

OWASP Top 10 Schulung

Agenda

- Übersicht
- A1:2017 – Injection
- A2:2017 – Broken Authentication
- A3:2017 – Sensitive Data Exposure
- A4:2017 – XML External Entities (XXE)
- A5:2017 – Broken Access Control
- A6:2017 – Security Misconfiguration
- A7:2017 – Cross-Site Scripting (XSS)
- A8:2017 – Insecure Deserialization (Community)
- A9:2017 – Using Components with Known Vulnerabilities
- A10:2017 – Insufficient Logging & Monitoring (Community)
- What's Next for Developers?
- Übungen an WebGoat

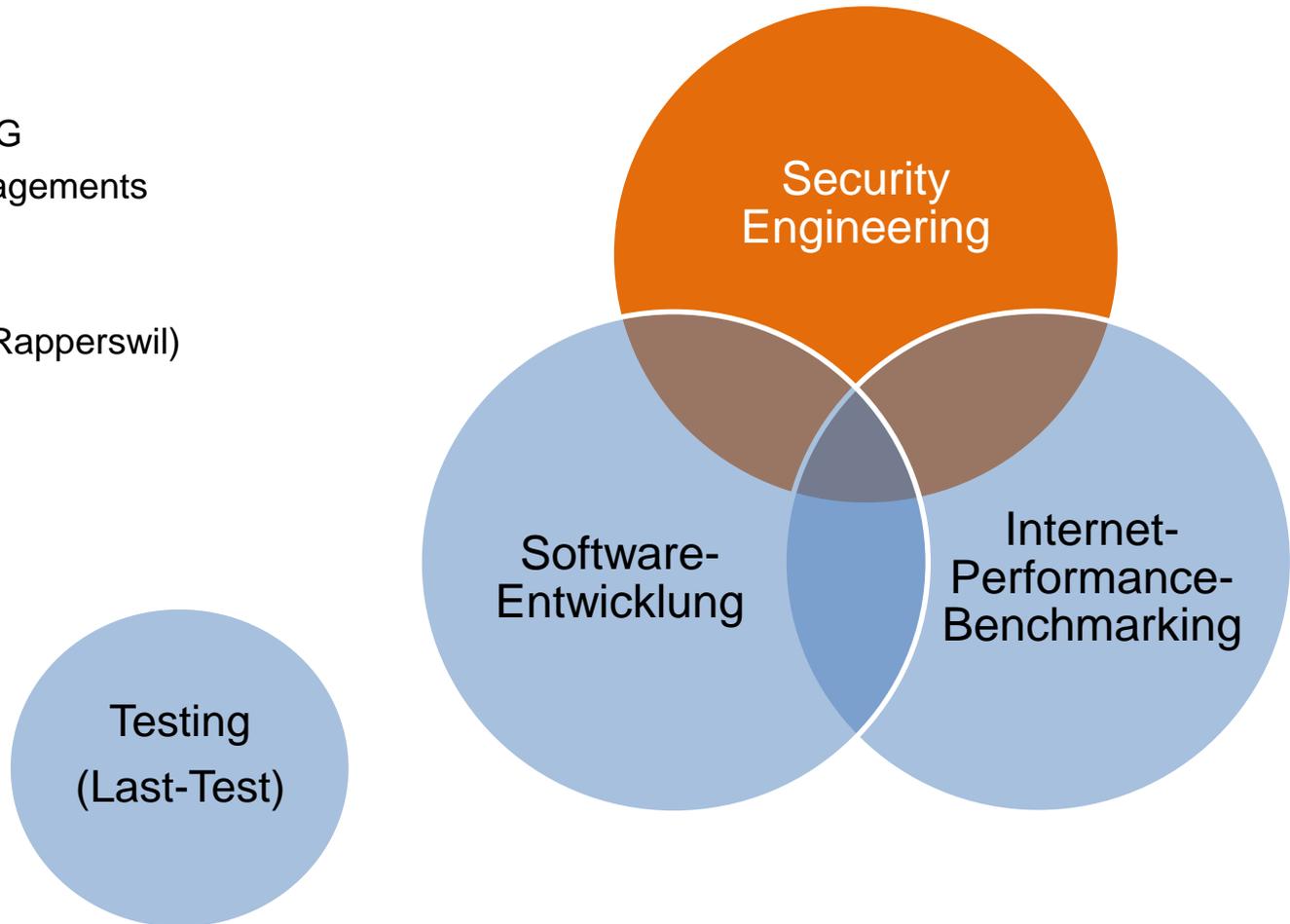
Cnlab Gruppe

cnlab Organisation

- 15 Ingenieure
- Sitz Rapperswil / SG
- Im Besitz des Managements

cnlab Partner

- HSR (Hochschule Rapperswil)
- ETH Zürich
- CSI Consulting



Aufbau

- Worum geht's?
- Wie finden?
- Was dagegen tun?

| Threat Agents | Exploitability | Weakness Prevalence | Weakness Detectability | Technical Impacts | Business Impacts |
|------------------------------|----------------|---------------------|------------------------|-------------------|----------------------|
| Appli- cation Specific | Easy: 3 | Widespread: 3 | Easy: 3 | Severe: 3 | Business Specific |
| | Average: 2 | Common: 2 | Average: 2 | Moderate: 2 | |
| | Difficult: 1 | Uncommon: 1 | Difficult: 1 | Minor: 1 | |

Begriffe ¹

- OWASP Top 10 Web Application Security Risks
- OWASP: Open Web Application Security Project
- Schwachstelle hinter dem Risiko wird hier betrachtet.

