IT-Sicherheit – Sicherheit vernetzter Systeme

Vorlesung im Wintersemester 2024/2025 (LMU)

21.11.2024
WS 2024/2025

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

21.11.2024
WS 2024/2025

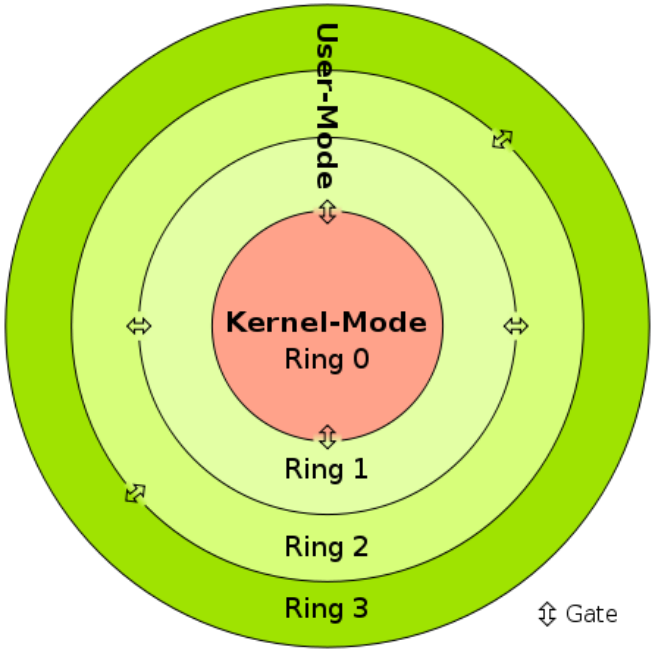
Übungsblatt 4:
Aufgabe 1: (T) Rootkits

Nachdem ein Angreifer erfolgreich Zugang zu einem IT-System, etwa durch das Ausnutzen einer dort vorhandenen Schwachstelle, erlangen konnte, wird dort meist ein Rootkit installiert.

a) Es wird grundsätzlich zwischen zwei Varianten von Rootkits unterschieden: **User-Mode-** und **Kernel-Mode-**Rootkits. Erläutern Sie diese kurz.

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

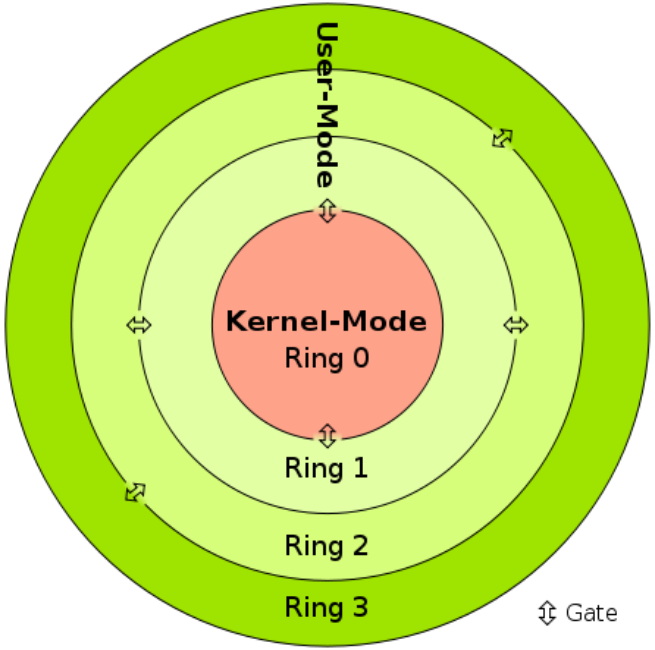
- User-Mode-Rootkit
  - Wird mit Rechten des angemeldeten Nutzers ausgeführt (Ring 3)
  - Können sich als dynamisch gelinkte Bibliothek (z.B. .DLL) an andere Prozesse hängen
  - Manipulieren von Sicherheitseinstellungen



https://en.wikipedia.org/wiki/File:CPU_ring_scheme.svg

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

- Kernel-Mode-Rootkit
  - Wird mit höchsten Systemrechten ausgeführt (Ring 0)
  - Modifiziert den Kernel durch eigene Module oder Gerätetreiber
  - Führt (eventuell) zu Instabilitäten des Systems



https://en.wikipedia.org/wiki/File:CPU_ring_scheme.svg

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

b) Wie unterscheidet sich ein Rootkit von anderer Malware, z.B. Viren, Würmer und Trojanischen Pferden?

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

b) Wie unterscheidet sich ein Rootkit von anderer Malware, z.B. Viren, Würmer und Trojanischen Pferden?

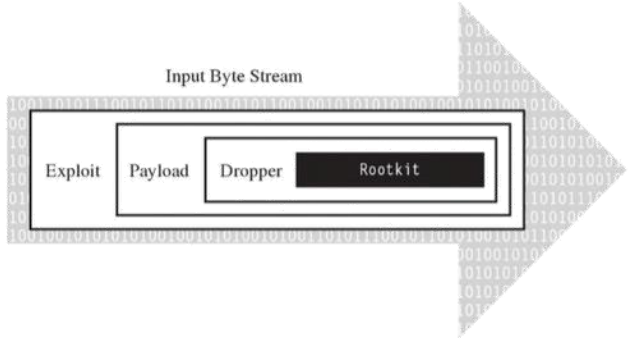
- Rootkits…
  - verbreiten sich nicht, wie z.B. Würmer, **automatisch** weiter.
  - führen keinen Massenversand von Spam oder (D)DoS durch.
  - stellen verdeckten Zugang zu einem System bereit.
- Allerdings in der Praxis oft keine klare Abgrenzung möglich.

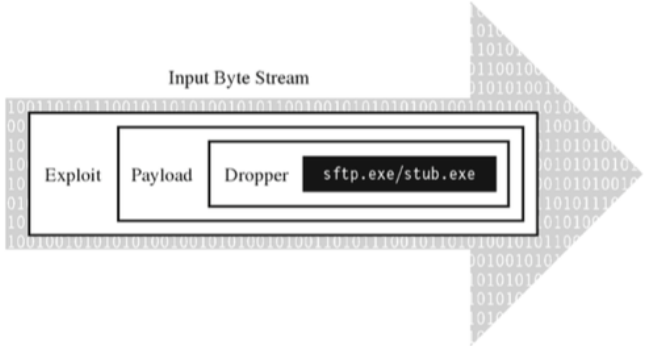
Übungsblatt 4:
Aufgabe 1: (T) Rootkits

c) Rootkits verfügen im Allgemeinen über eine sogenannte Dropper-Komponente. Welchem Zweck dient diese Komponente. Was versteht man unter einem Multistage Dropper?

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

- Dropper
  - Installation und Verstecken des Rootkits
  - Selbstzerstörung des Droppers
- Multi-Stage-Dropper
  - Installation des Rootkits über das Netz (z.B. per SFTP)
  - Dropper lädt weiteren Dropper per SFTP nach





Übungsblatt 4:
Aufgabe 1: (T) Rootkits

(d) Charakteristisch für Rootkits sind sogenannte Anti-Forensik-Maßnahmen.

Erläutern Sie folgende Maßnahmen

- Data Destruction
- Data Concealment
- Data Fabrication

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

- Data Destruction
  - „Entsorgung“ nicht länger benötigter Rootkit-Komponenten/Daten/Spuren
  - Sabotage der von Forensic-Tools verwendeten Datenstrukturen
  - „Dissolving batch files“
    → Löschen eines Binary von der Platte per Skript

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

- Data Concealment
  - „Security by Obscurity“

    → Verstecken des Rootkits in speziellen Bereichen, z.B. System Volume Information

  - Active concealment

    → Modifikation des OS nach erfolgreicher Rootkit-Installation

Übungsblatt 4:
Aufgabe 1: (T) Rootkits

- Data Fabrication
  - Flooding mit falschen und irreführenden Informationen, um den Forensiker zu „beschäftigen“ → Ablenkung
  - Beispiel: Identifikation eines Angreifers durch File-Checksums angestrebt
    → Ändern einer sehr großen Anzahl von Dateien

Übungsblatt 4:
Aufgabe 2: (T) XSS

21.11.2024
WS 2024/2025

Übungsblatt 4:
Aufgabe 2: (T) XSS

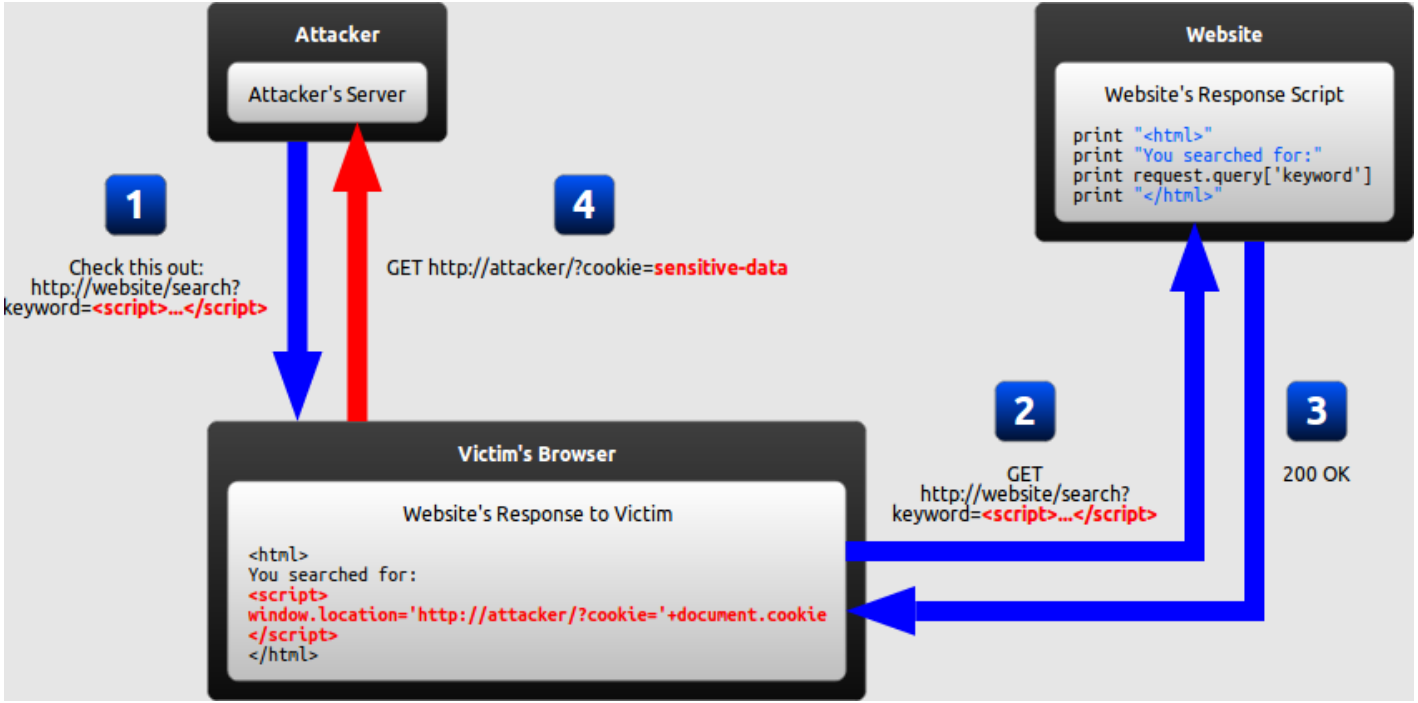
(a) In der Vorlesung wurden drei verschiedene Arten von Cross-Site-Scripting (XSS) vorgestellt.
Welche der Varianten besitzt das höchste Bedrohungspotential?

Übungsblatt 4:
Aufgabe 2: (T) XSS

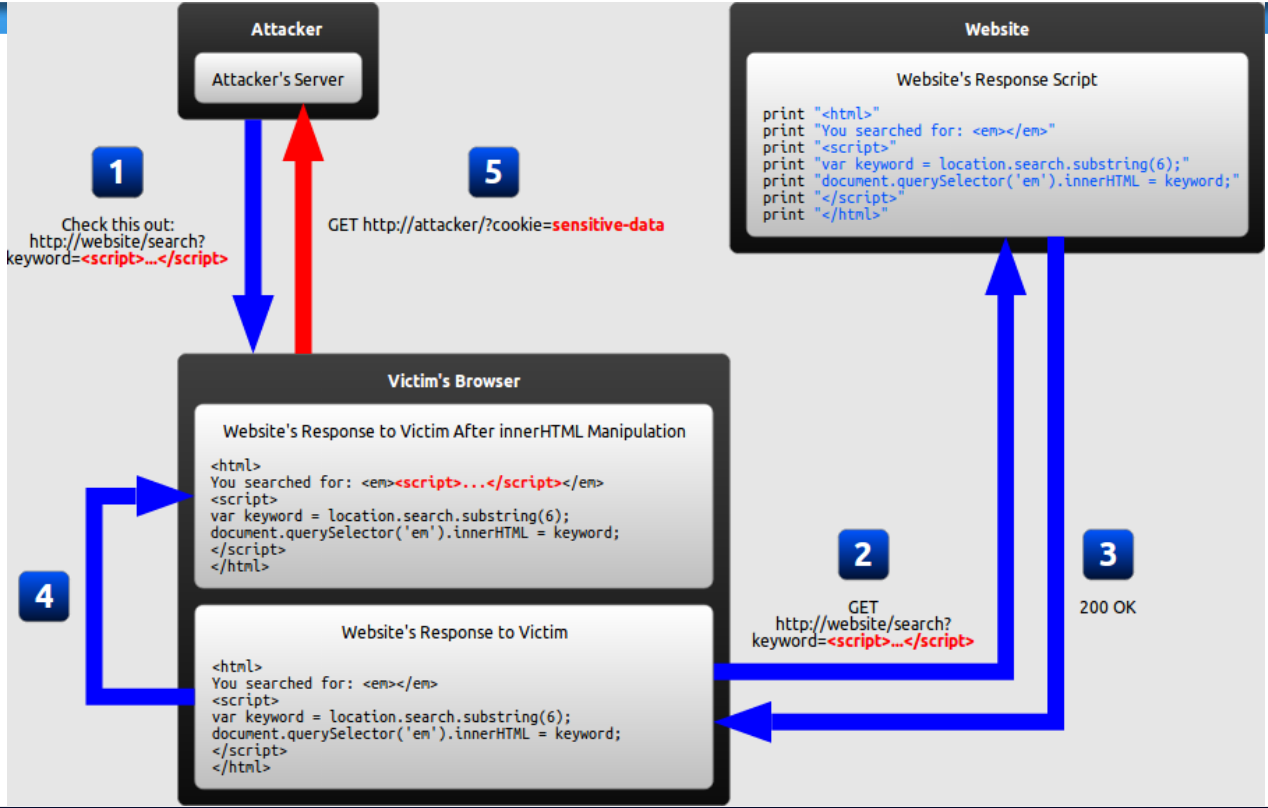
- lokales/DOM-basiertes: In der URL mitgegebene Parameter werden nicht geprüft und lokal mit JavaScript in die Webseite eingebaut.
- reflexives/nicht-persistentes: Durch den Webserver wird ein Parameter in die Webseite eingebaut.
- persistentes/stored: Der XSS-String kann dauerhaft auf dem Server gespeichert werden.

  → Jeder Seitenaufruf der so manipulierten Seite, enthält den XSS-Code.

Übungsblatt 4:
Aufgabe 2: (T) XSS: Reflected XSS



Übungsblatt 4:
Aufgabe 2: (T) XSS: DOM-based XSS



Übungsblatt 4:
Aufgabe 2: (T) XSS

(b) Zum Schutz vor XSS existieren verschiedene Maßnahmen, welche einen Angriff verhindern oder dessen Auswirkungen verringern sollen.
Beschreiben Sie kurz die folgenden Techniken:

- Eingabevalidierung
- Content Security Policy (CSP)
- HTTPonly

Übungsblatt 4:
Aufgabe 2: (T) XSS

Eingabevalidierung / Ausgabevalidierung

- Überprüfen von Eingaben durch Filterregeln, Escaping, Bereinigung, …
  - Nutzereingaben (z.B. Formulare, URL)
  - Daten aus anderen Systemen (z.B. Datenbank, API)
- Validierung von Daten vor der Ausgabe

Übungsblatt 4:
Aufgabe 2: (T) XSS

Content Security Policy (CSP)

- HTTP-Header-Erweiterung, e.g.

  Content-Security-Policy: default-src: 'self'; script-src: 'self' security.lmu.de

- Allowlists von vertrauenswürdigen Quellen, von denen Ressourcen wie CSS, Bilder und JavaScript geladen werden
- Verbot von inline JavaScript

Übungsblatt 4:
Aufgabe 2: (T) XSS

HttpOnly

- Zusätzliches Cookie Flag

  Set-Cookie: <name>=<value>[; <Max-Age>=<age>] [; expires=<date>]
  *[; domain=<domain_name>] [; path=<some_path>][; secure][; **HttpOnly**]*

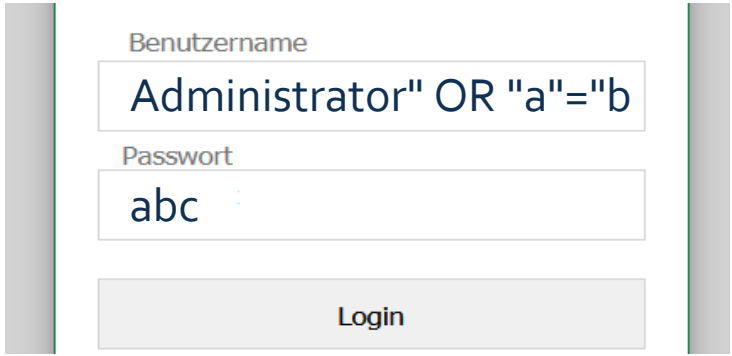
- Muss vom Browser unterstützt werden
- Verhindert den scriptbasierten Zugriff auf das Cookie

Übungsblatt 4:
Aufgabe 2: (T) XSS & SQL-Injection

```
$sql_query = ’SELECT uid FROM users
WHERE username="’ + $username + ’" AND password ="’ + $password + ’"’
```

Abfrage-Query

```
’ PHP-String-Delimiter
+ PHP-String-Concatenator
" SQL-Attribute
```



Ergebnis

```
$sql_query = ’SELECT uid FROM users
WHERE username="’ + administrator" OR "a"="b + ’" AND password ="’ +
abc + ’"’
```

→ `SELECT uid FROM users WHERE username="administrator" OR "a"="b" AND password ="abc"`

Übungsblatt 4:
Aufgabe 3: (T) Security Code Review 101

21.11.2024
WS 2024/2025

Übungsblatt 4:
Aufgabe 3: (T) Security Code Review 101

Online:

- https://owasp.org/SecureCodingDojo/codereview101/

Übungsblatt 4:
Aufgabe 4: (T) CVSS

21.11.2024
WS 2024/2025

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

In einer weit verbreiteten Webanwendung, die in ihrem Unternehmen als Kundenportal zur Verwaltung von Softwarelizenzen verwendet wird und daher öffentlich im Internet zugänglich sein muss, existiert eine cross-site request forgery (CSRF) Schwachstelle. Durch diese Schwachstelle können Angreifer aus der Ferne Aktionen mit den Rechten des angegriffenen Benutzers ausführen, wenn der Benutzer eine aktive Session hat und dazu gebracht werden kann, einen schädlichen Link zu öffnen.

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

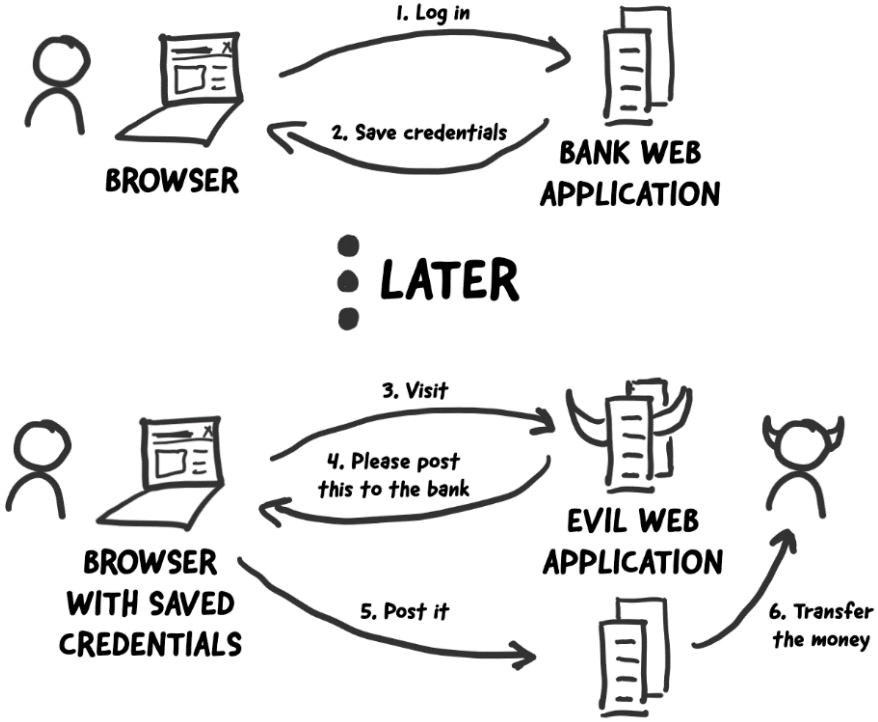
(a) Beschreiben Sie kurz wie ein Angriff per CSRF üblicherweise funktioniert.

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

(a) Beschreiben Sie kurz wie ein Angriff per CSRF üblicherweise funktioniert.

- Voraussetzung: Der Nutzer hat an einem Webserver eine **aktive Session**.
- Der Angreifer bringt einen Nutzer dazu, z.B. einem Link, der eine bestimmte Aktion auslöst (z.B. manipulierter Facebook-Post), zu folgen.
- Der Webserver führt die Aktion durch, da er darauf vertraut, dass der (legitime) Nutzer die Aktion abgesetzt hat.

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3



Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

(b) Berechnen Sie mithilfe des unter https://www.first.org/cvss/calculator/3.1 verfügbaren CVSSv3-Calculators für die beschriebene Schwachstelle den CVSSv3 Base-Score. Vergleichen Sie diesen mit dem über https://nvd.nist.gov/cvss.cfm?calculator&version=2 berechneten CVSSv2 Base-Score.

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

Live Demo

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

(d) Bereits am nächsten Tag tauchte in einschlägigen Foren ein Exploit für diese Schwachstelle auf. Dieser besitzt keine besonderen Voraussetzungen und ist somit in jeder Situation funktional. Wie verändert sich dadurch der CVSSv3 Base-/Temporal-Score aus Aufgabe (c)?

Übungsblatt 4:
Aufgabe 4: (T) Common Vulnerability Scoring System 3

Live Demo

Hausaufgaben

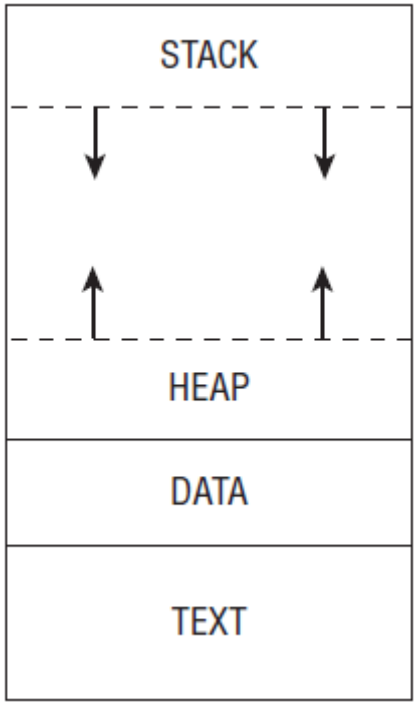
Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

Angreifer nutzen oftmals Schwachstellen in lokal installierten Applikationen.

(a1) Erläutern Sie, was bei einem Buffer-Overflow genau passiert und wie ein Angreifer diesen für einen Angriff ausnutzen könnte?

Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

- Text: Enthält das Prozessabbild
- Data: Enthält globale und statische Variablen
- Heap: Enthält manuell reservierten Speicher (malloc)
- Stack: LIFO Struktur, die die automatischen Variablen enthält.



Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

(a1) Erläutern Sie, was bei einem Buffer-Overflow genau passiert?

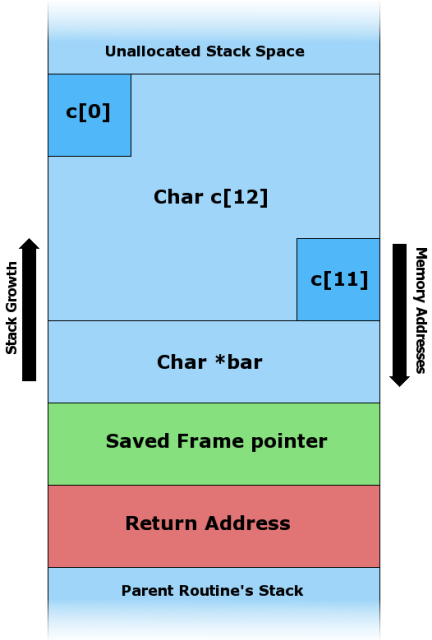
- Buffer: Temporärer Bereich im Speicher
- Buffer Overflow: Schreiben von Daten in Buffer, deren Größe die Buffer-Größe übersteigen.

