



Technische Hochschule
Ingolstadt

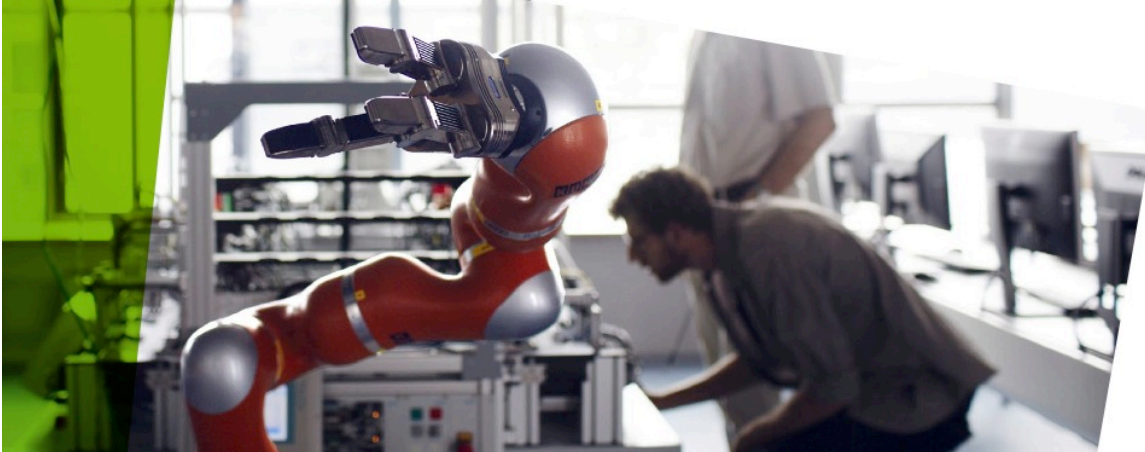
Fakultät für Elektrotechnik
und Informatik

*Zukunft in
Bewegung*

Abschlusspräsentation

Security Work Bench

Projektteam Security Work Bench 13.01.2016





Technische Hochschule
Ingolstadt

Fakultät für Elektrotechnik
und Informatik

*Zukunft in
Bewegung*

Abschlusspräsentation

Wireless Security

Josef König, Christian Paulus 13.01.2016





1. Idee

2. Umsetzung

1. Image
2. Hardware & Aufbau
3. Dokumentation & HTML Anweisungen

3. Die Demos

1. WEP
2. DoS
3. Fake-AP
4. WPA/WPA2
5. WPS

4. Für die Zukunft

Die Idee

- Was hatten wir geplant -

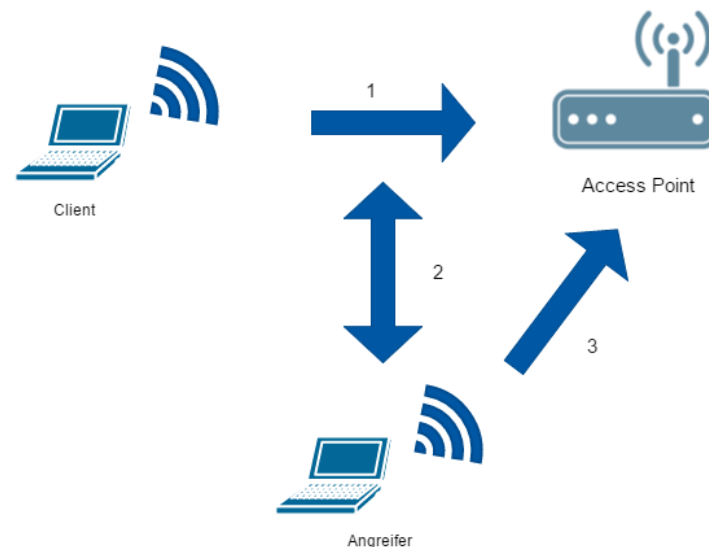


- **Angriffsszenarien als Demos aufbereiten**
- **Dokumentation mit Erklärungen für das Verständnis**
 - Theoretischer Hintergrund und Einleitung
 - Voraussetzungen für die Hacks (Hardware & Software)
 - Durchführung der Angriffe
 - Tools, Befehle und Parameter mit Beschreibung
- **Praktisches Anwenden des Gelernten**



- **Demos in das gemeinsame genutzte Debian Image integriert**
- **Das Image läuft in einer virtualisierten Umgebung**
- **Benötigte Tools sind auf dem Image vorinstalliert**
 - Aircrack Suite
 - crunch
 - Hashcat
 - usw.
- **Abgespeckte Version der Komplettdokumentation als HTML Version für die direkte Arbeit mit dem Image**

- **Welche Hardware ist vorhanden und wird verwendet**
 - 1 Rechner mit dem vorbereiteten Debian Image
 - 1 Alfa Wireless USB Adapter (insgesamt 2 vorhanden)
 - 1 Router mit dem frei verfügbaren OpenWRT (beliebig konfigurier- und erweiterbar)
 - Ggf. einen weiteren Rechner als Client (abhängig von der Demo)





- **Hauptdokumentation als PDF vorliegend**
- **Gemeinsames Dokument mit den anderen Teilgruppen**
- **Gliederung**
 - Szenario
 - Vorbereitungen
 - WEP
 - WPA/WPA2
 - WPS
 - DoS
 - Fake-AP
 - Sicherungsmaßnahmen und Bewertung
- **Anweisungen für die Demos zusätzlich als HTML Version**



- **WEP ist nach heutigem Stand veraltet und gilt als unsicher**
- **Seit 2013 dürfen durch die Wi-Fi Alliance zertifizierte Access Points kein WEP mehr anbieten**
- **Unterstützung verschiedener Authentifizierungsmethoden**
 - Shared Key Authentication
 - Open System Authentication
- **Ansatzmöglichkeiten**
 - Datenverkehr auf dem Access Point
 - Kaum Datenverkehr auf dem Access Point
 - Angriff auf den Client



■ **Ablauf eines Angriffs**

- Identifizieren des anzugreifenden Access Points
- Versetzen des WLAN Interfaces in den Monitoring Mode
- Aufzeichnen des Datenverkehrs zwischen Client und Access Point
- Optional: generieren von zusätzlichem Datenverkehr durch den Angreifer
- „Offline“ errechnen des Schlüssels aus dem aufgezeichneten Datenverkehr



- Bei wenig genutzten Access Points vergeht evtl. viel Zeit bis die benötigte Menge an Datenpaketen aufgezeichnet wurde
- Der Angriff kann je nach Vorgehensweise komplett passiv erfolgen
- Generierung von zusätzlichem Datenverkehr durch den Angreifer (Packet Injection)
 - Open System Authentication
 - Generierung von ARP-Requests durch geeignetes aufgezeichnetes Datenpaket oder aufgezeichnete ARP-Requests



■ Ziele

- Blockieren von Traffic auf dem Ziel-Access Point
- Aufzeichnen eines Handshakes
- Erzwingen schwächerer / keiner Verschlüsselung
- Verbindung auf einen anderen Access Point erzwingen

■ Möglichkeiten für Störungen

- Beacon Flood Mode
- Authentication DoS Mode
- WPA Downgrade test
- Michael shutdown exploitation (TKIP)

■ **Beschränkung auf einen Access Point bei vielen Angriffen möglich**

■ **Angriffe nicht immer erfolgreich durchführbar**



■ **Angriffsmöglichkeiten**

- Ausgabe als Hotspot
- Ersetzen eines bisherigen Access Points in der Umgebung

■ **Ziele**

- Ausspähen von Informationen (Passwörter, Kreditkarten)
- Einschleusen von Schadcode

■ **Ablauf**

- Eventuelles Blockieren eines vorhandenen Access Points
- Erstellen eines eigenen Access Points / Hotspots
- Warten auf sich verbindende Clients
- Verteilung einer Anmeldemaske (Kreditkarten / Passwörter) / Manipulation des Datenverkehrs / Infektion mit Schadcode



- **WPA2 gilt, bei ausreichend langem Key, bisher als sicher**
- **Angriffe auf den PSK von WPA/WPA2 gesicherten Netzwerken**
- **Verfahren für WPA und WPA2 identisch**
- **Ablauf:**
 - Dazu einen Handshake zwischen einem Client und dem AP aufzeichnen
 - Identifizieren des anzugreifenden Access Points und der SSID
 - Versetzen des WLAN Interfaces in den Monitoring Mode
 - Optional: MAC-Spoofing
 - Aufzeichnen eines Handshakes zwischen Client und Access Point
 - „Offline“ Cracken des Keys in der Aufzeichnung



