Analyse von IT Sicherheitsproblemen

Torge Hinrichs

AG Test & Safety

HAW Hamburg

Agenda

1. Einleitung
2. Fallbeispiel: OpenSSL anhand Heartbleed
3. Testverfahren und Ansätze
4. Nächste Schritte
5. Fazit

Einleitung

Einleitung

Schaden durch Cyber-Angriffe allein in Deutschland
100 Milliarden Euro / Jahr [1]
(Stand 2014)

[a]

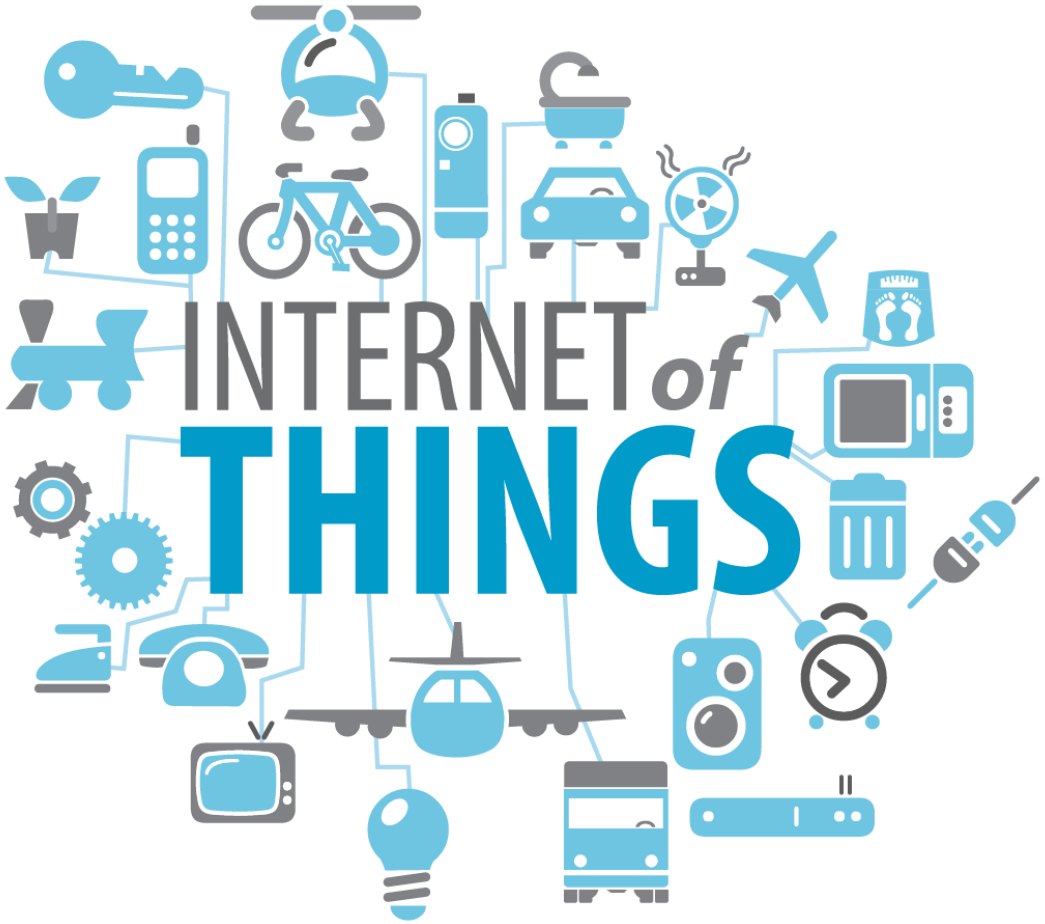
Weltweit:
100 – 500 Milliarden Dollar / Jahr (Stand 2015, McAfee) [2]

Haben wir überhaupt ein Problem?

Ursachen für Schäden

[b]

Ursachen für Schäden



[c]

Ursachen für Schäden

[d]

?

[e]

Ursachen für Schäden

[f]

Einordnen der Problemstellung

HAW HAMBURG

Netzwerk / Protokollbasierte Probleme

- Spoofing
- Denial-of-Service
- Man-in-the-Middle

Anwendungsbasierte Probleme

- SQL-/Injection
- Killscripts
- Read, Add, Delete, Modify – Attacks

[3]

Read, Add, Delete, Modify-Attacken

Beschreibt Manipulation an Software

z.B. Buffer-Overflow:

> Überläufe entstehen in Programmiersprachen, die keine Überwachung von Speichergrenzen mitliefern

In Sprachen wie C/C++ der Fall

Warum dann C/C++?