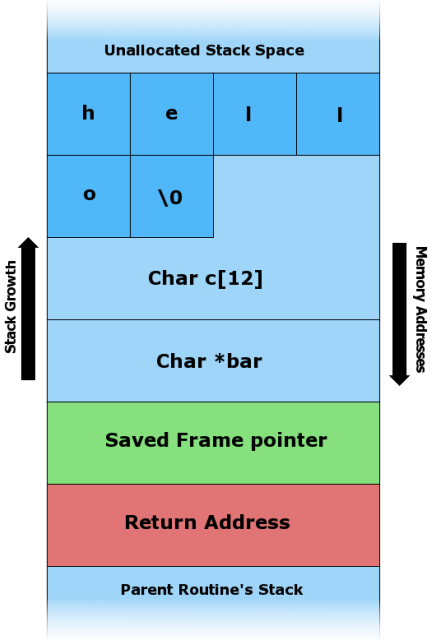
  Wenn Größe der Eingabe nicht geprüft wird (Bound checking), werden nachfolgende Speicher-Bereiche überschrieben.

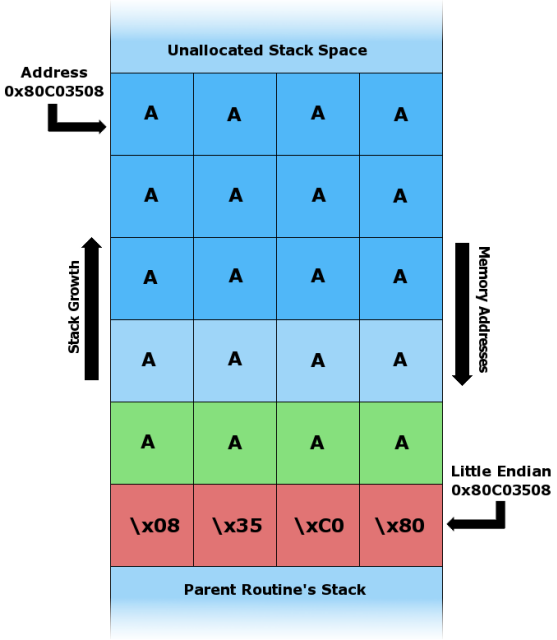
Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

```c
#include <string.h>

void foo (char *bar)
{
   char  c[12];

   strcpy(c, bar);  // no bounds checking
}

int main (int argc, char **argv)
{
   foo(argv[1]);

   return 0;
}
```

Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow







Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

(a2) Erklären Sie, wie ein Angreifer einen Buffer-Overflow für einen Angriff ausnutzen könnte.

+ Absturz des betreffenden Programms
+ Verfälschung von Daten
+ Manipulation der Laufzeitumgebung eines Programms, insb. **Ändern der Rücksprungadresse**

  → Ausführung von beliebigem Code mit Rechten des Programms

Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

(b) Beschreiben Sie den Unterschied zwischen einem klassischen Buffer-Overflow und einem return-to-libc Angriff.

Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

Buffer-Overflow

- Überschreibt Speicherbereiche im Stack
- Kann dadurch:
  - Shellcode platzieren
  - Rücksprungadresse überschreiben
  - Daten manipulieren

return-to-libc

- Überschreibt die Rücksprungadresse so, dass eine Funktion (z.B. system) aus libc aufgerufen wird
- Umgeht damit NX (no-execute) bits, die den Stack als nicht ausführbar markieren

Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

(c) Nennen und beschreiben Sie mindestens drei Schutzmaßnahmen, die zum Schutz vor Buffer Overflows eingesetzt werden können.

- Executable space protection, das NX bit verhindert Ausführung des Stack
- ASLR wählt Adressbereiche für Text, Data, Heap und Stack zufällig → Adressen können nicht mehr deterministisch angesprungen werden

Übungsblatt 4:
Aufgabe 5: (H) Buffer Overflow

(c) Nennen und beschreiben Sie mindestens drei Schutzmaßnahmen, die zum Schutz vor Buffer Overflows eingesetzt werden können.

+ Canaries (Terminator, Random, Random XOR)
+ Bounds-Checking (Größen-Info zur Laufzeit speichern)
+ Testing / Fuzzing