- **Verschiedene Ansatzmöglichkeiten**
 - Bruteforce
 - Dictionary-Attacks
 - Rule-based Attacks
- **Demonstrativer Vergleich der Geschwindigkeiten zwischen Grafikkarte und CPU (nicht auf dem virtuellen Image nachstellbar)**
- **Vergleich verschiedener Tools**
 - z.B. aircrack \leftrightarrow hashcat



- Beim Bruteforcing werden verschiedene Passwörter nach einem bestimmten Erzeugungsmuster durchgetestet
 - Es wird dabei die Länge und die zu testenden Zeichen festgelegt
 - Die Laufzeit kann, abhängig von Länge und Komplexität, von wenigen Minuten bis zu vielen Jahren betragen
-
- Bei einem Dictionary Angriff werden viele Passwortkandidaten aus einer Wörterliste durchgetestet
 - Wörterlisten sind frei im Internet verfügbar und beinhalten häufig verwendete Passwörter



- **Angriff auf den PIN, der für bei WPS für die Verbindung zwischen Client und Access Point verwendet wird**
- **WPS muss dauerhaft auf dem AP aktiviert sein**
- **Angriff dauert zwischen 4 – 6 Stunden**
- **Für eine praktische Vorführung eher ungeeignet**
- **Theoretische Durchführung des Angriffs in der Dokumentation**
 - Angriff kann auch zu Hause, bei Interesse, leicht durchgeführt werden

Für die Zukunft

- was gibt es noch zu tun? -



- **Die Dokumentation ist sicher noch nicht „perfekt“**
- **Ausarbeitung weiterer Angriffsszenarien & Angriffsarten**
- **Es wird in Zukunft weitere Sicherheitslücken im Bereich Wireless Security geben**



Technische Hochschule
Ingolstadt

Fakultät für Elektrotechnik
und Informatik

*Zukunft in
Bewegung*

Abschlusspräsentation

Netzwerk

Julian Rieder, Sebastian Schuster 13.01.16





- **Implementierungen**

- ARP-Spoofing
- DNS-Spoofing
- SSL-Strip
- SYN-Flooding
- Fake-IPv6-Netz

- **Fazit**

- **Live-Demo**



- **Zugang zum angegriffenen Netz**
- **Angriffsrechner mit Ettercap und Wireshark**



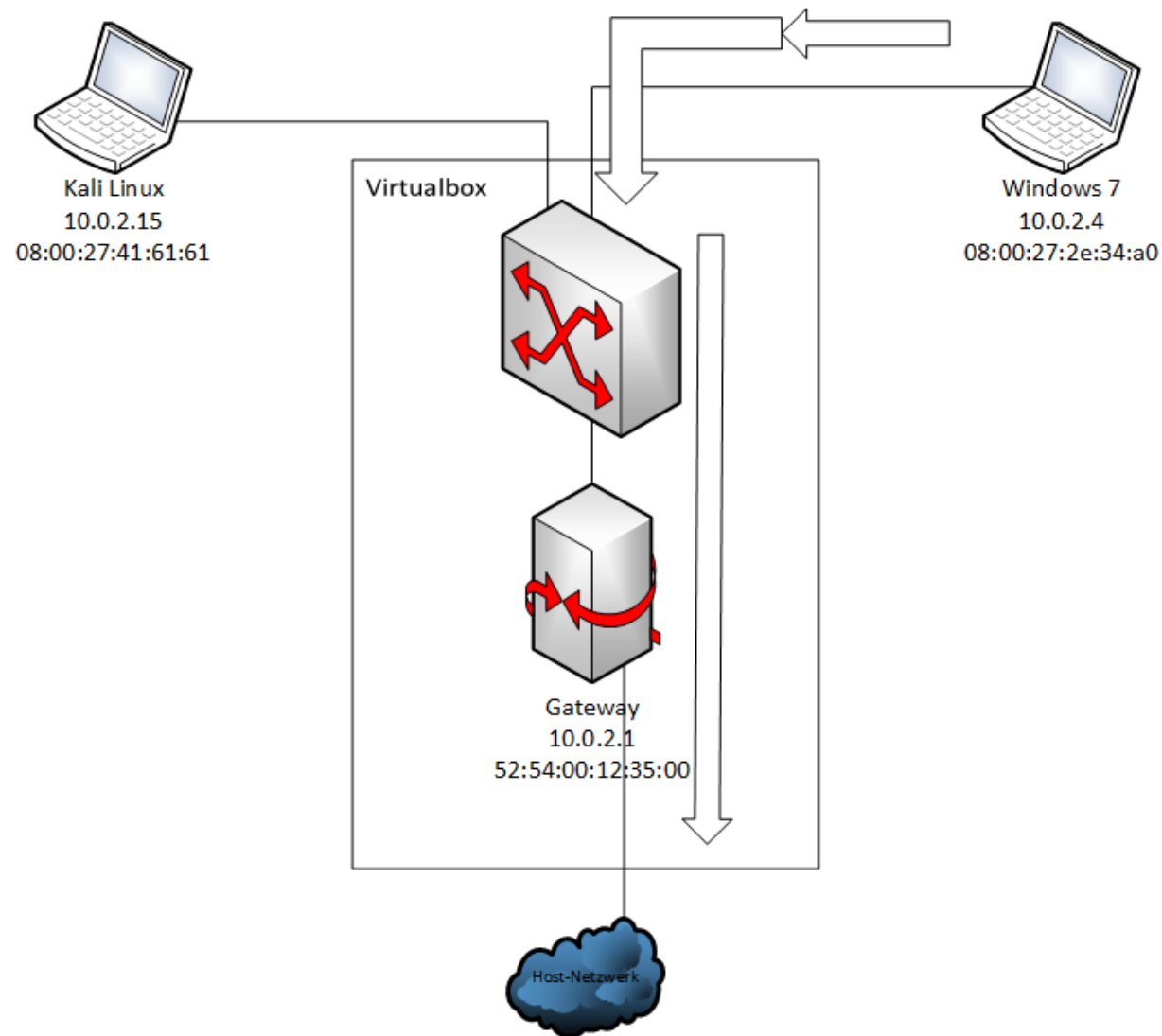
- Man-In-The-Middle Angriff
- Mitlesen und Manipulieren von Netzwerkverkehr
- Eigener Rechner erscheint als Gateway (ARP-Replys)
- ARP Tabelle Opfer:

```
Schnittstelle: 10.0.2.4 --- 0xb
Internetadresse  Physische Adresse  Typ
10.0.2.1        52-54-00-12-35-00   dynamisch
10.0.2.2        52-54-00-12-35-00   dynamisch
10.0.2.15       08-00-27-41-61-61   dynamisch
10.0.2.255      ff-ff-ff-ff-ff-ff   statisch
224.0.0.22      01-00-5e-00-00-16   statisch
224.0.0.252     01-00-5e-00-00-fc   statisch
```

```
Schnittstelle: 10.0.2.4 --- 0xb
Internetadresse  Physische Adresse  Typ
10.0.2.1        08-00-27-41-61-61   dynamisch
10.0.2.2        08-00-27-41-61-61   dynamisch
10.0.2.15       08-00-27-41-61-61   dynamisch
10.0.2.255      ff-ff-ff-ff-ff-ff   statisch
224.0.0.22      01-00-5e-00-00-16   statisch
224.0.0.252     01-00-5e-00-00-fc   statisch
```

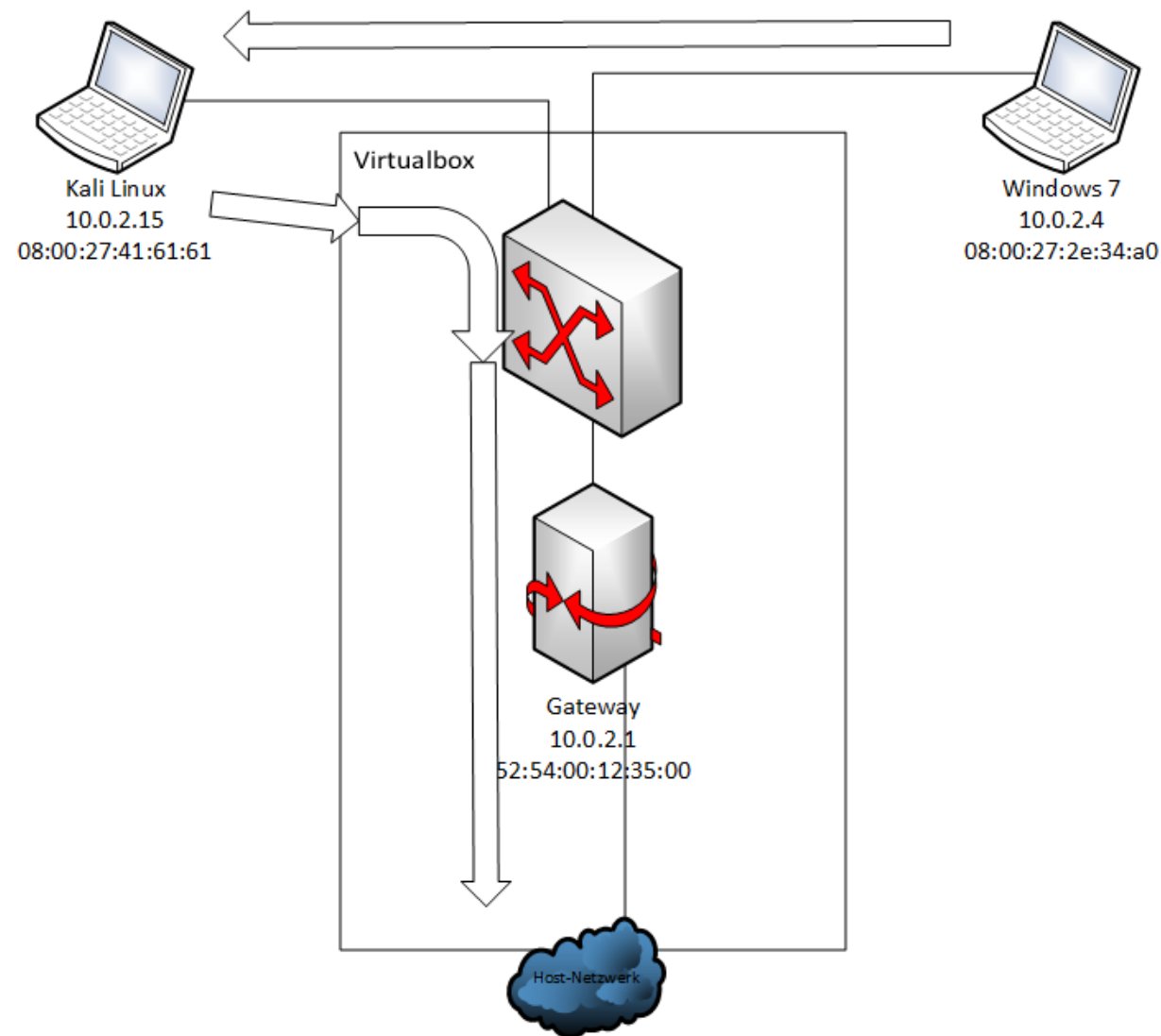
Security-Workbench

ARP-Spoofing



Security-Workbench

ARP-Spoofing





■ Funktionsweise:

- Manipulation eines DNS-Eintrags (Zuordnung Domainname <-> IP-Adresse) auf „falsche“ IP-Adresse, um Datenverkehr unbemerkt mitlesen zu können.
- Durch „gefälschte“ DNS-Responses wird falscher IP-Eintrag an Client übermittelt.
- Bei Aufruf dieser Domain stellt Client Verbindung zu „falscher“ IP-Adresse her.



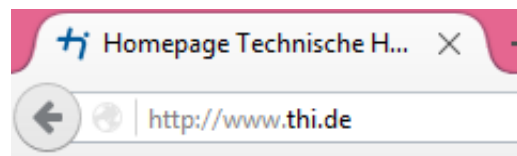
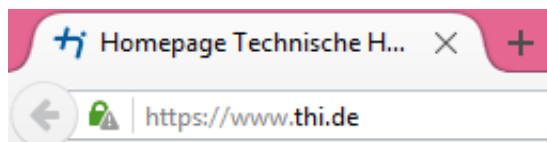
■ DNS-Cache (vorher/nachher)

```
test@test-virtual-machine ~ $ nslookup www.thi.de
Server:      127.0.1.1
Address:     127.0.1.1#53

Non-authoritative answer:
Name:   www.thi.de
Address: 194.94.240.179
```

■ Funktionsweise:

- HTTPS-Verbindungen können ohne (größeren) Aufwand nicht entschlüsselt werden.
- Ziel: Browser / Benutzer dazu bringen, unverschlüsselte HTTP-Verbindung zu verwenden.
- Durchführung: Gros der Benutzer wird Unterschied von https:// und http:// in Browser nicht bemerken.
- Vorgehen: Umwandlung aller https://-Verbindungen in http://-Verbindungen im Quelltext. Anschließend MITM (Client <-http-> MITM <-https->)





■ Funktionsweise:

- TCP-Verbindungen verwenden zum Aufbau einen 3-Way-Handshake (SYN->SYN-ACK->ACK)
- Wenn auf SYN-ACK von Server der (angreifende) Client keine ACK-Antwort sendet, werden für eine bestimmte Zeit Ressourcen auf Server für (angreifenden) Client aufgespart.
- Vielzahl der Anfragen sorgen für Überlastung des Servers und damit Unerreichbarkeit für „normale“ Benutzer.



■ Funktionsweise:

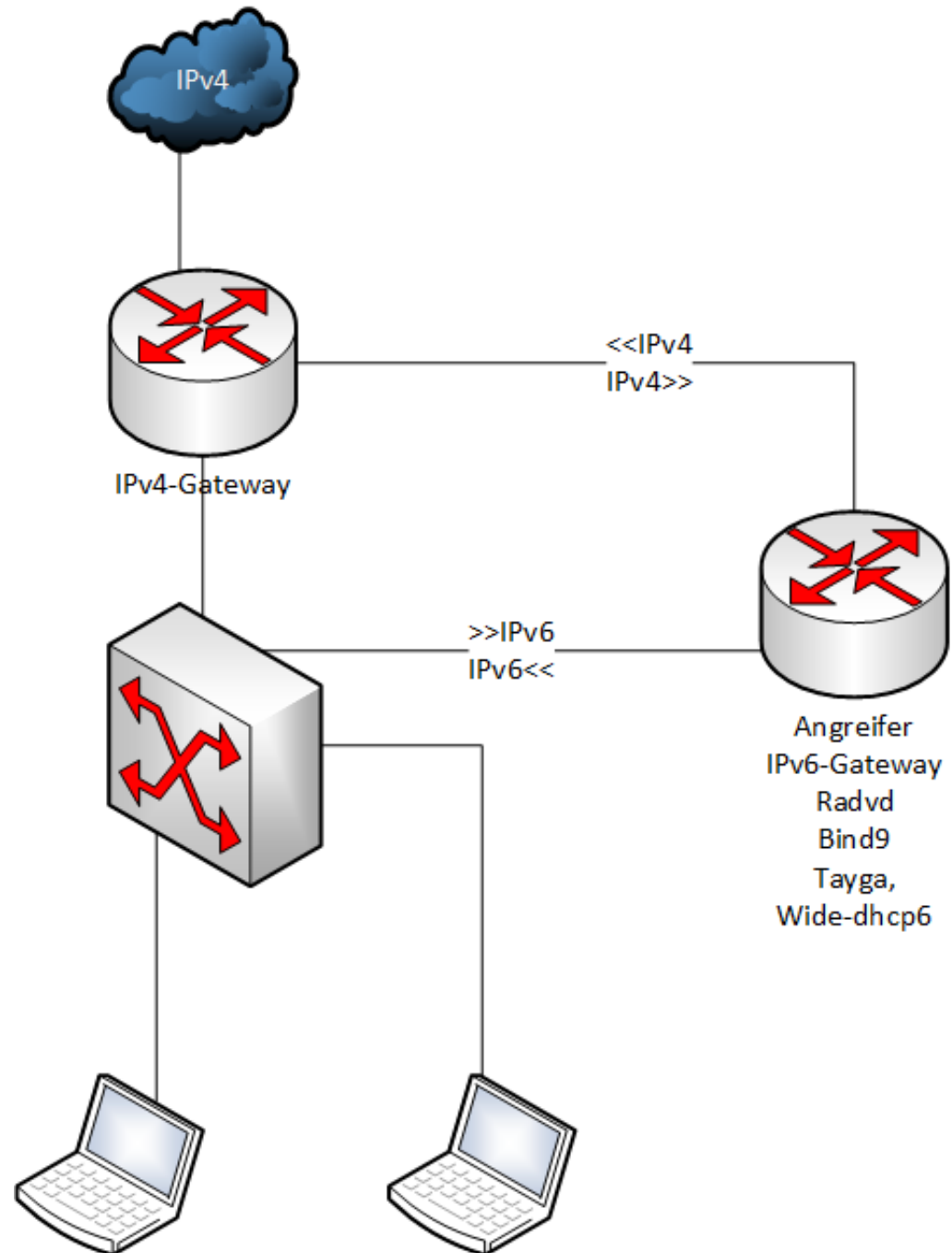
- TCP-Verbindungen verwenden zum Aufbau einen 3-Way-Handshake (SYN->SYN-ACK->ACK)
- Wenn auf SYN-ACK von Server der (angreifende) Client keine ACK-Antwort sendet, werden für eine bestimmte Zeit Ressourcen auf Server für (angreifenden) Client aufgespart.
- Vielzahl der Anfragen sorgen für Überlastung des Servers und damit Unerreichbarkeit für „normale“ Benutzer.



- Zugang zum angegriffenen IPv4 Netz
- Angegriffene Rechner mit unkonfiguriertem, aber aktiviertem IPv6
- Angriffsrechner mit radvd, nat64, bind9, wide-dhcp6 server, Wireshark



- Zugang zum angegriffenen IPv4 Netz
- Angegriffene Rechner mit unkonfiguriertem, aber aktiviertem IPv6
- Angriffsrechner mit radvd, tayga, bind9, wide-dhcp6 server, Wireshark





- **In Summe sechs lauffähige Angriffsszenarien**
- **Angriffe können automatisiert vorgeführt werden**
 - ⇒ Schnelle Demonstration vor Publikum möglich
- **Implementierungen sind nach einem festen Schema aufgebaut**
 - ⇒ Wartbarkeit und Erweiterbarkeit damit gegeben
- **Ausführliche Dokumentation**
 - Funktionsweise des Angriffs wird theoretisch erklärt
 - Benutzung der Tools für den Angriff inkl. Parameter sind dokumentiert
 - Anwendung der (Python-)Implementierungen werden beschrieben



ARP-Spoofing

DNS-Spoofing



Technische Hochschule
Ingolstadt

Fakultät für Elektrotechnik
und Informatik

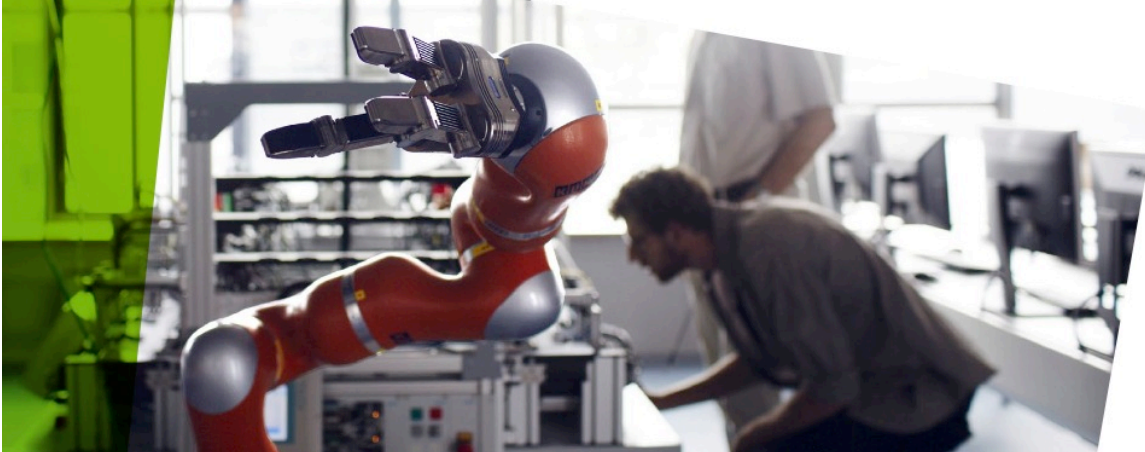
*Zukunft in
Bewegung*

Abschlusspräsentation

International Capture The Flag

**Stefan Zandtner, Maximilian Wenzl, Philipp Weitzl,
Dominik Schlecht, Michael Löckler, Sebastian Beck**

13.01.16





- | | |
|--------------------------------|-------------------------|
| 1. Roxychains & Tor | Dominik Schlecht |
| 2. Buffer Overflow | Maximilian Wenzl |
| 3. Nmap | Michael Löckler |
| 4. Cross Site Scripting | Stefan Zandtner |
| 5. Command Injection | Philipp Weitl |
| 6. Mitmproxy | Sebastian Beck |



- | | |
|--------------------------------|-------------------------|
| 1. Roxychains & Tor | Dominik Schlecht |
| 2. Buffer Overflow | Maximilian Wenzl |
| 3. Nmap | Michael Löckler |
| 4. Cross Site Scripting | Stefan Zandtner |
| 5. Command Injection | Philipp Weitl |
| 6. Mitmproxy | Sebastian Beck |



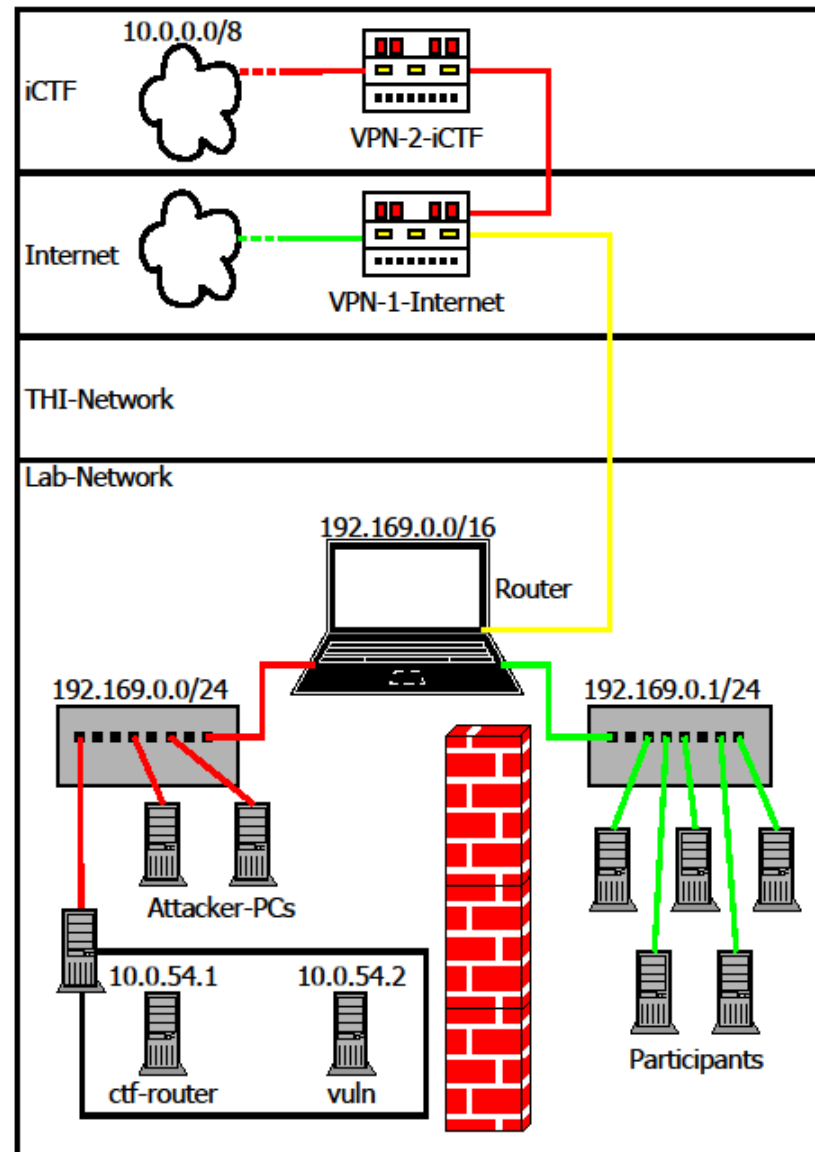
- **iCTF**
 - Aufbau
 - Services - Schwachstellen
- **SecurityWorkBench**
 - Debian
- **Demo: P**
- **roxychains + Tor**

Vorteile


- Sicherheit für Teilnehmer
- Sicherheit für THI

Nachteile:

- Wenig PC's mit Zugriff auf die Vuln-Maschine
- Keine Zentrale Vuln-VM



```
int main(int argc, char *argv[])
{
    srand(time(NULL));
    size_t output;
    char *invalid_characters = decode("O3w+PGAkLQ==", (size_t) 12, &output);
    char str[BUFSIZ];
    char command[100]; // yes I know..
    int ran = rand_lim(100);
    int ciFound = 0;
    size_t input = 108;
```



```
// Read the command
strcpy(str, argv[1]);

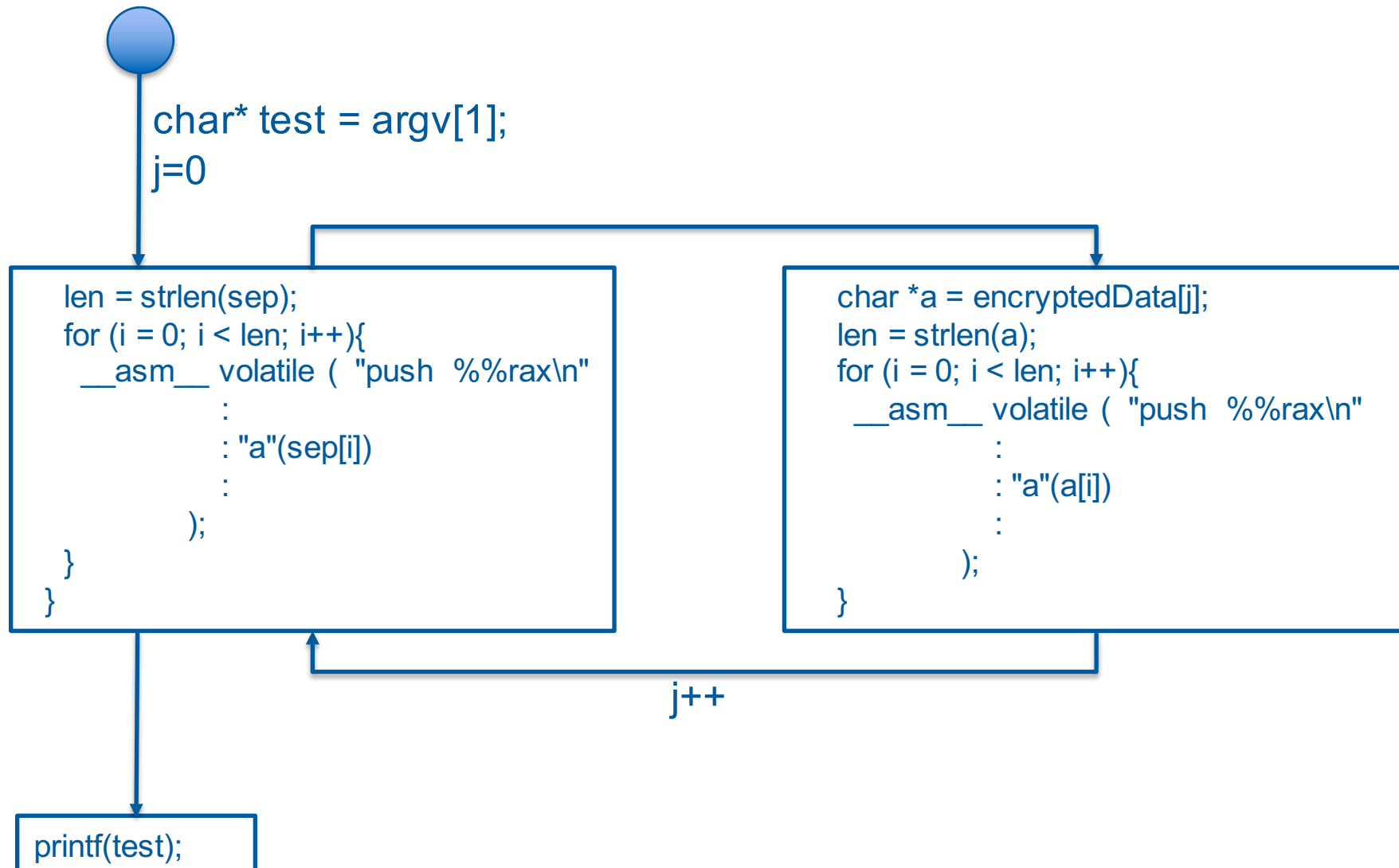
// Check the user input for bad characters

if(test_regex(str) == -1){
    printf("Na\n");
    return -1;
}

char *c = str;
while (*c)
{
    if (strchr(invalid_characters, *c))
    {
        printf("Na\n");
        // Set flag if found
        ciFound = 1;
    }
    c++;
}

sprintf(
    command,
    decode("Y2F0IC4uL3J3L2luZm8vRmFocnpldWdudW1tZXJuLmNzdiB8IGdy
    ZXAqJXMgfCBoZWFKIC0xIHwgYXdrIC1GIcc7JyAne3ByaW50ICQyfSc=",
    (size_t) 108,
    &output),
    str);
char *cmd = command;

if(ciFound == 0){
    FILE *ls = popen(cmd, "r");
    char buf[256];
    while (fgets(buf, sizeof(buf), ls) != 0) {
        printf(buf);
    }
    pclose(ls);
}
return 1;
}
```



■ Inhalte:

■ Website (Flask)

- Anleitungen
- Demos
- Praktikaufgaben

■ Diverse Tools:

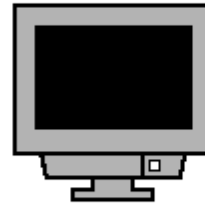
- DNSSPOOF
- ARPSPOOF
- Nmap
- Sslstrip
- ...

Demonstration

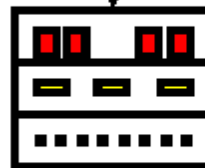
Proxychains + Tor



IP: 31.7.3.42



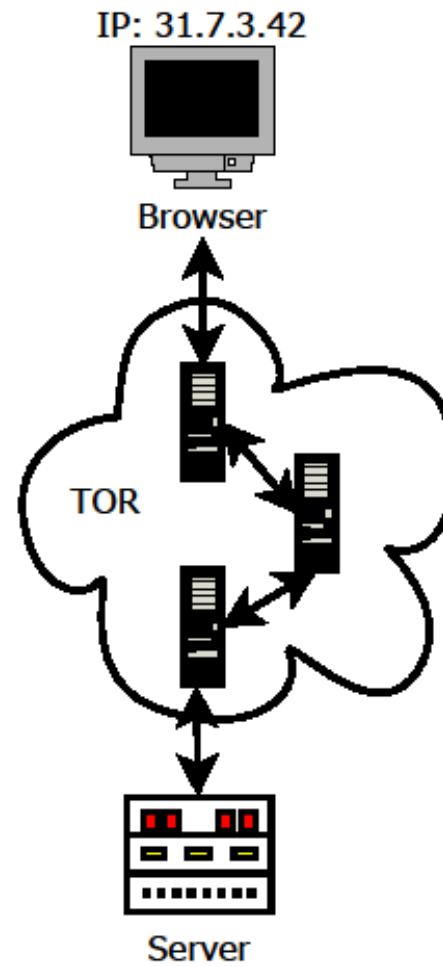
Browser



Server

Demonstration

Proxychains + Tor



Demonstration

Proxychains + Tor



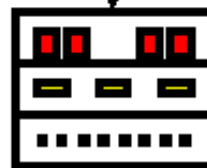
IP: 31.7.3.42



telnet



?



Server

Probleme:

- Kein Proxy-Support!

Lösung:

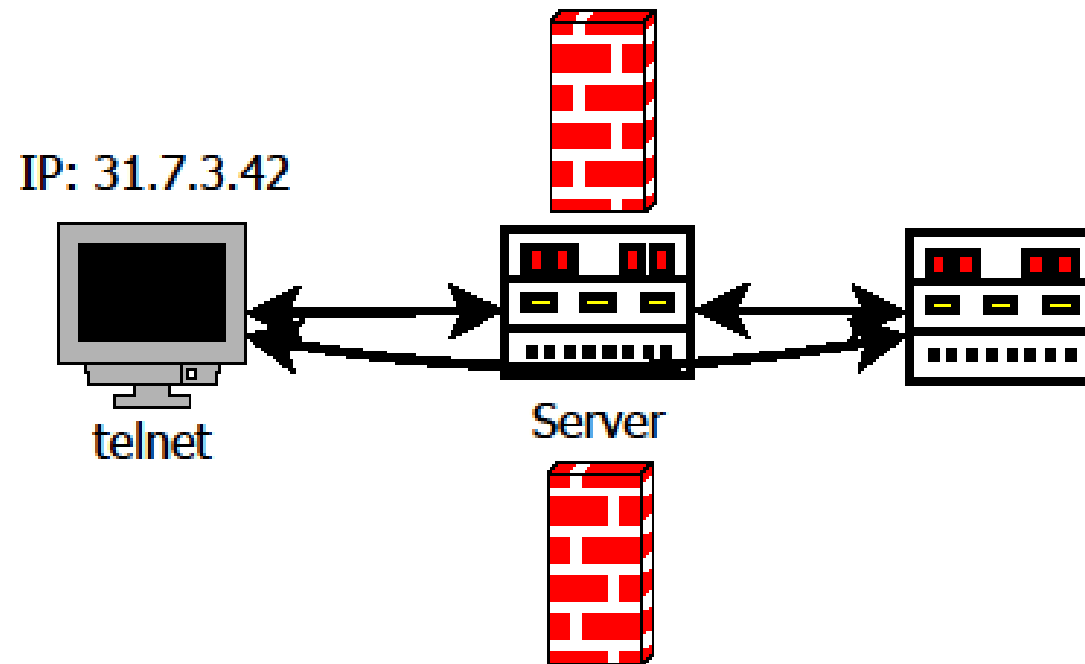
- proxychains-ng



Demo-Time!

Demonstration

Proxychains + Tor





- | | |
|---------------------------|-------------------------|
| 1. Roxychains & Tor | Dominik Schlecht |
| 2. Buffer Overflow | Maximilian Wenzl |
| 3. Nmap | Michael Löckler |
| 4. Cross Site Scripting | Stefan Zandtner |
| 5. Command Injection | Philipp Weitl |
| 6. Mitmproxy | Sebastian Beck |

- **Verschlüsselung der Bayerischen Übersetzungsdatei**
- **OpenSSL AES encryption**
- **Zum Entschlüsseln:**
 - Erstellung einer Textdatei mit Zufallsnamen
 - Herauslesen der benötigten Zeile
 - Löschen der Textdatei
- **Zusammenfassung der Service-Beschreibungen**



- Eine der ältesten Sicherheitslücken die es gibt
- Immer noch Bestandteil der meisten Würmer
- Ein Buffer Overflow kommt zustande, wenn mehr Speicher als vorgesehen benutzt wird
- Beispiel: Buffer wird mit `char a[10]` initialisiert und 15 Zeichen werden eingegeben

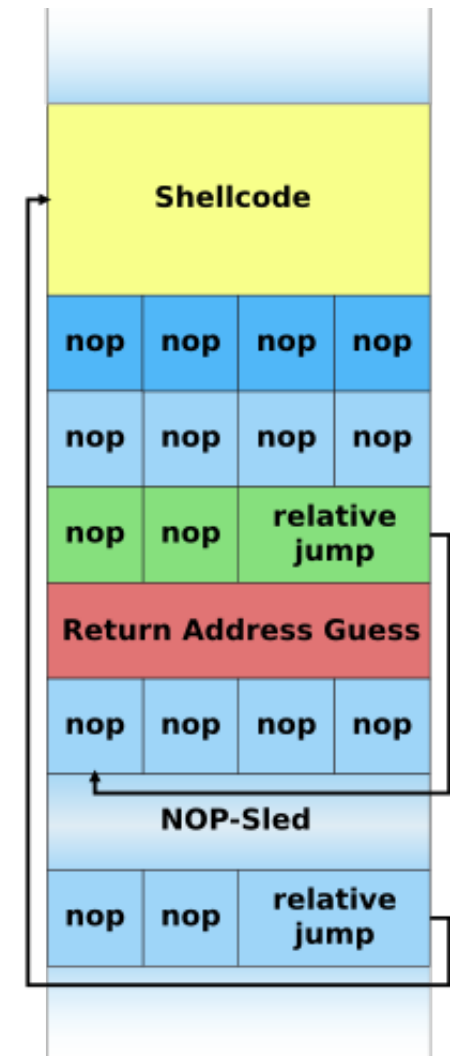


- **Führt meist zu einem Programmabsturz, da Speicherstellen, auf die kein Zugriff besteht, überschrieben werden**
- **Kann jedoch von Hackern ausgenutzt werden, um Sicherheitsschranken zu umgehen**
- **Dazu sind Assembler Kenntnisse nötig**



- Fortgeschritten ist das Überschreiben einer Rücksprungadresse, um in jeden Teil des Programms zu springen
- Dabei muss bekannt sein, wo im Speicher diese Adresse steht
- Der Buffer muss soweit überfüllt werden, dass die Rücksprungadresse überschrieben werden kann.
- Es ist sogar möglich mit dieser Methode eigenen Shellcode zu injizieren und auszuführen

- Eine Shellcodeinjection auf dem Stack sieht in etwa so aus
- Das NOP ist ein Assembler Befehl, der nichts tut
- Es wird genutzt damit die Sprungadresse in jedem Fall erreicht wird





- **ICTF**
 - Marketing
 - Benign
 - Server für Fileshare und Etherpad
 - Unterstützung beim Coding und Anpassung des Service
- **Nmap Tutorial**



- | | |
|-------------------------|------------------------|
| 1. Roxychains & Tor | Dominik Schlecht |
| 2. Buffer Overflow | Maximilian Wenzl |
| 3. Nmap | Michael Löckler |
| 4. Cross Site Scripting | Stefan Zandtner |
| 5. Command Injection | Philipp Weitl |
| 6. Mitmproxy | Sebastian Beck |



- **Plakate**

- je 6 Plakate für die Infoveranstaltung und ICTF
- Zusätzlich Text für die Monitore zusammengestellt

- **Infoveranstaltung**

- Ca. 50 Interessenten an zwei Treffen



- **Service für den Veranstalter zum prüfen des Volkswagenservice**
- **Vom Veranstalter gefordert**
- **Verschleiert Setflag und Getflag aufrufe**
- **Ruft alle Funktionen des Service auf und prüft den Rückgabewert**
- **Wirft einen Fehler, wenn der Service nicht ordnungsgemäß läuft**



■ Server

- Windowslaptop aus dem Netzwerklabor mit Ubuntu als Virtuelle Maschine

■ Etherpad

- Tool zur synchronen Textbearbeitung
- Verwendet um Informationen über zu Hackende Services schnell und effizient zu übermitteln

■ Fileshare

- Network File Server eingerichtet und Wiki zum verbinden erstellt
- Verwendet um größere Dateien, wie die Virtuelle Maschine oder decompilierte C-Files zu verteilen



- **Nmap kann verwendet werden, zum:**
 - Testen von verfügbaren Hosts in einem Netzwerk
 - Testen von offenen Ports eines Hosts
 - Herausfinden des Betriebssystems und der Version eines Hosts
 - Herausfinden der Art von Paketfiltern/-Firewalls

Vorführung



1. Roxychains & Tor Dominik Schlecht
2. Buffer Overflow Maximilian Wenzl
3. Nmap Michael Löckler
- 4. Cross Site Scripting Stefan Zandtner**
5. Command Injection Philipp Weitl
6. Mitmproxy Sebastian Beck

