittener Überschreitung der Länge kann Speicher überschrieben werden → **Buffer Overflow**
- Besonders fatal, wenn es Angreifer schafft, Speicher so zu überschreiben, dass Programmfluss dafür sorgt, dass überschriebener Speicherbereich ausgeführt werden
- **Gezielte Buffer Overflows** sind „hohe Kunst“, also **sehr schwierig** umzusetzen – aber bei Erfolg **sehr gefährlich**

Injection – Variante der Remote Code Execution

- Angreifer bettet ausführbaren Programmcode in seine Nutzereingabe ein – **Code Injection**, z.B.
 - SQL-Code zur Abfrage der Datenbank
 - Javascript-Code in HTML-Eingabefeldern
- Aufgabe des Entwicklers
 - **Eingaben** auf unerlaubte Code Injection zu **überprüfen**
 - Darauf zu achten, dass Eingaben **niemals direkt ausgeführt** werden
- Geschieht dies nicht, wird injizierter Code im Programm ausgeführt. **Folgen sind ähnlich fatal**, wie beim Overflow
 - Angreifer kann **beliebigen Code** in angegriffenem Dienst ausführen
 - Meist genutzt, um eine **Hintertür zum Server** zu öffnen

Bekannter Vorfall: Heartbleed (1/5)

Heartbleed ist die Bezeichnung einer Schwachstelle in der **weit verbreiteten** SSL/TLS-Implementierung **OpenSSL**

■ **Erinnerung an SSL/TLS:**

- Sicherungsschicht im Internetprotokollstapel
- Sorgt für Ende-zu-Ende Verschlüsselung und sichere Authentifizierung
- OpenSSL ist verbreitete Open Source Implementierung von SSL/TLS ...



Bekannter Vorfall: Heartbleed (2/5)

Heartbleed ist die Bezeichnung einer Schwachstelle in der **weit verbreiteten** SSL/TLS-Implementierung **OpenSSL**

- Heartbleed erlaubte das Auslesen von kleinen, zufälligen Teilen des Programmspeicher bei entfernten Diensten, die OpenSSL verwenden
 - Bei ausreichender Wiederholung konnte sogar gesamter Programmspeicher ausgelesen werden
 - **Besonders kritisch:** Auch **privater Schlüssel** des Servers und **Passwörter** befinden sich im Programmspeicher!
 - Schwachstelle wurde schnell behoben, Patches veröffentlicht
- **Achtung:** Ältere Versionen, die evtl. noch in Betrieb sind, müssen unbedingt aktualisiert werden



Bekannter Vorfall: Heartbleed (3/5)

Heartbleed ist die Bezeichnung einer Schwachstelle in der weit verbreiteten SSL/TLS-Implementierung **OpenSSL**

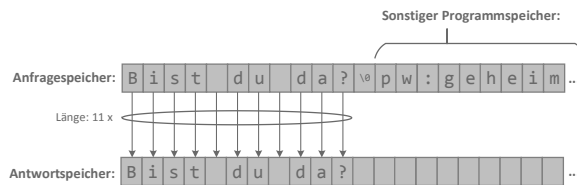


- Hängt nicht mit fehlerhafter Umsetzung in Kryptografiefunktionen zusammen sondern mit:
 - Zusatzfunktion im SSL/TLS-Protokoll, die permanent einen „Herzschlag“ (Heartbeat) des Gegenübers abfragt
 - Herzschlaganfrage wird mit gleichem Inhalt wie Anfrage beantwortet

Anfrage:



Antwort:



Bekannter Vorfall: Heartbleed (4/5)

Heartbleed ist die Bezeichnung einer Schwachstelle in der weit verbreiteten SSL/TLS-Implementierung **OpenSSL**



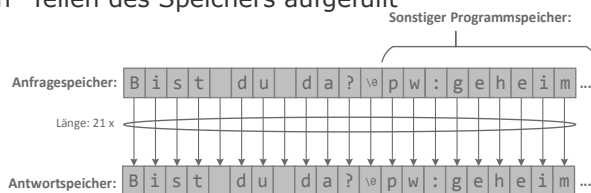
Problem:

- Heartbeat-Anfrage enthält nicht nur „Anfragetext“, sondern auch Längenangabe von Anfrage und gleichlautender Antwort
- OpenSSL prüfte nicht, ob tatsächliche Länge der gesendeten Anfrage mit angegebener Länge übereinstimmt
- Heartbeat-Antwort wurde bis zur angegebenen ungeprüften Länge mit „zufälligen“ Teilen des Speichers aufgefüllt

Anfrage:



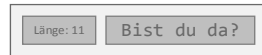
Antwort:



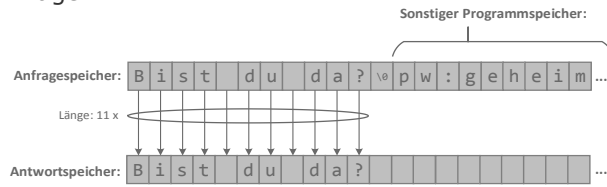
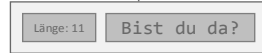
Bekannter Vorfall: Heartbleed (5/5)

Normale Heartbeat-Anfrage:

Anfrage:



Antwort:



Bösartige Heartbeat-Anfrage:

Anfrage:



Antwort:

