Juventus Technikerschule HF

Hacking Exposed

#6 Websecurity II

Pascal Knecht

Video 0: Überblick

Wichtiger Hinweis

- Dies ist eine Lehrveranstaltung.
- Die im Rahmen der Hacking-Exposed-Vorlesung vermittelten Kenntnisse sollen dazu beitragen, dass Sie Informationssicherheitsaspekte beachten und in Ihren Projekten berücksichtigen.
- Die HE-Vorlesung ist keineswegs als Anstiftung zum Hacken zu verstehen.

Inhalt heute Abend

- OWASP Top 10
- Cross-Origin Resource Sharing (CORS)
- Cross-Site Request Forgery (CSRF)
- SQL-Injection
- OpenSSL Heartbleed

Ziele

- Sie kennen die OWASP Top 10 und wissen, welche Themen die einzelnen Kategorien abdecken.
- Sie kennen CORS und verstehen das Zusammenspiel mit SOP
- Sie kennen die Funktionsweise und die Voraussetzungen für die Websecurity Attacken CSRF, XSS und SQL Injection.
- Sie sind in der Lage, einfache Attacken im Übungslabor auszuführen und verstehen ihre Funktionsweisen.
- Sie können mit dem Analyse Tool Burp Suite Webservices untersuchen.

#01 OWASP Top 10

Video 1:
OWASP Top 10

Was sind die Top 10 Websecurity Schwachstellen?

1) Injection
   (8 Min)
2) Broken Authentication
   (10 Min)
3) Sensitive Data Exposure
   (10 Min)
4) XML External Entities (XEE)
   (10 Min)
5) Broken Access Control
   (10 Min)
6) Security Misconfiguration
   (11 Min)
7) Cross-Site Scripting (XSS)
   (11 Min)
8) Insecure Deserialization
   (9 Min)
9) Using Components with Known Vulnera bilities
   (10 Min)
10) Insufficient Logging & Monitoring
   (14 Min)

OWASP Top 10 (2017) => OWASP Top 10 (2021)

Deutsche Übersetzung OWASP Top 10

#02 Cross-Origin Resource Sharing (CORS)

Video 2: CORS

Cross-Origin Resource Sharing

- SOP verhindert, dass Cross-Side Responses in den Browser-Context weitergegeben werden.
  - Nachladen von Schadsoftware kann so verhindert werden.
- Sollen Cross-Side Requests und Responses dennoch erlaubt sein kann CORS helfen.
  - Server-Admin / Entwicklerin kann so bspw. ein API gezielt für ausgewählte Seiten erreichbar machen.
- CORS im Detail bei MDN

CORS Übersicht

XMLHttpRequest

```
var oReq = new XMLHttpRequest();
oReq.addEventListener("load", reqListener);
oReq.open("GET", "http://www.bar.lab/example.txt");
oReq.send();
```

JavaScript

```
HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 00:23:53 GMT
Server: Apache/2
Access-Control-Allow-Origin: *
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/xml

Hello World...
```

```
GET /example.txt HTTP/1.1
Host: www.bar.lab
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14;
rv:71.0) Gecko/20100101 Firefox/71.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Connection: keep-alive
Origin: http://www.foo.lab
```

Soweit identisch
Mit SOP

www.bar.lab

CORS


Client
Server
GET /doc HTTP/1.1
Origin: https://foo.example
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://foo.example

CORS Beispiele

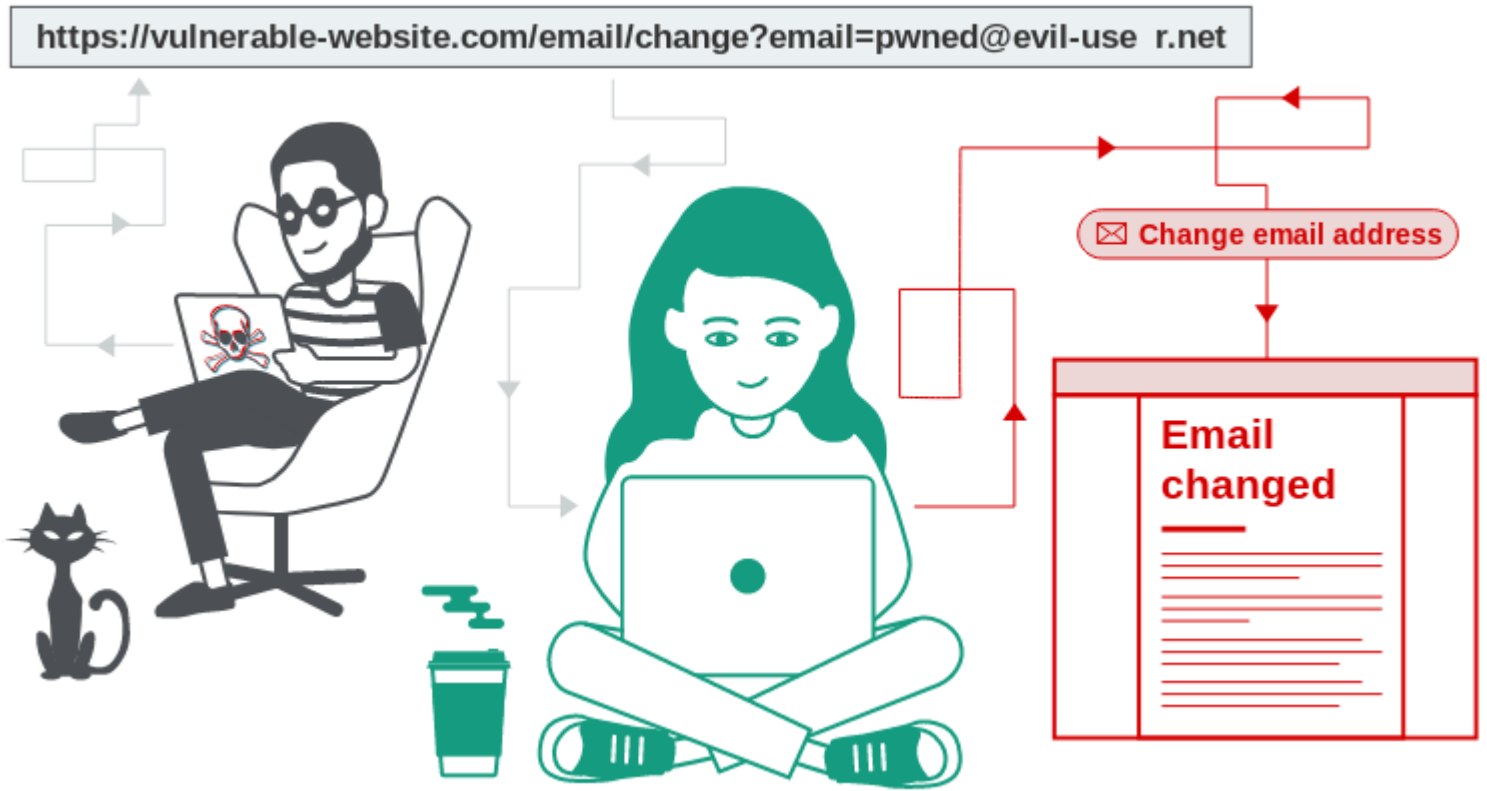
- Öffentliche Schriftart
  - Soll von allen Webseiten eingebunden werden können: `Access-Control-Access-Control-Allow-Origin: *`
- Bilder API einer Suchmachine A
  - Läuft auf Server B soll das API ausschliesslich Server A verfügbar machen: `Access-Control-Allow-Origin: A`
- Kundenportal einer Bank
  - Soll prinzipiell möglichst wenig Angriffs-/Interaktionsfläche bieten: kein ACAO-Header

#03 Cross-Site Request Forgery (CSRF)

Video 3: CSRF

Video: CSRF Self Contained

CSRF im Überblick



https://portswigger.net/web-security/csrf

Bedrohungslage durch CSRF

- Ausführung vom User unbeabsichtigter Aktion
  - Email ändern, Passwort ändern, Geldtransaktion auslösen, Löschen des Accounts etc.
- Erhaltung voller Kontrolle über den User-Account
- Aktuelles Beispiel: Cert-Bund war anfällig für CSRF-Angriff (30. Okt. 2019)

Funktionsweise von CSRF

- Zwingende Voraussetzung
  - Relevante Aktion auf Website
  - Cookie-Based Session Handling
  - Keine unbekannten/unvorhersagbaren Request Parameter

- Erfüllt beispielsweise alle Voraussetzungen:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE

email=wiener@normal-user.com
```

CSRF-Attack-Server

- Auf einem Webserver wird folgender Exploitcode bereitgestellt:

```
<html><body>
  <form action="https://vulnerable-website.com/email/change" method="POST">
    <input type="hidden" name="email" value="pwned@evil-user.net" />
  </form>
  <script>document.forms[0].submit();</script>
</body></html>
```

CSRF-Schwachstelle ausnutzen

- Angreifer hinterlegt bösartiges HTML auf Attack-Server.
- Angreifer lässt Opfer URL auf dieses HTML zukommen.

- Bei einer Self-contained Attacke wird eine Schwachstelle in einem Webservice lediglich mit einem GET-Request ausgenutzt. Kein Attack-Server nötig.
  - `https://vulnerable-website.com/email/change?email=pwned@evil-user.net`

Verhindern von CSRF-Attacken

- Häufigste Lösung sind CSRF-Tokens mit folgenden Eigenschaften
  - Unvorhersagbarkeit des Token
  - An User-Session gebunden
  - Vom Server immer validiert vor jeglicher Aktion
- `<input type="hidden" name="csrf-token" value="CIwNZNlR4XbisJF39I8yWnWX9wX4WFoz" />`

#04 SQL-Injection

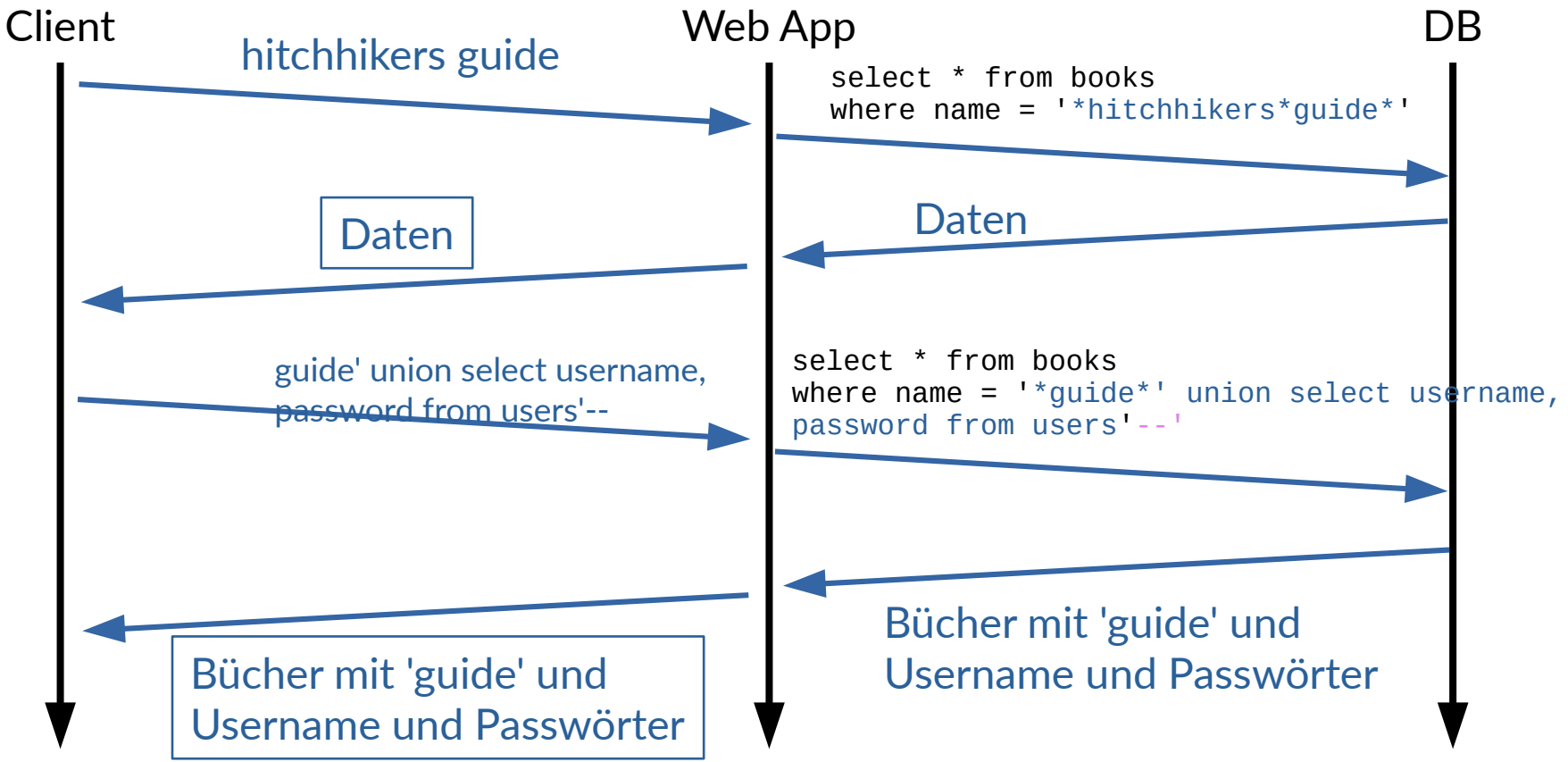
Video 4: SQL Injection

Video: SQL Injection Demo

SQL-Injection Idee

- Webapplikation mit SQL-Datenbank
- Über Formular wird eine Query an die DB formuliert
- Durch eine parsing Schwachstelle können Anfragen formuliert werden, sodass bspw. sensitive Daten wie Usernamen von der Datenbank zurück an die Webapplikation geschickt werden.

SQL-Injection allgemein



Selbststudium

- Erarbeiten Sie sich den Inhalt zum Thema SQL-Injection im Selbststudium.
- Nutzen Sie Ihr Vorwissen aus dem Datenbank Modul.
- Dieses Video (10 Min) gibt einen guten Einstieg in die Thematik.
- Dieses Video (17 Min) geht mehr auf den Prozess zur Erstellung eines SQL-Statements ein um eine SQL-Injection auszunutzen.

Labor 6

Übungen: HE6

Labor: github.com/ryru/HackingExposed

#05 OpenSSL Heartbleed

Video 5:
OpenSSL Heartbleed

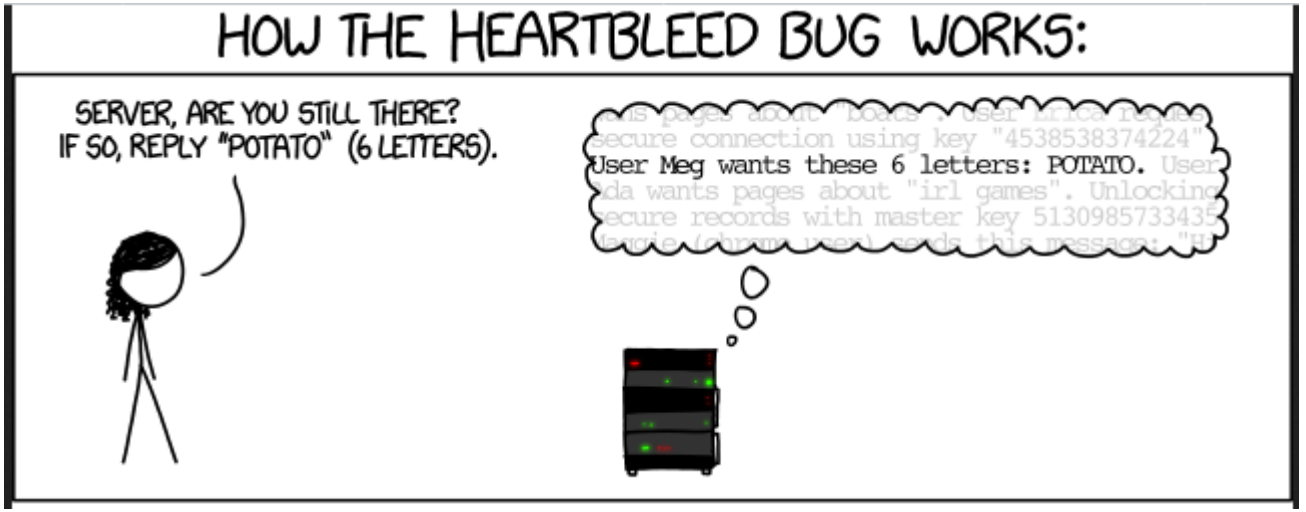
Heartbleed Überblick

- Implementationsproblem von OpenSSL
- Betroffene Version 1.0.1 ab Dezember 2011
  - Commit: https://github.com/openssl/openssl/commit/4817504d069b4c5082161b02a22116ad75f822b1
- Problem liegt in der TLS Erweiterung Heartbeat
  - https://de.wikipedia.org/wiki/Heartbleed
  - https://nvd.nist.gov/vuln/detail/CVE-2014-0160
- Fix in Version 1.0.1g ab April 2014
  - Commit: https://github.com/openssl/openssl/commit/731f431497f463f3a2a97236fe0187b11c44aead
  - Comment: «A missing bounds check in the handling of the TLS heartbeat extension can be used to reveal up to 64k of memory to a connected client or server.»

TLS Heartbeat Erweiterung

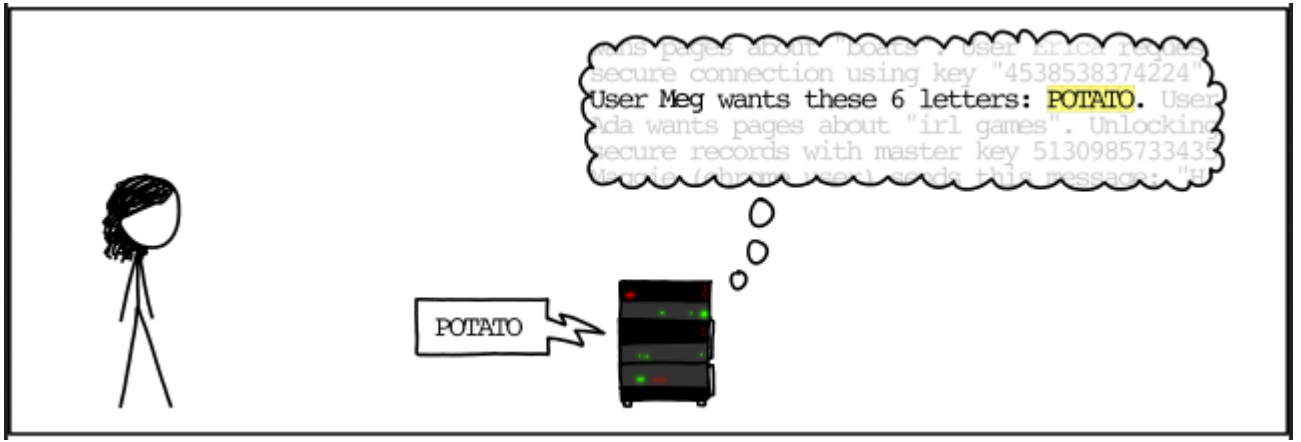
- Idee: Steht Verbindung noch?
  - Peer schickt bis zu 16 Kilobyte beliebige Daten
  - Gegenstelle schickt dieselben Daten zurück
- Problem: Fehlender Check der Länge
  - Client kann bis zu 64 Kilobytes Daten aus RAM lesen

Heartbleed erklärt



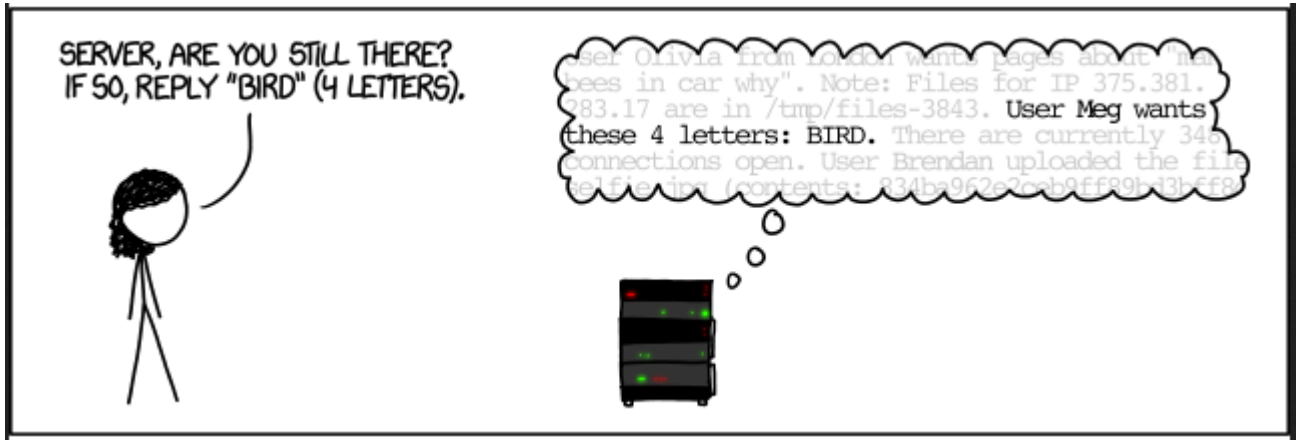
Server, bist du noch da?
Falls ja, antworte mir bitte «POTATO» (6 Buchstaben)

Heartbleed erklärt



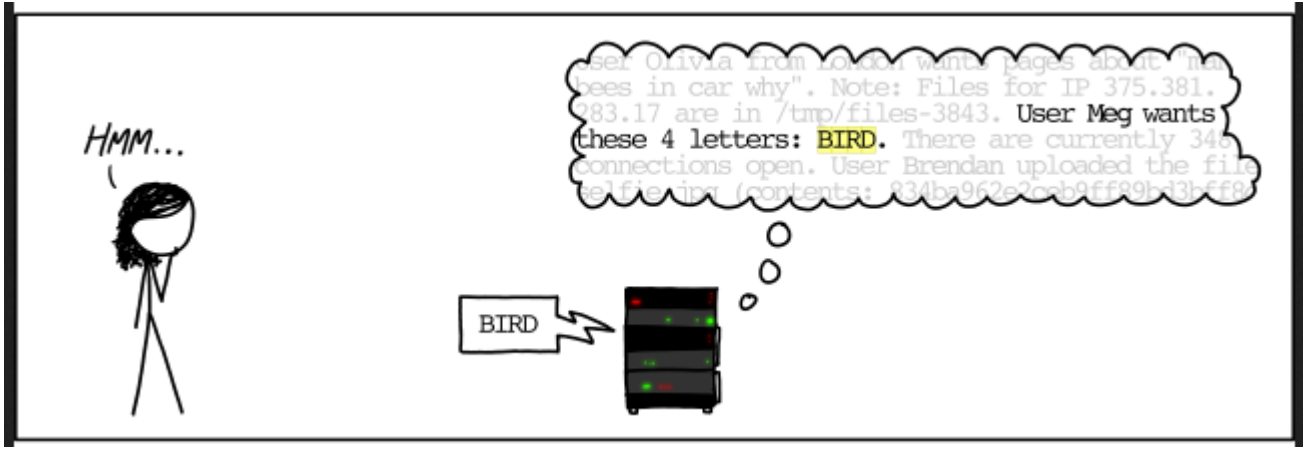
POTATO

Heartbleed erklärt



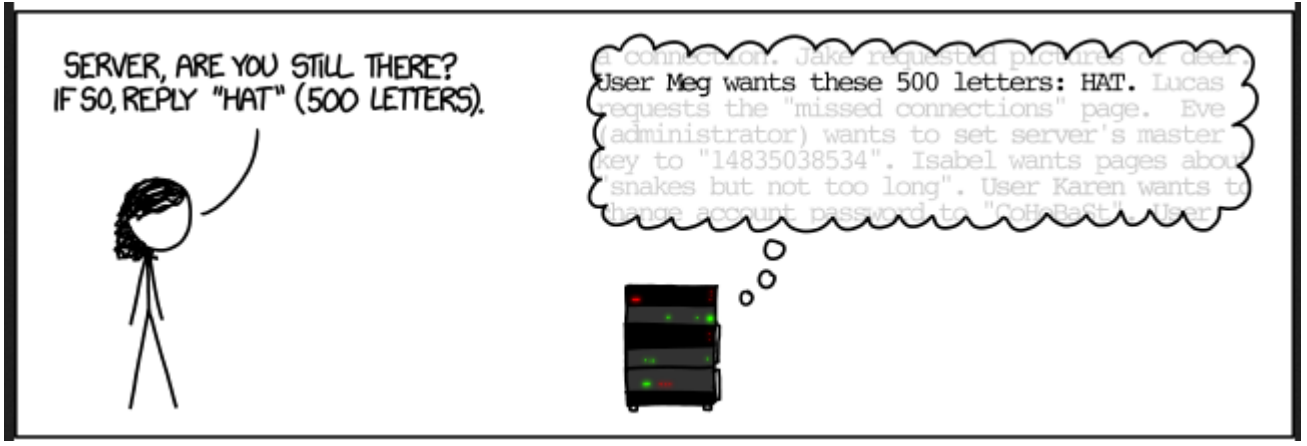
Server, bist du noch da?
Falls ja, antworte mir bitte «BIRD» (4 Buchstaben)

Heartbleed erklärt



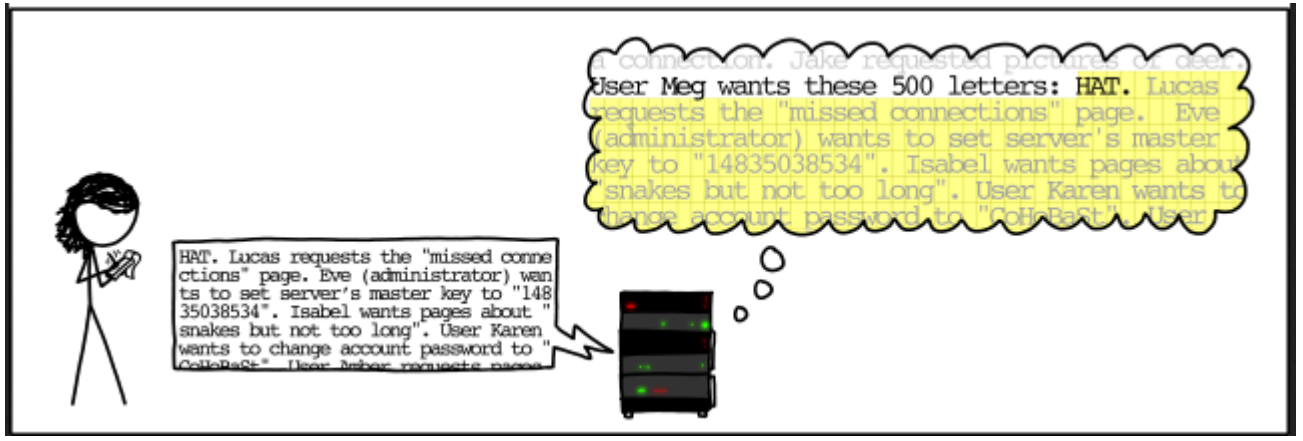
BIRD

Heartbleed erklärt



Server, bist du noch da?
Falls ja, antworte mir bitte «HAT» (500 Buchstaben)