HAW HAMBURG

Performanz

- C/C++ bietet Kontrolle über Caches und Prozessoren
- Cache Lines

Hardwarenähe

- Direkte Abstraktion zu Assembler

Systemsprache

- Linux Kernel wird in C geschrieben

Hochsprachen

HAW HAMBURG

„Dann schreiben wir den Code eben in Java!“

Die Java Virtuelle Maschine ist in C/C++ geschrieben… [4]

Legacy Code – alte meist schlecht gewartete Software

Sandbox

HAW HAMBURG

„Dann lassen wir den Code in einer Sandbox laufen!“

Sandbox – Isolierte Umgebung, ohne Einfluss auf die Umwelt

Genau das ist die Aufgabe vieler C/C++ Anwendungen

Fallbeispiel: OpenSSL anhand Heartbleed

Warum OpenSSL?

OpenSSL (früher SSLeay):

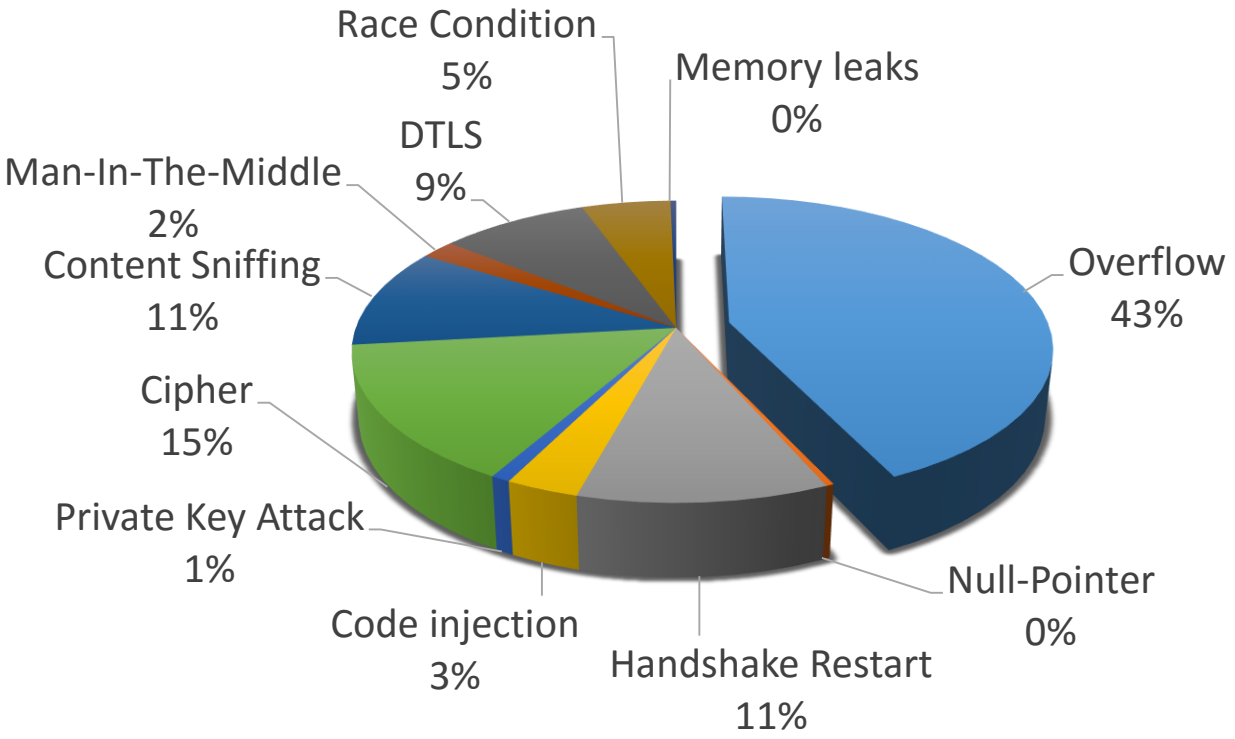
- C –Bibliothek
- Implementierung des Secure Socket Layers (SSL)