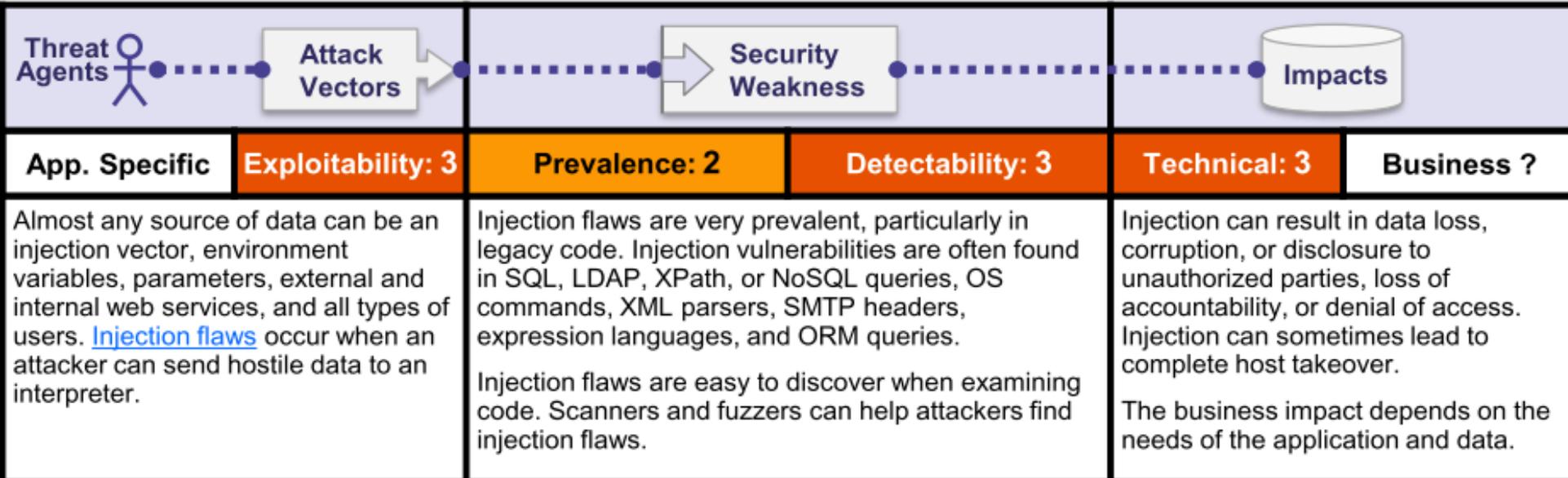
| Deutsch | Englisch | Berechnung |
|--------------------|----------------|---|
| Bedrohung | threat | |
| Gefahr, Gefährdung | applied threat | $F(\text{threat}, \text{weakness})$ |
| Schwachstelle | weakness | |
| Risiko | risk | $\text{Probability} * \text{Intensity}$ |

1. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html

Entwicklung der OWASP Top 10

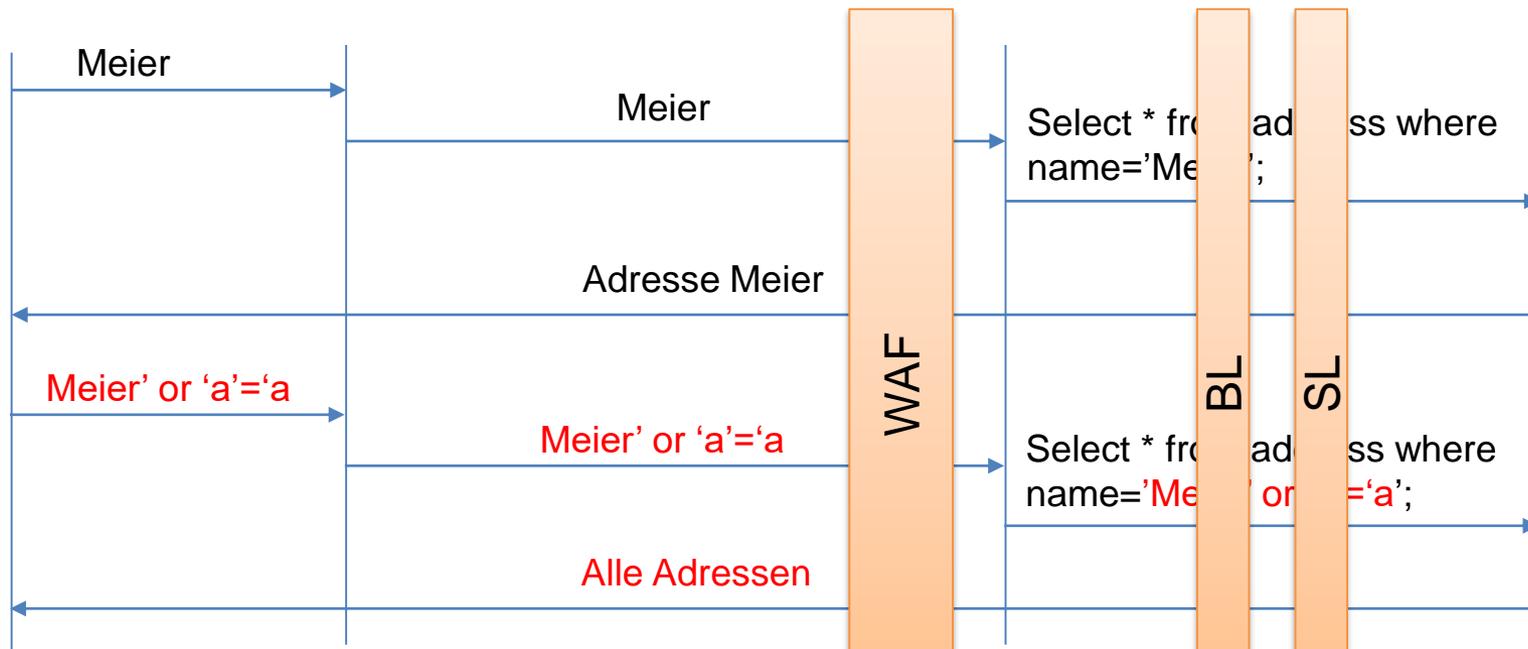
| OWASP Top Ten - 2010 | OWASP Top Ten - 2013 | OWASP Top Ten - 2017 |
|---|---|---|
| A1 – Injection Flaws | A1 – Injection Flaws | A1 – Injection Flaws |
| A2 - Cross Site Scripting (XSS) | A2 - Broken Authentication and Session Management | A2 - Broken Authentication and Session Management |
| A3 - Broken Authentication and Session Management | A3 - Cross Site Scripting (XSS) | A3 – Sensitive Data Exposure |
| A4 - Insecure Direct Object Reference | A4 - Insecure Direct Object Reference (fließt in 2017 A5 ein) | A4 – XML External Entities (XXE) |
| A5 - Cross Site Request Forgery (CSRF) | A5 - Security Misconfiguration | A5 – Broken Access Control |
| A6 - Security Misconfiguration | A6 - Sensitive Data Exposure | A6 – Security Misconfiguration |
| A7 - Insecure Cryptographic Storage (fließt in 2013 A6 ein) | A7 - Missing Function Level Access Control (fließt in 2017 A5 ein) | A7 – Cross-Site Scripting (XSS) |
| A8 - Failure to Restrict URL Access | A8 - Cross Site Request Forgery (CSRF) | A8 – Insecure Deserialization |
| A9 - Insufficient Transport Layer Protection (fließt in 2013 A6 ein) | A9 - Using Known Vulnerable Components | A9 - Using Known Vulnerable Components |
| A10 - Unvalidated Redirects and Forwards (Fließt in 2013 A6 ein) | A10 - Unvalidated Redirects and Forwards | A10 – Insufficient Logging & Monitoring |

A1:2017 – Injection - **Worum geht's?**



Ein guter WAF findet die Standard-Patterns

A1:2017 – Injection - **Worum geht's?**



A1:2017 – Injection - **Wie finden?**

- Code-Review auf sicheren Gebrauch von Interpretern
 - Aufruf von Interpretern muss zwischen Eingabedaten und Befehlen unterscheiden
- Scanner können hilfreich sein.
 - SAST (Static Application Security Testing, Source-Code-Analyse)
 - DAST (Dynamic Application Security Testing)

A1:2017 – Injection - Was dagegen tun?

- Nutzung einer sicheren API
- Rechte der Anwendung limitieren
- Validierung des User-Input auf dem Server per «Whitelist»
- Metazeichen entschärfen
- Stored Procedures verwenden
- Keine detaillierten Fehlermeldungen

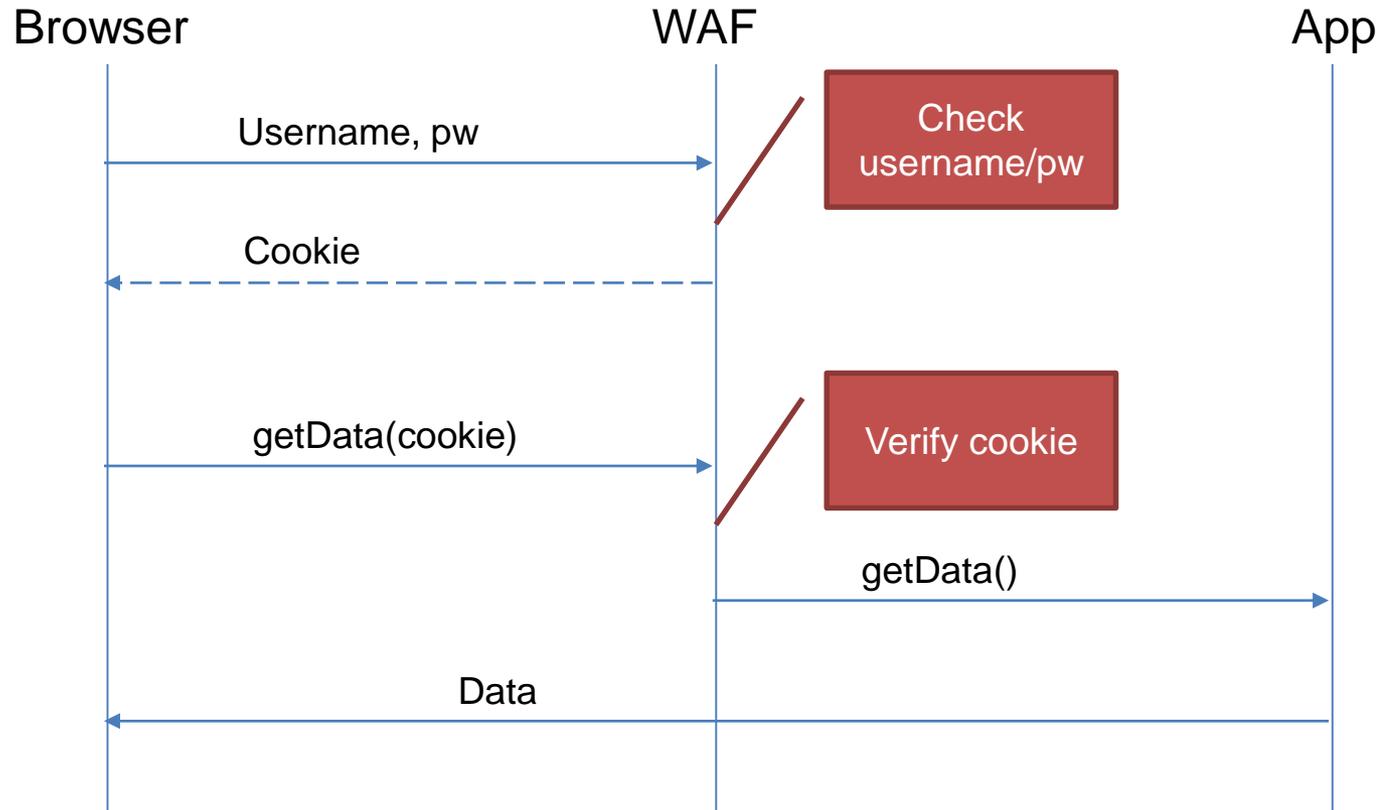
A2:2017 – Broken Authentication and Session Management - Worum geht's?

| | | | | | |
|---|--------------------------|---|-------------------------|--|-------------------|
| | | | | | |
| App. Specific | Exploitability: 3 | Prevalence: 2 | Detectability: 2 | Technical: 3 | Business ? |
| <p>Attackers have access to hundreds of millions of valid username and password combinations for credential stuffing, default administrative account lists, automated brute force, and dictionary attack tools. Session management attacks are well understood, particularly in relation to unexpired session tokens.</p> | | <p>The prevalence of broken authentication is widespread due to the design and implementation of most identity and access controls. Session management is the bedrock of authentication and access controls, and is present in all stateful applications.</p> <p>Attackers can detect broken authentication using manual means and exploit them using automated tools with password lists and dictionary attacks.</p> | | <p>Attackers have to gain access to only a few accounts, or just one admin account to compromise the system. Depending on the domain of the application, this may allow money laundering, social security fraud, and identity theft, or disclose legally protected highly sensitive information.</p> | |

- Authentisierung und Session Management sind zentral für die Sicherheit von Web-Applikationen.

A2:2017 – Broken Authentication and Session Management

Worum geht's?



A2:2017 – Broken Authentication and Session Management - **Wie finden?**

- Wie wird der Benutzer authentisiert?
- Wie werden die Benutzer-Passwörter gespeichert?
- Wie wird die Benutzererkennung erstellt bzw. verwaltet?
- Wie werden Passwörter, Session IDs und Benutzername übertragen?
- Handhabung von Session IDs?
 - Sichtbar in der URL?
 - Session-Fixation möglich?
 - Sessions laufen nicht ab / Token werden beim Logout nicht invalidiert

A2:2017 – Broken Authentication - Was dagegen tun?

- Standard-Mechanismen für die Authentisierung verwenden
- Multi-Faktor-Authentisierung verwenden
- Default-Zugänge entfernen / Passwort ändern
- Starke Passwort-Policy (z.B. NIST Richtlinien¹) / Schwache Passwörter ausschliessen
- Passwörter nur verschlüsselt übertragen
- Sichere Passwort-Wiederherstellung
- Hardening von Registrierung, Passwort-Wiederherstellung, API-Zugriffen
- Detektion von und Alarmierung bei Angriffen (z.B. Brute Force)
- Gescheiterte Login-Versuche limitieren oder verzögern

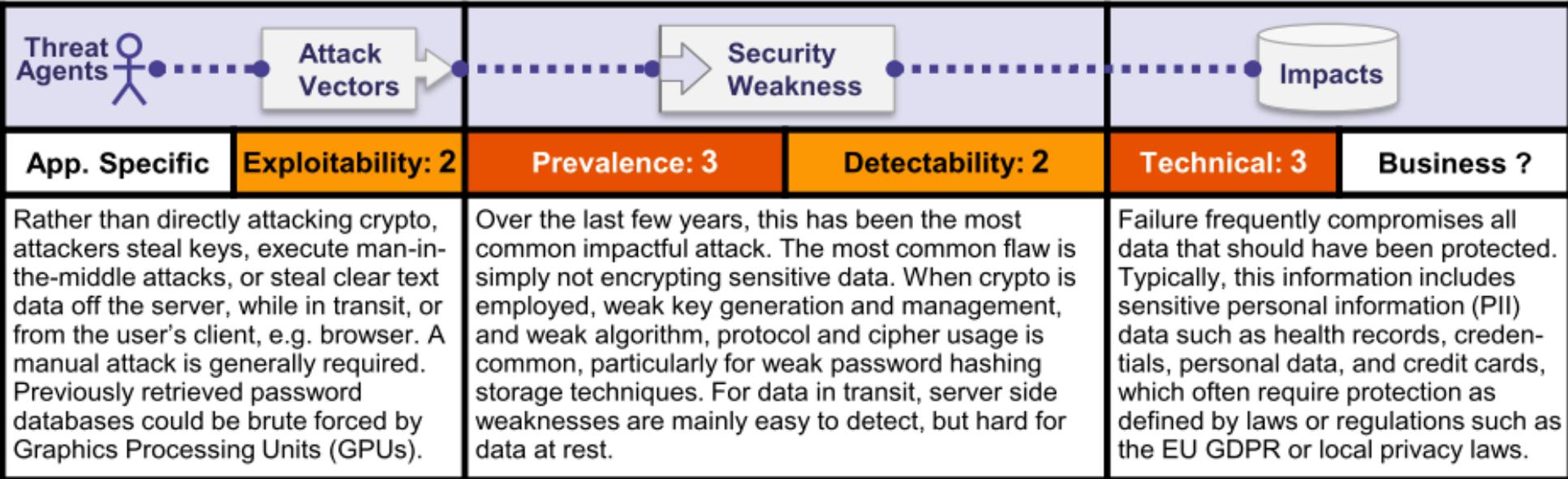
1. <https://pages.nist.gov/800-63-3/sp800-63b.html#memsecret>

A2:2017 – Broken Authentication and Session Management - Was dagegen tun?

- Zufällige Session IDs
- Session ID über HTTPS
- Cookies HTTPOnly
- Logout-Funktion
- Timeout
- Serverseitiger Session-Manager: Session ID zufällig, geschützt, nach Logout und Timeout invalidiert

Ein guter WAF garantiert, dass alle Internet-Verbindungen chiffriert sind

A3:2017 – Sensitive Data Exposure - **Worum geht es?**



- Z.B. Passwörter und Bankkundendaten in Logfiles
- Passwörter als Klartext gespeichert.

A3:2017 – Sensitive Data Exposure - **Wie finden?**

- Welche Daten sind besonders schützenswert?
- Wo und wie werden diese Daten gespeichert, inkl. Backup?
- Wie werden diese Daten übertragen?
- Werden schwache / veraltete Kryptoverfahren / veraltete SSL Versionen genutzt?
- Werden schwache Schlüssel erzeugt?
- Fehlt eine geeignete Schlüsselverwaltung?
- Wird Verschlüsselung erzwungen?

A3:2017 – Sensitive Data Exposure - Was dagegen tun?

- Daten klassifizieren
- Restriktive Rechtevergabe («deny by default»)
- Nur nötige Daten speichern
- Schützenswerte Daten verschlüsseln (Speicherung und Übertragung)
- Starke Algorithmen und Schlüssel verwenden
- Angemessene Schlüsselverwaltung
- HTTP Strict Transport Security (HSTS)
- Passwörter mit angemessener Hash-Funktion mit Salt speichern (z.B. PBKDF2)
- Autovervollständigung und Caching für Formulare mit vertraulichen Inhalten deaktivieren

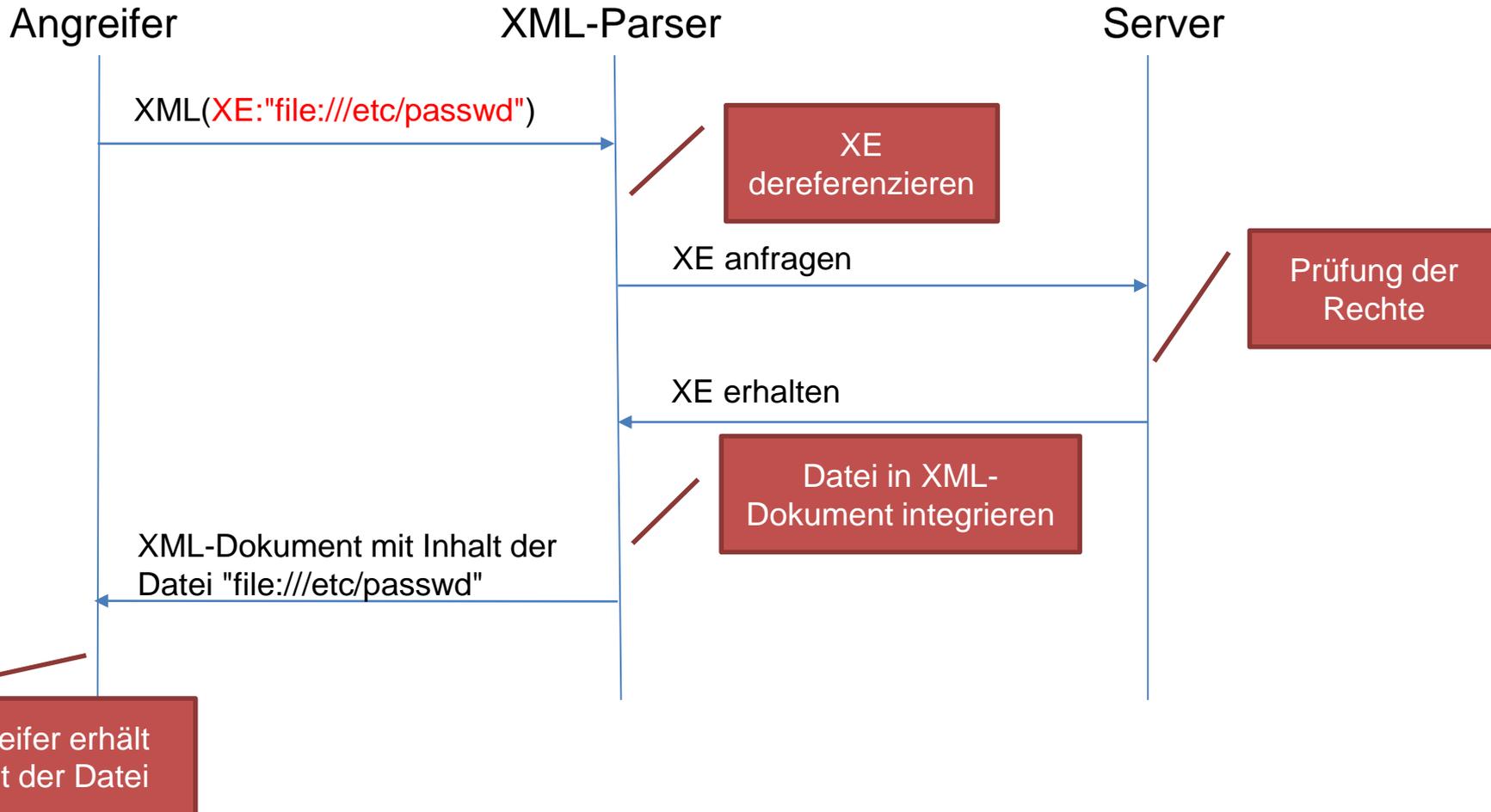
A4:2017 – XML External Entities (XXE) - **Worum geht's?**

| | | | | | | | |
|--|--------------------------|---|-------------------------|---|-------------------|---|--|
|  | |  | |  | |  | |
| App. Specific | Exploitability: 2 | Prevalence: 2 | Detectability: 3 | Technical: 3 | Business ? | | |
| <p>Attackers can exploit vulnerable XML processors if they can upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies or integrations.</p> | | <p>By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. SAST tools can discover this issue by inspecting dependencies and configuration. DAST tools require additional manual steps to detect and exploit this issue. Manual testers need to be trained in how to test for XXE, as it not commonly tested as of 2017.</p> | | <p>These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks.</p> <p>The business impact depends on the protection needs of all affected application and data.</p> | | | |

- SAST (Static Application Security Testing → Source Code Review)
- DAST (Dynamic Application Security Testing)

Ein WAF kann DTD verbieten und damit XXE verhindern.

A4:2017 – XML External Entities (XXE) - Worum geht's?



A4:2017 – XML External Entities (XXE) - **Wie finden?**

- Source Code Analysis Tools (SAST)
- Dynamic Application Security Testing (DAST)
- Manuelle Tests
 - Akzeptiert die Anwendung XML (direkt oder als Upload)?
 - Document Type Definitions (DTD) sind aktiviert
 - SAML kann anfällig sein
 - SOAP vor V1.2
- «Billion Laughs»

A4:2017 – XML External Entities (XXE) - Was dagegen tun?

- Schulungen
- Nur Dateiformate ohne Referenzen von nicht vertrauenswürdigen Quellen akzeptieren
- XML-Prozessoren und -Bibliotheken aktuell halten
- SOAP 1.2 oder neuer verwenden
- XML External Entity und DTD abschalten
- Serverseitiges Whitelisting
- Code-Review
 - Manuell
 - SAST (Source Code Analysis Tools)

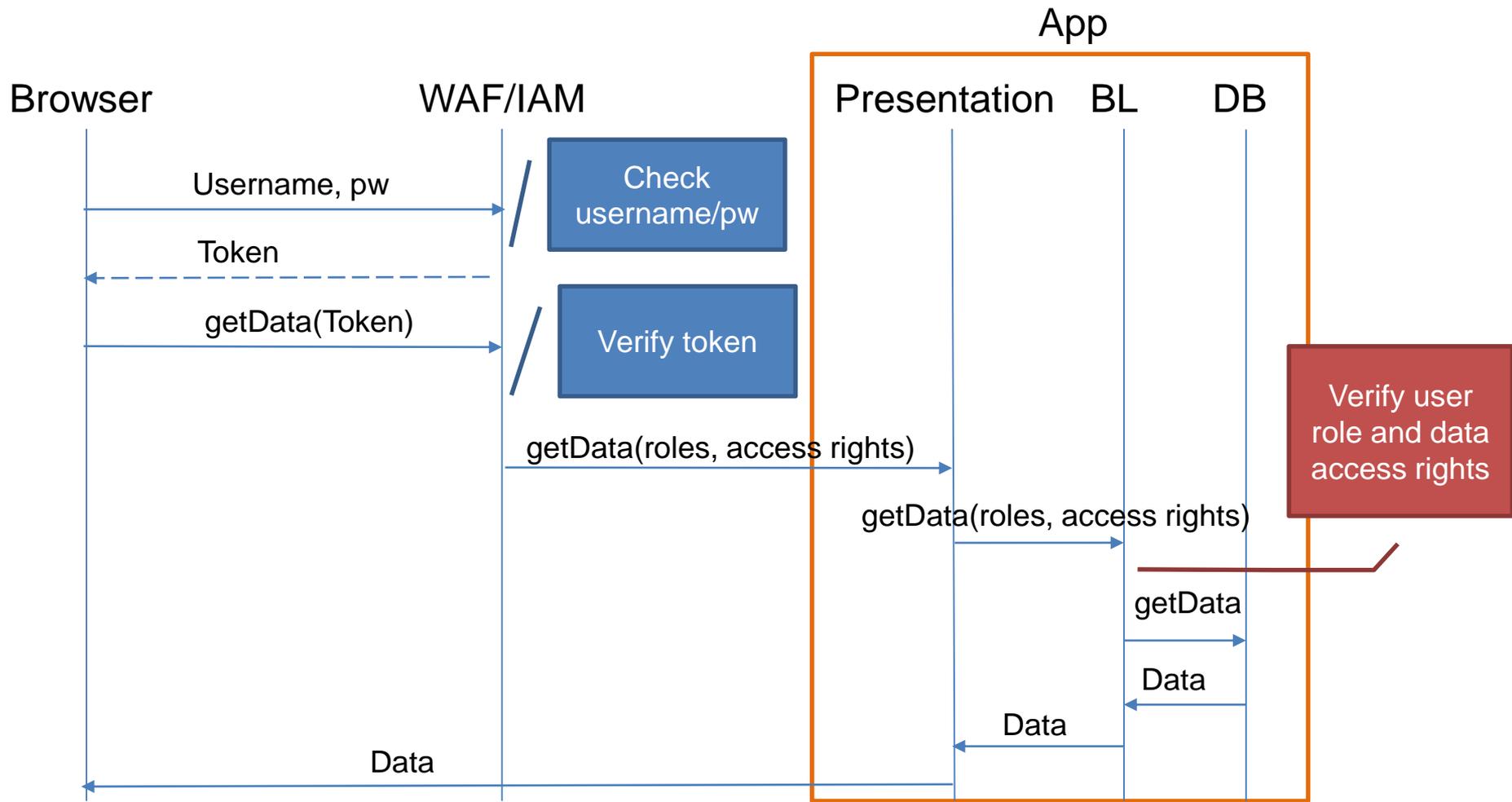
A5:2017 – Broken Access Control - **Worum geht's?**

| | | | | | |
|--|--------------------------|---|-------------------------|---|-------------------|
| | | | | | |
| App. Specific | Exploitability: 2 | Prevalence: 2 | Detectability: 2 | Technical: 3 | Business ? |
| <p>Exploitation of access control is a core skill of attackers. SAST and DAST tools can detect the absence of access control but cannot verify if it is functional when it is present. Access control is detectable using manual means, or possibly through automation for the absence of access controls in certain frameworks.</p> | | <p>Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers.</p> <p>Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc.</p> | | <p>The technical impact is attackers acting as users or administrators, or users using privileged functions, or creating, accessing, updating or deleting every record.</p> <p>The business impact depends on the protection needs of the application and data.</p> | |

- SAST (Static Application Security Testing → Source Code Review)
- DAST (Dynamic Application Security Testing)

Häufig kommen die Rollen aus dem IAM. Die Anwendungen müssen sie umsetzen.

A5:2017 – Broken Access Control - **Worum geht's:**



A5:2017 – Broken Access Control - **Wie finden?**

- DAST (Dynamic Application Security Testing) / SAST (Source Code Analysis Tools) erkennen fehlende Zugangskontrolle
ABER: Funktionsfähigkeit wird nicht bewertet!
- Manuelle Tests (Fachbezug notwendig)

A5:2017 – Broken Access Control - Was dagegen tun?

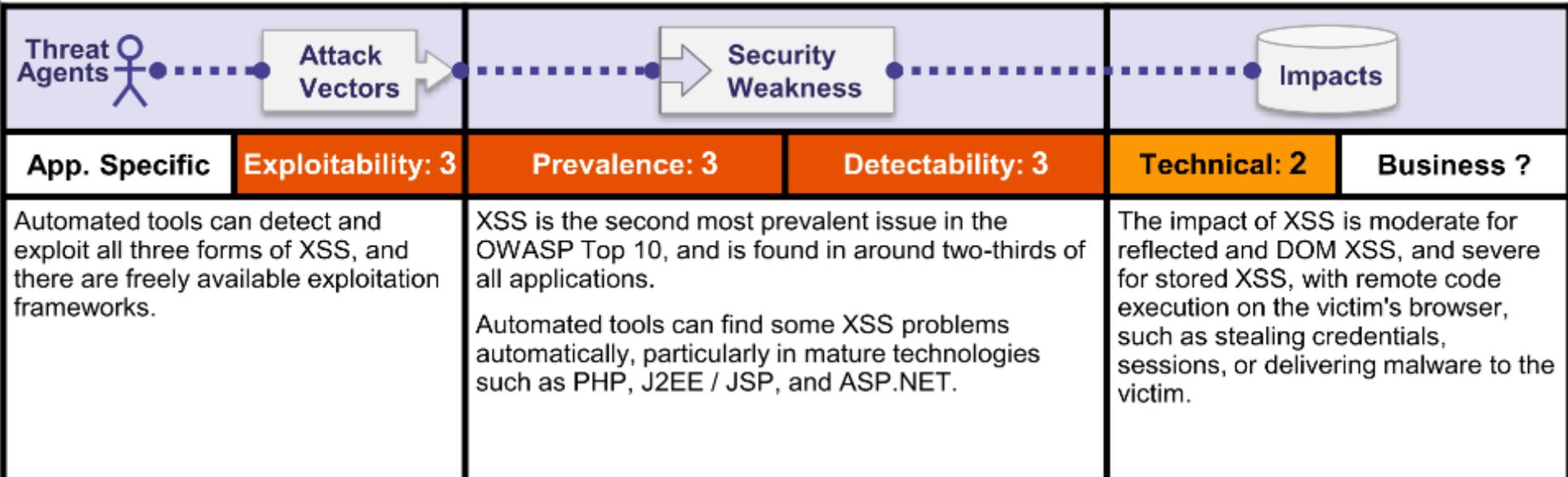
- Berechtigungszuweisung nach «Deny by default» (Ausnahme: öffentliche Ressourcen)
- Zentrale Prüfung von Zugriffsrechten
- Indirekte / zufällige Objektreferenzen wählen
- Jeden Zugriff auf ein Objekt autorisieren
- Token und Referenzen nach Logout invalidieren

A6:2017 – Security Misconfiguration

| | | | | | |
|---|--------------------------|---|-------------------------|---|-------------------|
| | | | | | |
| App. Specific | Exploitability: 3 | Prevalence: 3 | Detectability: 3 | Technical: 2 | Business ? |
| <p>Attackers will often attempt to exploit unpatched flaws or access default accounts, unused pages, unprotected files and directories, etc to gain unauthorized access or knowledge of the system.</p> | | <p>Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Automated scanners are useful for detecting misconfigurations, use of default accounts or configurations, unnecessary services, legacy options, etc.</p> | | <p>Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise. The business impact depends on the protection needs of the application and data.</p> | |

- Auf Anwendungsstufe Verantwortung des Entwicklers (z.B. Libraries bei Mobile-App).

A7:2017 – Cross-Site Scripting (XSS) - **Worum geht's?**



- Drei Arten XSS:
 - Reflektierte XSS
 - Persistente XSS
 - DOM-basierte XSS

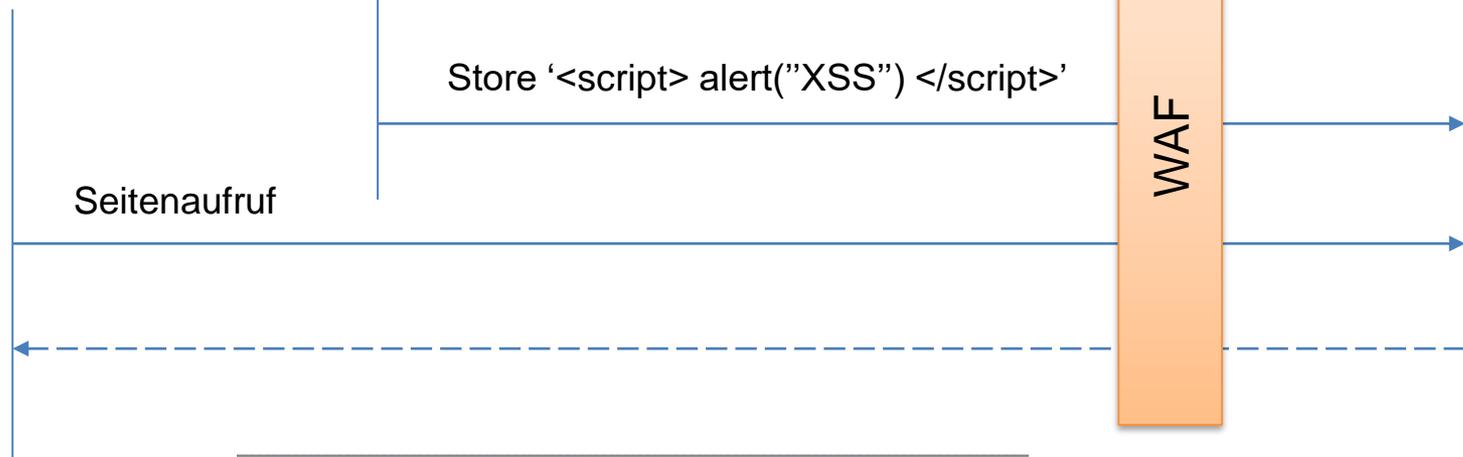
Ein guter WAF kann heikle Pattern erkennen

A7:2017 – Cross-Site Scripting (XSS) - Worum geht's?

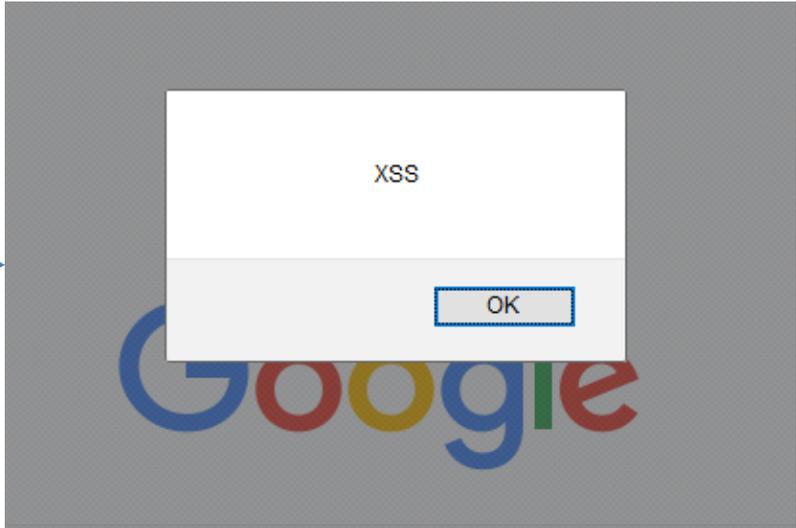
Stored XSS

Opfer

Angreifer



Ausführung



A7:2017 – Cross-Site Scripting (XSS) - **Wie finden?**

- Automatische Tools (z.B. Burp)
- Code-Analyse

A7:2017 – Cross-Site Scripting (XSS) - Was dagegen tun?