XSS – Cross Site Scripting

Übersicht

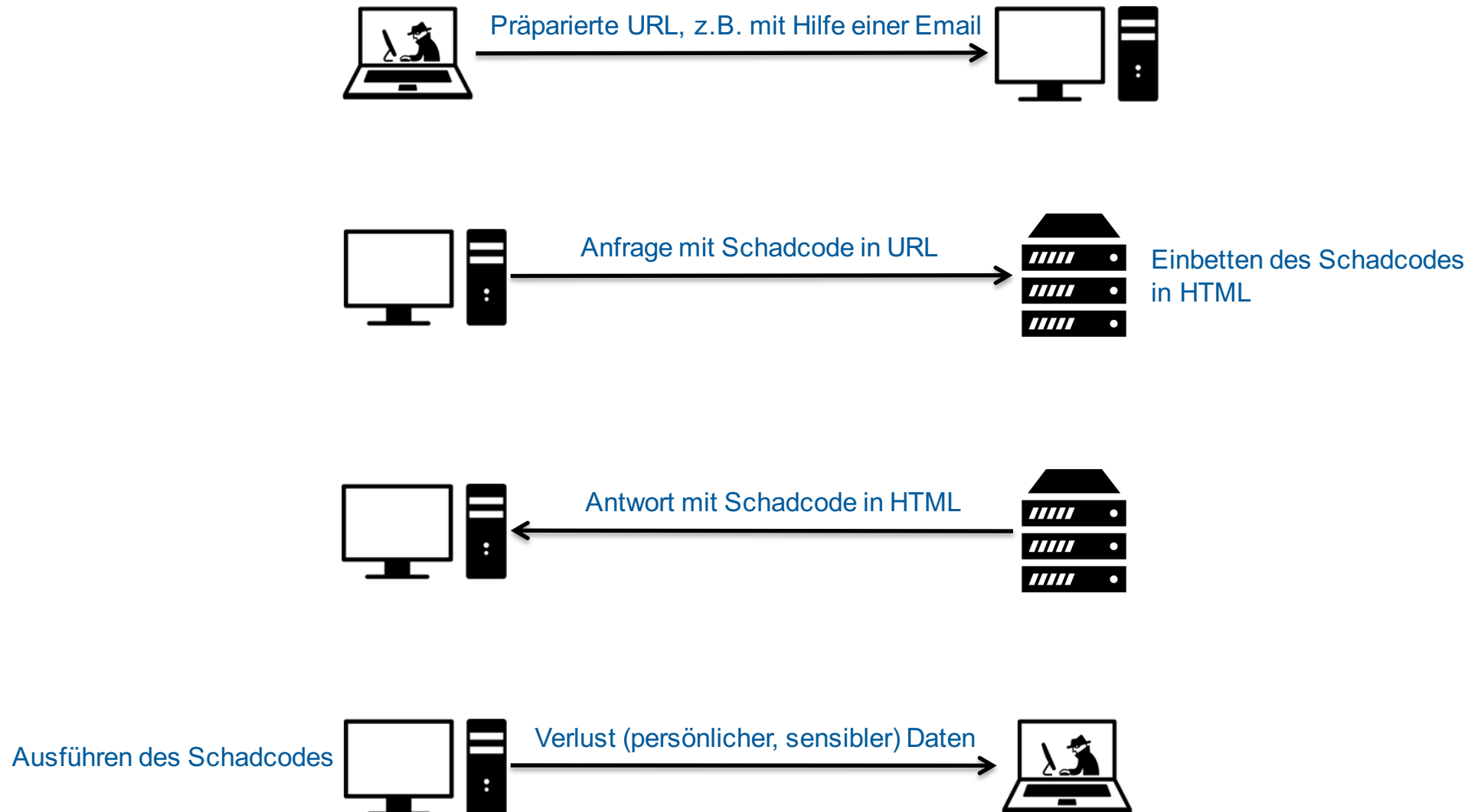


- Sicherheitslücke in Webanwendungen
- Nicht vertrauenswürdige Inhalte werden als vertrauenswürdig eingestuft
- Ziel: Erbeutung (sensibler) Daten des Nutzers

- Arten:
 - Reflected XSS
 - Stored XSS
 - Local XSS

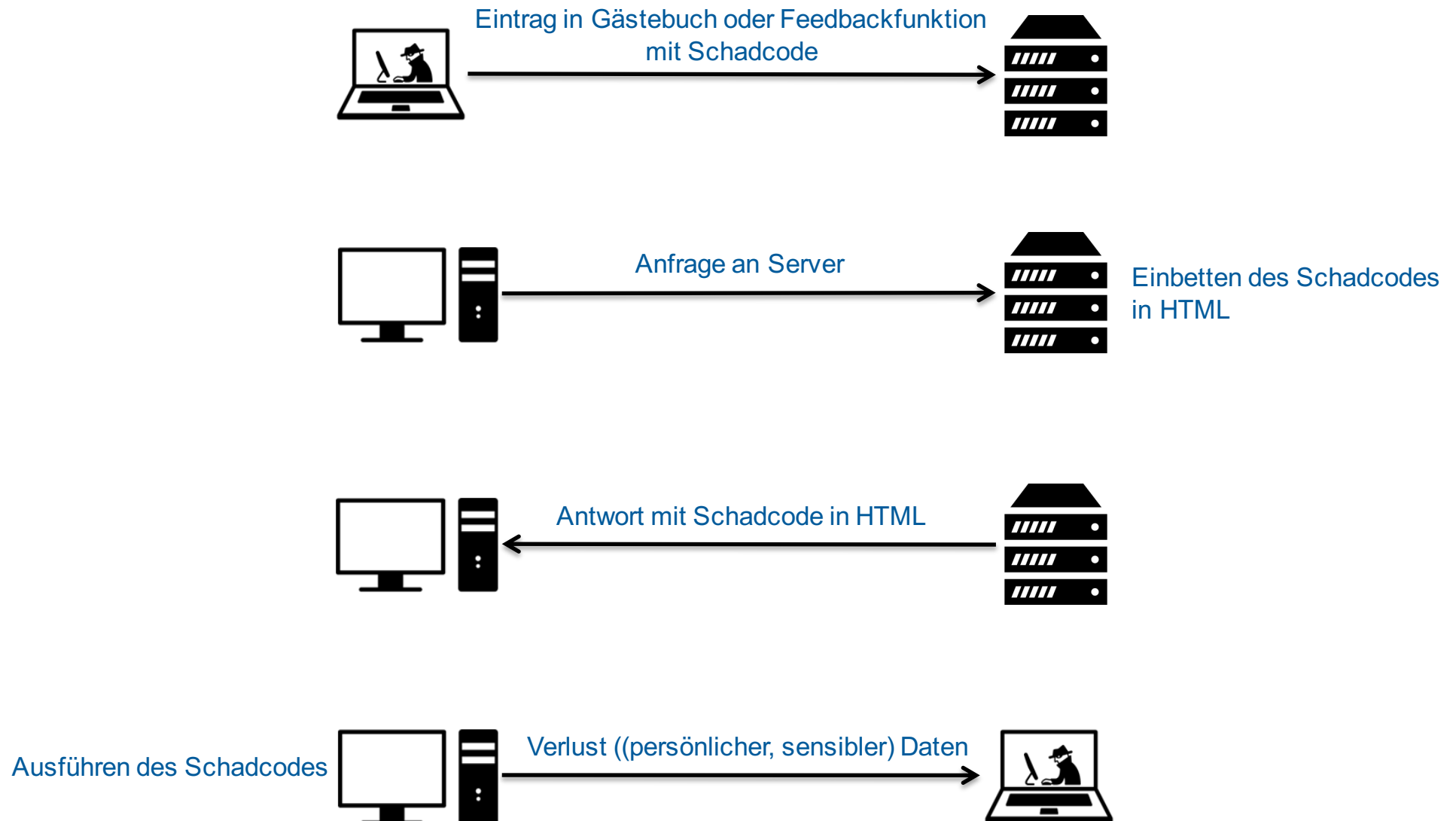
XSS – Cross Site Scripting

Reflected XSS (non-persistent)



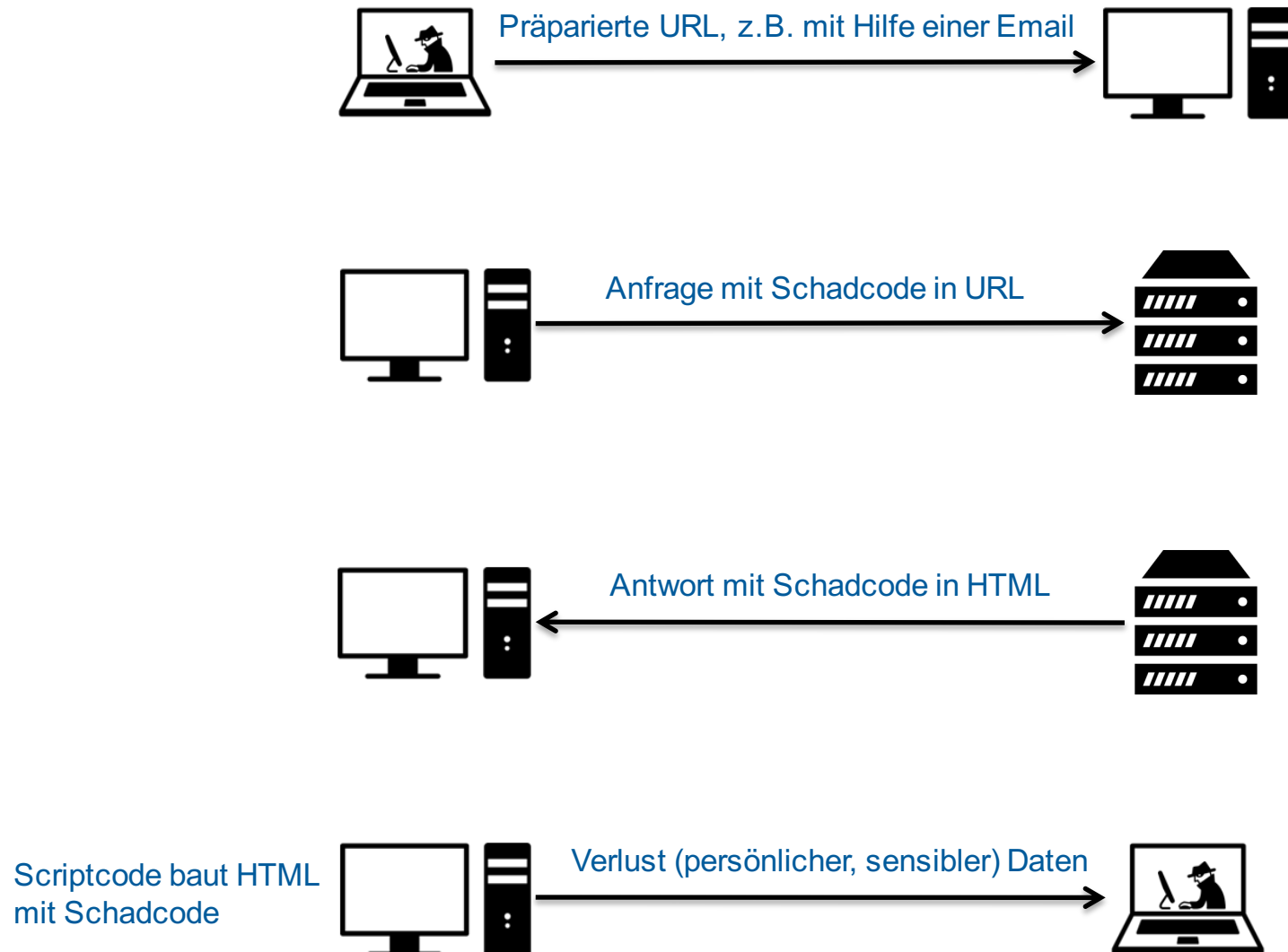
XSS – Cross Site Scripting

Stored XSS (persistent)