1026 Common Vulnerabilities and Exposures (CVE) Nummern analysiert [5]

Ergebnis:

- Als Testling gut geeignet (komplexes Projekt)
- Quellenoffen (Code einsehbar)
- Gut verstandene Funktionsweise

Auswertung OpenSSL CVEs 07.2009 – 03.2015



Secure Socket Layer (SSL)

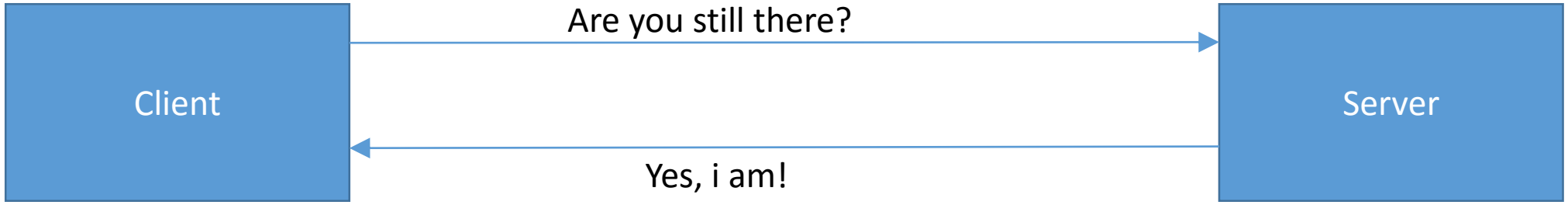
1. Zertifizierung am Server
2. Authentifizierung am Server
3. Validierung des Zertifikats
4.+ Datenaustausch

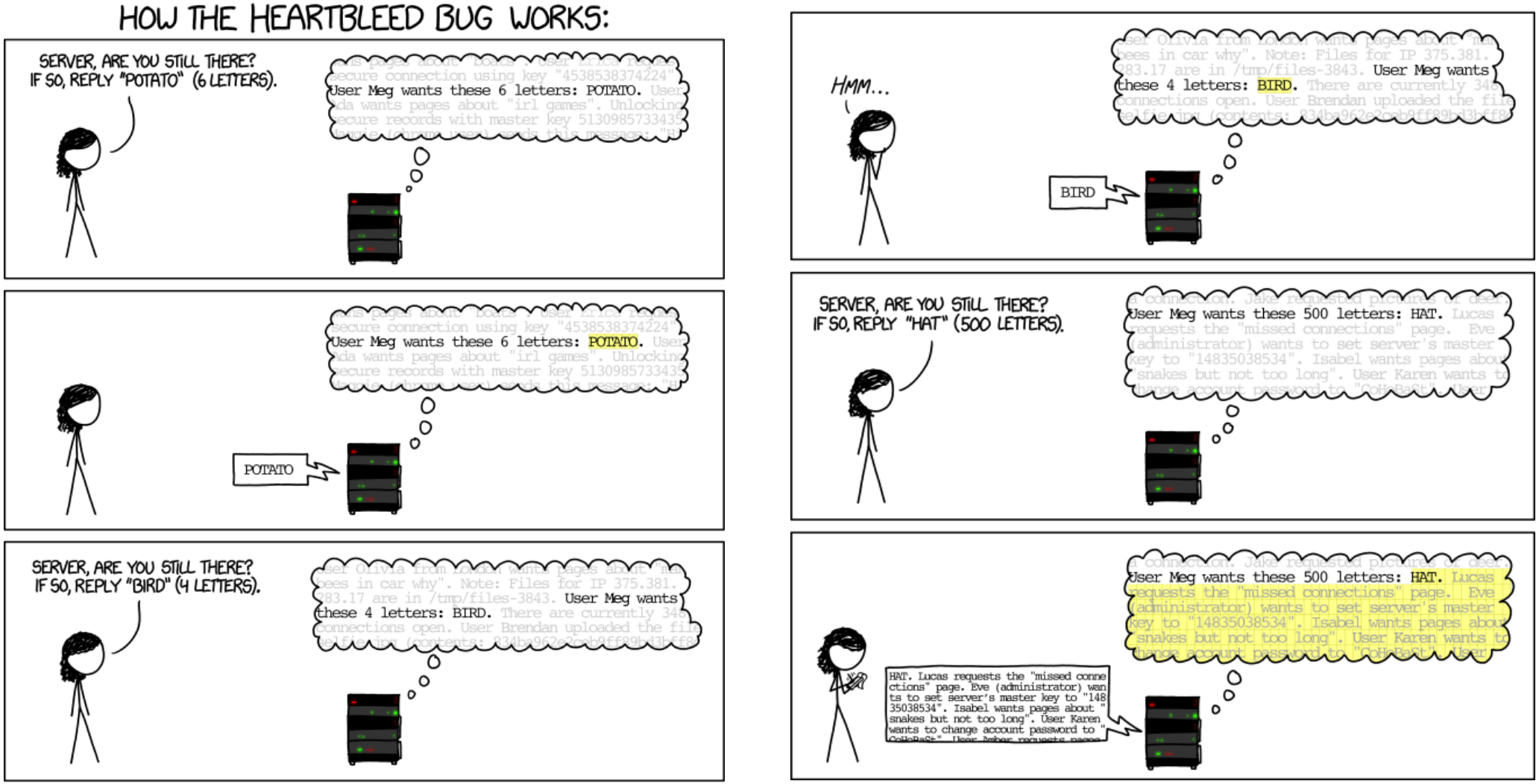
[6]