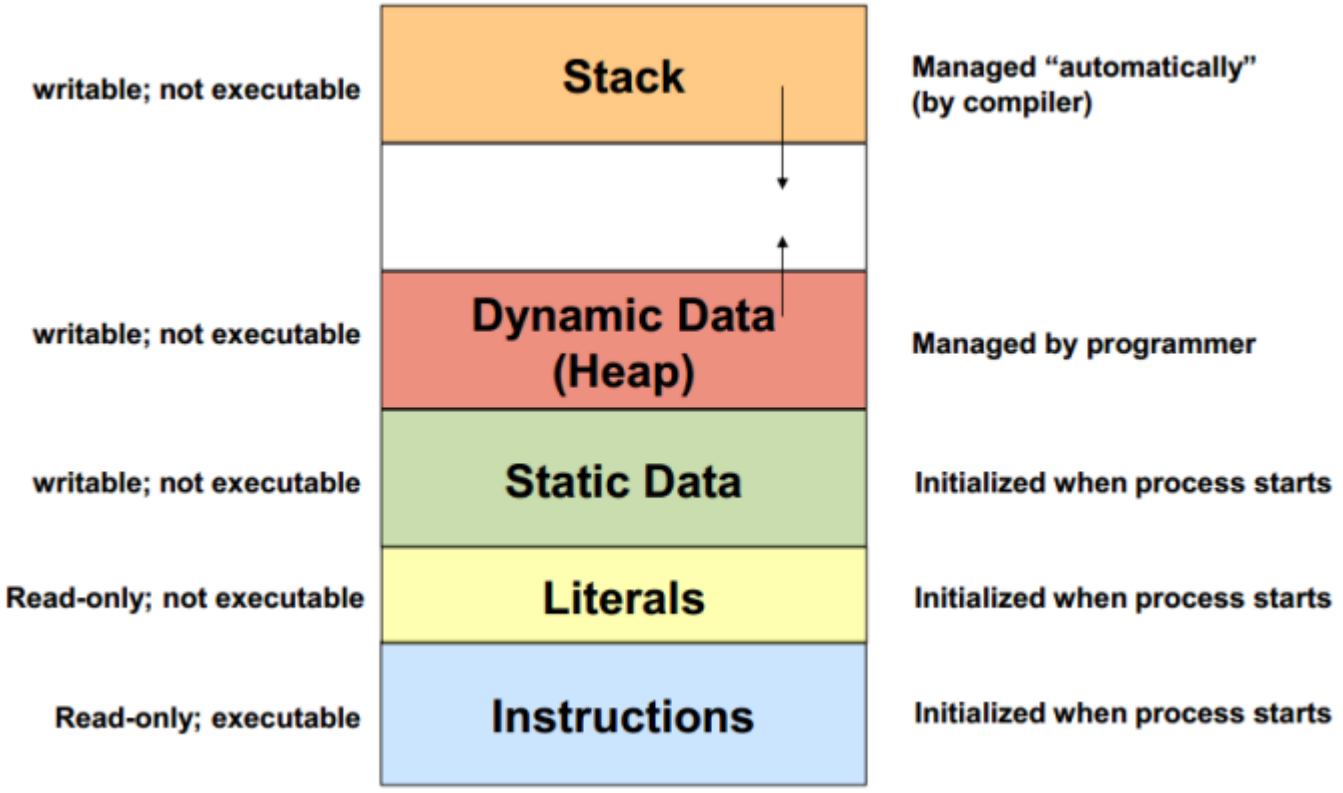
Heartbleed erklärt



HAT. Lucas fragt nach «verlorene Seite» Übersicht. Eve (Administratoring) möchte den Server Master Key auf «1483505853⁴» setzen. Isabel…

https://xkcd.com/1354/

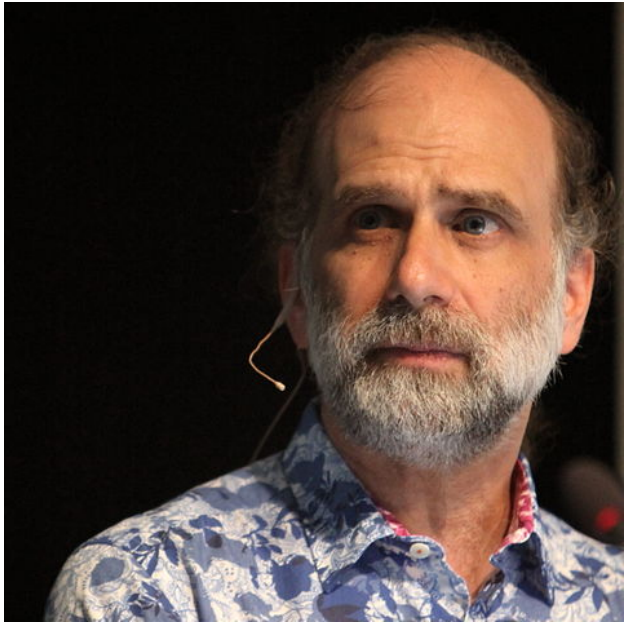
Exkurs Stack und Heap



https://stackoverflow.com/questions/32418750/stack-and-heap-locations-in-ram

Bruce Schneier

- «Catastrophic is the right word. On the scale of 1 to 10, this is an 11.»
- «Katastrophal ist das richtige Wort. Auf einer Skala von 1 bis 10 ist dies eine 11.»



Code

- Heartbeat Struktur

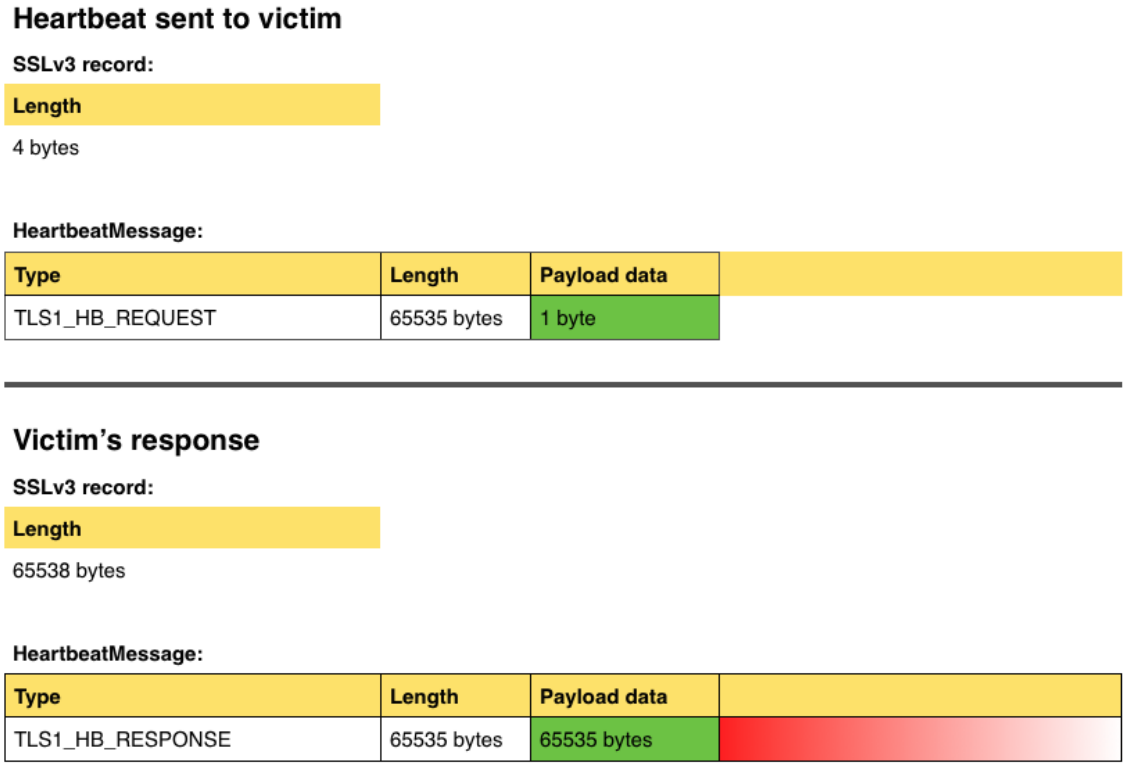
```
struct
{
  HeartbeatMessageType type;
  uint16 payload_length;
  opaque payload[HeartbeatMessage.payload_length];
  opaque padding[padding_length];
} HeartbeatMessage;
```

- Record Struktur

```
struct ssl3_record_st
{
  unsigned int length;    /* How many bytes available */
  [...]
  unsigned char *data;    /* pointer to the record data */
  [...]
} SSL3_RECORD;
```

Demo

- Anfrage
- Antwort



- Demo: https://youtu.be/PaAKek4WvyY?t=45 (bis 1:55 Min)
- Weitere Details: https://www.theregister.co.uk/2014/04/09/heartbleed_explained/

#06 Diplomprüfung Hacking Exposed

Diplomprüfung Teil Hacking Exposed

- Genaues Datum wird noch bekanntgegeben
- Theorieprüfung 60 Minuten (schriftlich)
- Stoff: Alles was in den sechs Vorlesung HE besprochen wurde
  - Slides
  - Übungen
  - Labor
- Erlaubte Hilfsmittel: Eigene Zusammenfassung 2x A4 also 4 Seiten total
  - Papier, Stift und Taschenrechner erlaubt aber kein Notebook und keine alten Prüfungen

Das war's! Vielen Dank für's Mitmachen

Modulende

CC BY-SA 4.0