18 Softwaresicherheit

Beispiel: Heartbleed

- Veröffentlicht am 7.4.14: “responsible disclosure”.
- TLS heartbeat: Verbindung offen halten, ohne Daten auf Applikationsschicht zu senden.
  - Definiert in RFC 6520
  - Kann an TLS-Client oder TLS-Server gesendet werden
  - Enthält Nachricht und, separat, die Länge der Nachricht
  - Gegenstelle antwortet mit der selben Nachricht
  - Auch für MTU path discovery, NAT refresh
- OpenSSL Implementierung (Szenario: Client sendet hearbleed an Server):
  - Empfangene Nachricht wird in den Speicher geschrieben
  - Nachricht soll vom Server zurück gesendet werden, dafür wird mit memcpy() die Antwort zusammengebaut; allerdings wird hier die vom Client vorgegebene Länge verwendet!
  - Speicherinhalt wird übertragen, mit ein wenig Glück enthält dieser den geheimen Schlüssel
- Schlimm?
  - Typischerweise keine Spuren am Server
  - Geheimer Schlüssel verloren
  - Zukünftige Kommunikation kann abgehört werden, bis der Schlüssel getauscht wird
  - Vergangene Kommunikation (aufgezeichnet) kann entschlüsselt werden, falls nicht PFS verwendet
  - TLS-Standard nicht betroffen, andere Implementationen von TLS (Microsoft, GnuTLS, ...) nicht betroffen.

- Typisch für Softwaresicherheit: Eingaben werden nicht korrekt geprüft
- Was ist Softwaresicherheit? Software sollte keine Sicherheitslücken haben. Genauer: Vorlesung "Secure Software Engineering" nächstes Semester; wissenschaftlich aber eher schwer greifbar und wenig systematisch
- Problem: Benutzereingaben werden nicht korrekt geprüft (Speicherverwaltung)

Benutzereingaben

UTF-8 Decoding (Microsoft):

Szenario: Benutzer soll eine Datei unterhalb von /home/nils öffnen und bearbeiten können – und nur dort.

- Wir verbieten also "../" im Pfad
- Repräsentation des Pfades im Speicher durch UTF-8

  Exkurs: Unicode

  - ASCII ist ein 7-bit-Code, beispielsweise 010 0100 für $.
  - Unicode ist ASCII-kompartibel
  - Unicode enthält mehr als eine Million Zeichen. Wie?
  - Eine Bitfolge xxx für ein Zeichen wird je nach Platz-Bedarf in einem, zweien oder mehreren Byte kodiert:

    ```
    U000000-U00007F: 0xxxxxxx
    U000080-U0007FF: 110xxxxx 10xxxxxx
    U000800-U00FFFF: 1110xxxx 10xxxxxx 10xxxxxx
    ....
    ```

  - Beispiel: Copyright-Zeichen ©, Unicode Eintrag U00A9 = 1010 1001, mögliche UTF-8 Kodierungen:

```
          11000010 10101001
11100000 10000010 10101001
....
```

  - Gültig ist immer nur die kürzeste Repräsentation

- Microsoft IIS hat ungültige UTF-8 Kodierungen akzeptiert, was die Überprüfung für "../" ausgehebelt hat
- Problem: Benutzereingaben nicht korrekt geprüft
- Abstraktion als Problem

rlogin (GNU):

- Befehl login loggt einen Benutzer ein (Aufruf benötigt root-Rechte)
- Option -f¡benutzer¿ vermeidet Frage nach Benutzername, beispielsweise damit root die Identität eines anderen Benutzers annehmen kann
- rlogin (Vorläufer von SSH) erwartet Benutzername und Maschine zum einloggen: rlogin -l¡user¿ ¡machine¿
- rlogin -l -froot machine startet also login -froot machine
- Problem: Benutzereingaben nicht korrekt geprüft

SQL Injection

- Typisch für PHP-Web-Anwendungen
- Parametrisierte SQL-Anfragen, beispielsweise für Login auf einer Webseite: $sql = "SELECT id FROM users WHERE name = '$name' AND password = password('$password')"
- Setzen Sie als Kennwort ein: foo' or 1=1;–
- Ergebnis: SELECT id FROM users WHERE name = '$name' AND password = password('foo') or 1=1;–')

  - – kommentiert "überschüssige" Zeichen aus

  - Auswertung von WHERE-Ausdruck ergibt true (Links nach rechts auswerten!)
  - Angreifer kann beliebige Benutzer-ID auswählen

- Problem: Benutzereingaben nicht korrekt geprüft

Stack Overrun:

- Stack verwaltet Variablen und Rücksprungadressen des Programms (Skizze)
  - Rücksprungadresse für nach jedem Funktionsaufruf wird hier abgelegt (und vieles mehr)
  - (lokale) Variablen werden hier abgelegt
- Ursache: String wird in den Speicher geschrieben, ohne die Länge zu prüfen (Wurde überhaupt genug Platz für diesen String vorgesehen?) Dieser überschreibt die Rücksprungadresse
- Beispiel:

```
int login(void) {
  char username[10], password[10];

  gets(&username);
  gets(&passsword);

  return check_login(&username, &password);
}

void main(void) {

  if (login(&username, &password)) {
    do_something_cool();
  }

}
```

- Dem Benutzer wird hier durch gets() ermöglicht, beliebig viel Speicher über die 10 + 10 Byte hinaus zu beschreiben. Die Rücksprungadresse im Stack kann so überschrieben werden, dass der Rücksprung direkt auf do_something_cool() erfolgt
- Eine Menge Abwehrmechanismen existieren auf verschiedenen Ebenen: Übung.
- Eine Menge Angriffe auf diese Klasse von Lücken existiert: Übung.

Race Conditions: Starbucks

- Geld von einer Starbucks-Geschenkkarte zu einer anderen zu überweisen erfolgt über ein Zwei-Stufen-System:

  1. Betrag, Absender und Empfänger angeben
  2. Transaktion bestätigen

- Was passiert beim zweiten Schritt?

  1. Betrag der Absenderkarte wird gelesen
  2. Neuer Betrag der Absenderkarte wird geschrieben
  3. Betrag der Empfängerkarte wird gelesen
  4. Neuer Betrag der Empfängerkarte wird geschrieben

- Zweimal die gleiche Transaktion gleichzeitig durchführen kann zu folgender Reihenfolge der Ereignisse führen: A1 B1 A2 A3 A4 B2 B3 B4
- Solche Transaktionen müssen atomar ausgeführt werden
- Erfolg solcher Angriffe hängt vom genauen Timing ab, daher unzuverlässig (viele Versuche)