Feature Request: Heartbeat

Nach Verifikation können Daten ausgetauscht werden

Um Prozedur nach TimeOut nicht zu wiederholen
→ Heartbeats




HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
User Meg wants these 6 letters: POTATO.
POTATO
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
HMM...
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
User Meg wants these 500 letters: HAT.
HAT. Lucas requests the "missed conne ctions" page. Eve (administrator) wan ts to set server's master key to "148 35038534". Isabel wants pages about " snakes but not too long". User Karen wants to change account password to "

[g]

Was bedeutet das für die Praxis?

[e]

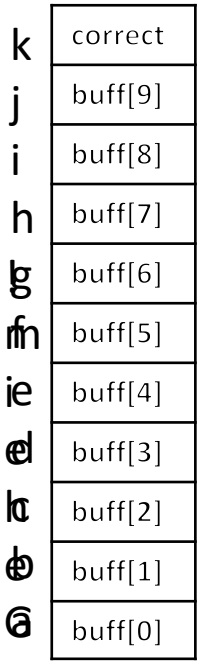
Einstieg: Stack Overflow Attack

```
int main(int argc, char const *argv[]){
        int correct = 0;
        char buff[10];

        printf("Please enter your password\n");
        gets(buff);

        if(strcmp(buff, „Geheim!")){ // 0 wenn gleich!
           printf("Wrong! Try again!\n");
        }else{
           correct = 1;
        }
        if(correct){
           grantSuperUserRights();
           // ... Do fancy stuff
        }

        return 0;
}
```

Input:
Geheim!

Input:
abcdefghijk

Stack:



Heartbleed im Detail

SSL – Heartbeat Message

```
int dtls1_process_heartbeat(SSL *s) {
        unsigned char *p = &s->s3->rrec.data[0], *pl;
        unsigned short hbtype;
        unsigned int payload;
        uint_16 payload_length;
        unsigned int padding = 16; /* Use minimum padding */
…
}
```

[7][8]

Kopieren der übertragenen Länge in einen SSL- Record

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload); // Nimmt 2 Bytes aus p und legt sie in payload
pl = p;
```

Malloc für die geheime Antwortphrase

```
unsigned char *buffer, *bp;
int r;
/* Allocate memory for the response, size is 1 byte
* message type, plus 2 bytes payload length, plus
* payload, plus padding */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
```

Heartbleed zukünftig vorbeugen

Ansatz aus der Wissenschaft:

[Hao et al., 2014]: **Security Management** in allen Ebenen der Entwicklung [10]

Planungsphase: Reviews, Requirements Analyse,…

Nachteile:

Ressourcenintensiv

Nur in großen Unternehmen denkbar

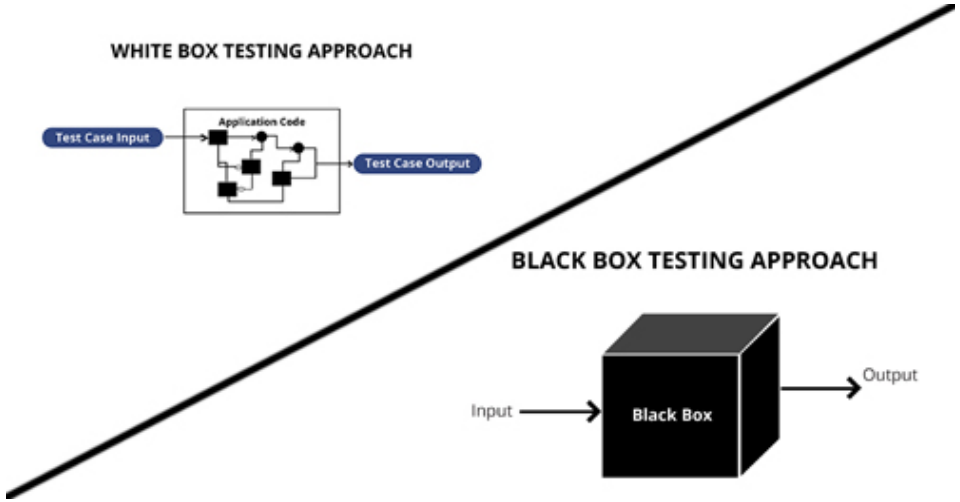
Aber was dann?

Verfahren und Techniken

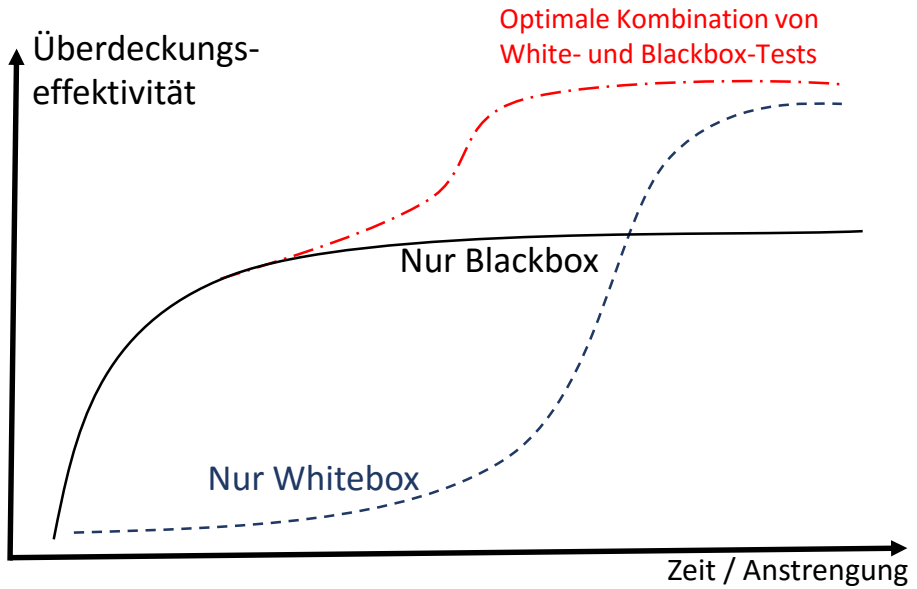
Strukturbasierte Verfahren

Erweitern klassische Blackbox Tests

Verbessern somit den Überdeckungsgrad und die Effektivität der Tests [10]



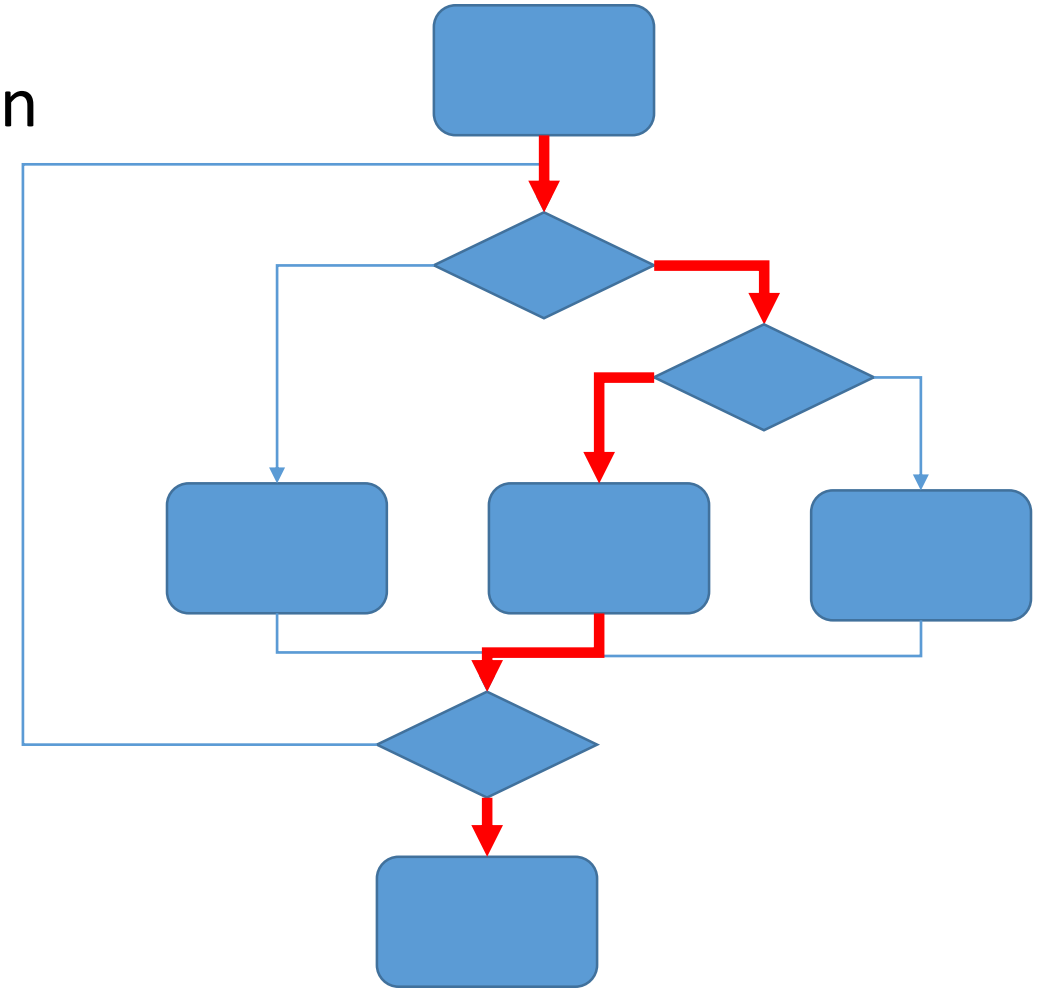
[h]



[i]

Zweigüberdeckung

Prozeduralen Anteil an überdeckten Pfaden bestimmen



[11]

Nachteile von strukturbasierten Verfahren

- Hohe Aufwände
  - Geforderte Code- und Pfadüberdeckunggrade können die Anzahl der Testfälle erheblich steigern
    - Viele Testfälle erforderlich, auch die letzten Prozente zu erreichen
  - Datensensitive Fehler können nicht immer erkannt werden [Copeland 03] [12]
    - x=y/z wird in allen Fällen korrekt sein, außer z = 0
  - Korrektheit der Struktur
    - Fehlende Anforderungen können nicht geprüft werden
  - Reihenfolge des Aufrufs
    - Abhängigkeiten können zu Fehlern führen

Fazit strukturbasierten Verfahren

[Sharma et al., 2013] [13]

Analyse durch klassische Verfahren, wie

Mutation Test, …

Datenfluss, Pfadüberdeckung,…