XSS – Cross Site Scripting

Local XSS (Dom Injection)





- | | |
|-----------------------------|----------------------|
| 1. Roxychains & Tor | Dominik Schlecht |
| 2. Buffer Overflow | Maximilian Wenzl |
| 3. Nmap | Michael Löckler |
| 4. Cross Site Scripting | Stefan Zandtner |
| 5. Command Injection | Philipp Weitl |
| 6. Mitmproxy | Sebastian Beck |

- **Dokumentation und Wiki**
- **Teile des Python-Server für die Annahme der Verbindungen**
- **Fahrgestellnummer**
 - Aufbau der Nummer
 - Erstellung der Informationstabellen
 - Überprüfung mittels Regex
- **Übersetzer**
 - Übersetzungstabelle bayrisch < > deutsch



Zugriff durch:

- **Unsichere Benutzer Schnittstellen**

Gründe:

- **Unsichere**

Ausführung:

- **Anhängen von Commands**



■ php Codebeispiel

```
<?php  
    echo shell_exec( 'cat' . $_GET[ 'command' ] );  
?>
```

■ Zugriff über

- |
- ||
- ;
- &&



DEMO

- **Angriff durch Blind Command Injection**

- **Senden einer Mail**
- **Pingen eines Servers**
- **Schreiben eines Files**



- **Typische Orte für Schwachstellen**

- **Laden von Bildern**
- **Externe Bibliotheken / Non-native Code (Perl)**
- **Direkte Codeausführung (php)**



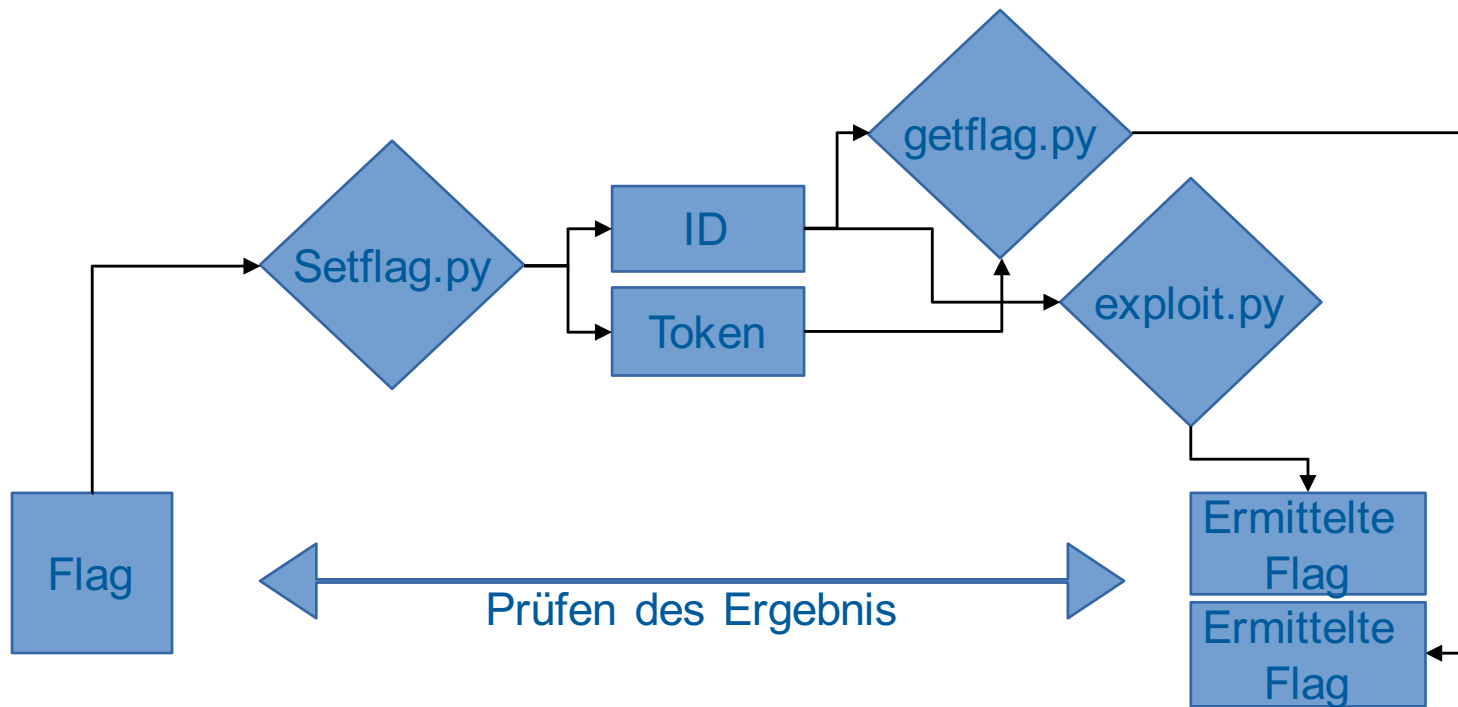
■ Vorkehrungen

- Vermeidung von Parameterübergabe an Betriebssystem
- Filterung der Zugriff-Kommandos (z.B.: &&)
 - Whitelist für Zugriffe
- Strikte Input-Filter
 - Externe und eigene Funktionen
- Limitierung der Eingabe
- Verwendung sichererer Programmiersprachen (Java)



- | | |
|-------------------------|-----------------------|
| 1. Roxychains & Tor | Dominik Schlecht |
| 2. Buffer Overflow | Maximilian Wenzl |
| 3. Nmap | Michael Löckler |
| 4. Cross Site Scripting | Stefan Zandtner |
| 5. Command Injection | Philipp Weitl |
| 6. Mitmproxy | Sebastian Beck |

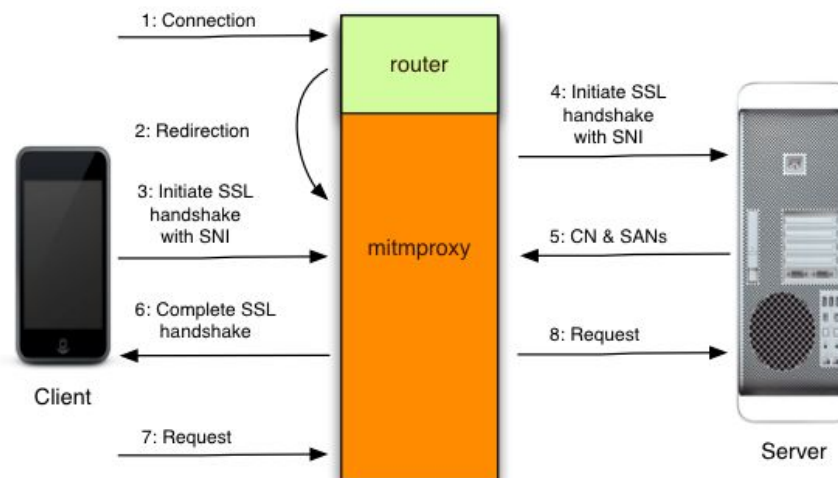
- Ziel: Automatisiertes Testen der Scripte
- setflag.py → getflag.py / exploit.py





- **Alternative Tools** die ein solches Feature (neben anderen Zusatzfunktionen) anbieten: Burp Suite, SSLSplit, usw.
- **Grundsätzliche Verfahren sicher**
- **Zertifikatsinfrastruktur hat aber Schwächen**
- **Beispielszenarien:**
 - **Zertifikatsstelle/Zertifikatsvergabe kompromittiert**
 - **Besitz des Server-Key wurde erlangt**
 - **Zugriff auf den Opfer-PC → lokal Vertrauen gegenüber neuen Zertifikaten aussprechen**
 - **Analyse des Https-Datenverkehr fremder Anwendungen**

■ Arbeitsweise von mitmproxy:



<http://mitmproxy.org/doc/howmitmproxy.html>



Live-Demo