Für generische Anwendungen zu umständlich → generisches Vorgehen

Fazit:

Generische Algorithmen modifizieren, um klassische Probleme zu erkennen

Analysetechniken(1)

[14]

Statische Analysen / Analysen auf dem Quellcode:

- Strukturelle Eigenschaften verifizieren
- Anomalien aufdecken

[AlBreiki & Mahmoud, 2013] [15]

Framework zum Erkennen von:

- OS Command Injection / SQL Injection
- Crosssite Scripting
- Information Exposure Through an Error Message

Fazit: Framework nutzt Code-Analyse Techniken wie: Bytecode Analyse(Java) → Übertragbar auf C/C++?

Analysetechniken(2)

[14]

Dynamische Analysen / Analysen auf laufendem Quellcode:

- Tote Pointer
- Speicherlecks
  - →Profiling

[Avots et al., 2005] [16]

Dynamische Analyse von Pointern

Zusätzliche Kontrolle für dynamische Bindung von Typen

String Verwundbarkeiten aufdecken

Fazit: Pointer Analyse zur Erkennung von Speicherverletzungen

These Masterarbeit

HAW HAMBURG

„Die Erkennung von Verwundbarkeiten in OpenSSL kann auf andere Software übertragen werden.“

Risiken:

Kann der Fehler in allen Ausprägungen erkannt werden?

Müssen Rahmenbedingungen gelten?

Was bisher geschah…

1026 Common Vulnerabilities and Exposures (CVE) Nummern analysiert

Abstraktion von SSL inklusive Heartbleed

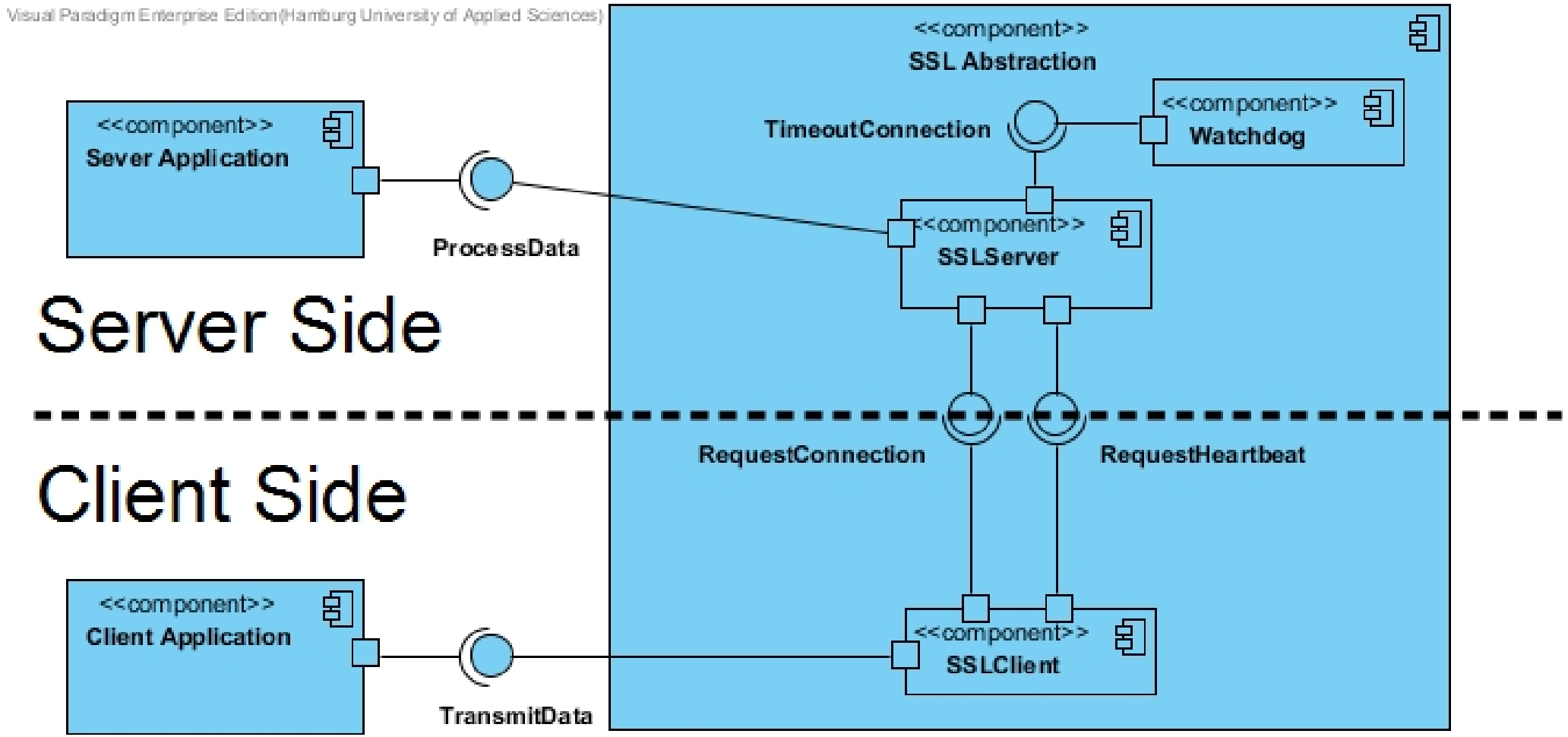
Dummy-Applikation entwickelt

Testplattform / Umgebung mit „Umwelt“ erstellt

Fazit der Vorarbeit:

→Leicht verständliche Probleme in komplexen Zusammenhängen

Die dummy Applikation



Nächste Schritte

Hauptprojekt:

Durchführung und Auswertung der Ansätze

Sind Variationen notwendig?

Sind Tools oder Werkzeuge nutzbar? (Eclipse Plugins,…)

Testplattform erweitern

Kann man die Erkenntnisse auf ähnliche Probleme anwenden?

Fazit

IT- Security Probleme sind i.d.R. einfach zu verstehen

Erkennen in komplexen Zusammenhängen aber schwierig

Tests allein reichen nicht aus

→ Kombinationen müssen ausgewählt und evaluiert werden

HAW HAMBURG

Fragen?

Bildquellen

[a](Stand 20.10.15) https://yasirtariq.files.wordpress.com/2011/09/social_engg.gif

[b](Stand 20.10.15) http://ajqi.com/wp-content/uploads/2013/07/Androidexploit.png

[c](Stand 20.10.15) https://ict.swisscom.ch/wp-content/uploads/2014/12/141225_IoT_FI.png

[d](Stand 20.10.15) http://www.graefe-fitzal.de/ABUS/EC55030_10ES.jpg

[e](Stand 20.10.15) http://www.aurelio-online.com/WebRoot/Store19/Shops/61694983/4C1F/5D6E/EF8F/3798/082B/C0A8/28BC/185D/Schluessel1.jpg

[f](Stand 20.10.15) http://quezi.com/wp-content/uploads/2012/04/software-bugs.jpg

[g](Stand 02.08.15) XKCD explainsheartbleed -https://xkcd.com/1354/

[e](Stand:20.10.15) Twitter Mark Loman - https://twitter.com/markloman

[h](Stand: 20.10.15) b/w Tests - http://www.invensis.net/blog/wp-content/uploads/2015/05/Difference-between-White-Box-Testing-and-Black-Box-Testing-Invensis.jpg

[i](Stand: 20.10.15) Bath& McKay, Test Analyst und Technical Test Analyst (Seite 309), 20. Auflage, 2015, ISBN 978-1-937538-44-6

Fachquellen

[1] FAZ (Stand: 02.08.2015) - http://www.faz.net/aktuell/wirtschaft/wirtschaftsspionage-ingenieursverband-100-milliarden-euro-schaden-12782369.html

[2] McAfee Predictions2015(Stand: 02.08.2015) - http://www.mcafee.com/es/resources/misc/infographic-threats-predictions-2015.pdf

[3] Microsoft Security Threads (Stand: 02.08.2015) - https://technet.microsoft.com/en-us/library/cc959354.aspx

[4] Java VM Spezifikation (Stand: 10.10.2015) - http://docs.oracle.com/javase/specs/jls/se8/html/index.html

[5] Vertraulich zur Verfügung gestellt vom DFN -CERT

[6] Funktionsweise SSL (Stand: 02.08.2015) -http://www.elektronik-kompendium.de/sites/net/0902281.htm

[7] CERT CVE Heartbleed(Stand: 02.08.2015) -https://www.dfn-cert.de/aktuell/OpenSSL-Heartbleed-Schwachstelle-CVE-2014-0160.html

[8] Quellcode OpenSSLGithub(Stand: 02.08.2015) -https://github.com/openssl/openssl

[9] Hao et al., 2014, OpenSSL HeartBleed: Security Management of Implements of Basic Protocols, P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2014 Ninth International Conference on

[10] Bath& McKay, Test Analyst und Technical Test Analyst (Seite 309), 20. Auflage, 2015, ISBN 978-1-937538-44-6

[11] Certified Tester FoundationLevel, Stand 2011

[12] Lee Copeland. A Practitioner‘sGuide toSoftware Test Design, 2003, ISBN 1-58053-791-X

[13] Sharma et al., 2013, A Survey on Software Testing Techniques using Genetic Algorithm

[14] Bath& McKay, Test Analyst und Technical Test Analyst (Seite 279), 20. Auflage, 2015, ISBN 978-1-937538-44-6

[15] AlBreiki & Mahmoud, 2013, Evaluation of Static Analysis Tools for Software Security

[16] Avtos et al., 2005, Improving Software Security with a C Pointer Analysis