- Benutzereingaben prüfen (nach Möglichkeit)
- Ausgabe maskieren (immer)
- Content Security Policy (CSP) aktivieren
- Benutzung von Framework mit automatischem XSS-Schutz, z.B.:
 - Ruby on Rails
 - React JS

A8:2017 – Insecure Deserialization - **Worum geht's?**

| | | | | | |
|---|--------------------------|---|-------------------------|--|-------------------|
| | | | | | |
| App. Specific | Exploitability: 1 | Prevalence: 2 | Detectability: 2 | Technical: 3 | Business ? |
| <p>Exploitation of deserialization is somewhat difficult, as off the shelf exploits rarely work without changes or tweaks to the underlying exploit code.</p> | | <p>This issue is included in the Top 10 based on an industry survey and not on quantifiable data.</p> <p>Some tools can discover deserialization flaws, but human assistance is frequently needed to validate the problem. It is expected that prevalence data for deserialization flaws will increase as tooling is developed to help identify and address it.</p> | | <p>The impact of deserialization flaws cannot be understated. These flaws can lead to remote code execution attacks, one of the most serious attacks possible.</p> <p>The business impact depends on the protection needs of the application and data.</p> | |

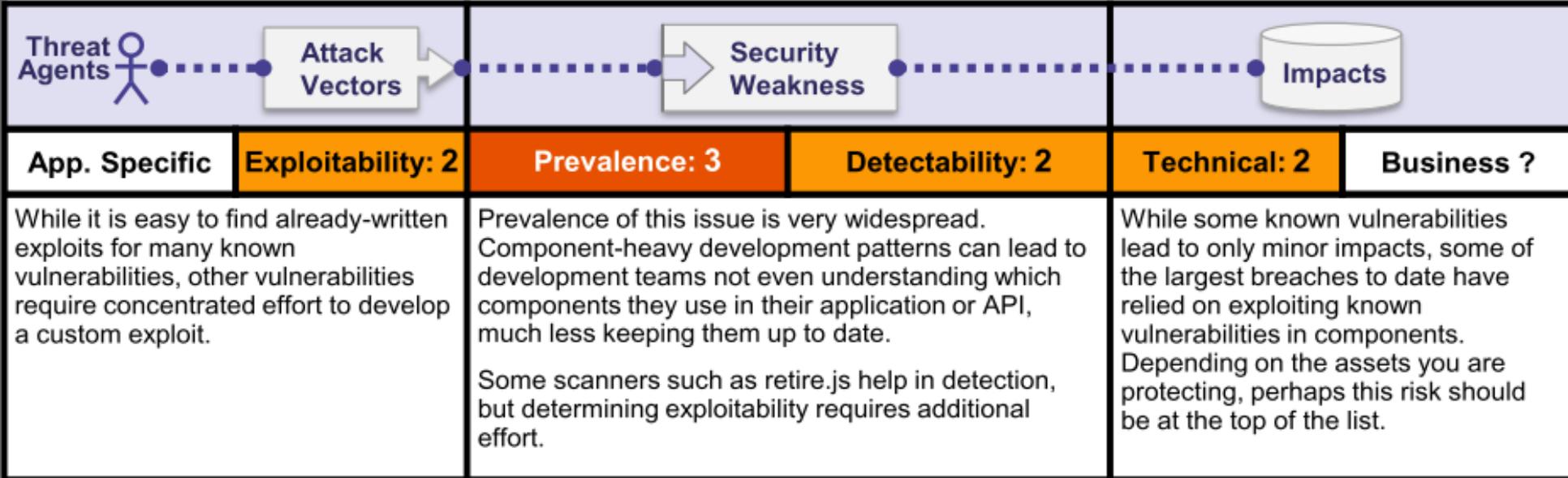
A8:2017 – Insecure Deserialization - **Wie finden?**

- Scanner (mit Einschränkungen)
- Deserialisierung von ungeprüften Daten bedeutet Anfälligkeit

A8:2017 – Insecure Deserialization - Was dagegen tun?

- Unrealistisch:
 - Keine serialisierten Daten aus unbekanntem Quellen akzeptieren
- Realistisch:
 - Integritätsprüfungen von serialisierten Objekten (digitale Signatur)
 - Enforce Type Constraints
 - Deserialisierung in isolierte Umgebung mit wenig Rechten
 - Logging und Monitoring

A9:2017 – Using Components with Known Vulnerabilities



A10:2017 – Insufficient Logging & Monitoring - **Worum geht's?**

| | | | | | |
|--|--------------------------|---|-------------------------|--|-------------------|
| | | | | | |
| App. Specific | Exploitability: 2 | Prevalence: 3 | Detectability: 1 | Technical: 2 | Business ? |
| <p>Exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident.</p> <p>Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected.</p> | | <p>This issue is included in the Top 10 based on an industry survey.</p> <p>One strategy for determining if you have sufficient monitoring is to examine the logs following penetration testing. The testers' actions should be recorded sufficiently to understand what damages they may have inflicted.</p> | | <p>Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to nearly 100%.</p> <p>In 2016, identifying a breach took an average of 191 days – plenty of time for damage to be inflicted.</p> | |

- Fehlgeschlagene Logins nicht geloggt.
→ Sind aber häufig interessant

A10:2017 – Insufficient Logging & Monitoring - **Wie finden?**

- Prüfen der Logs nach einem Penetrationstest
 - Sind die Aktionen in den Log-Dateien abgebildet?
 - Werden sinnvolle Alarme ausgelöst?
- Werden Logins / gescheiterte Logins geloggt?
- Werden Transaktionen geloggt?
- Warnungen und Fehler generieren keine / unklare Log-Einträge
- Logs von API und Applikation werden nicht überwacht
- Log-Dateien werden nur lokal gespeichert
- Fehlende / inadäquate Alarmierung
- Aktive Attacken werden nicht zeitnah detektiert und alarmiert

A10:2017 – Insufficient Logging & Monitoring - Was dagegen tun?

- Festlegen, welche Events geloggt werden müssen (nicht abschliessend):
 - Authentisierungs-/Autorisierungsversuche (erfolgreich und fehlgeschlagen)
 - Fehler (Input- / Output-Validierung, Session Management)
 - Aktivitäten auf hoch privilegierten Funktionen (z.B. von Admins)
- Log-Format zur zentralen Überwachung definieren und einhalten
- Audit-Trail sicherstellen
- Effektive Überwachung und Alarmierung
- Pläne für den Ereignisfall
- Wiederherstellungsplan

What's Next for Developers

- OWASP Top 10 **nicht** abschliessend
- Was ist Sicherheit für «meine» Applikation?
- Security by Design
- Standard-Kontrollmechanismen¹
- Sichere Entwicklungsstrategie (Secure Development Lifecycle)
- Verwendung von entsprechenden Frameworks
- Üben → heute: WebGoat
 - «HackingLab» als Alternative (<https://www.hacking-lab.com>)
- **Achtung: gemäss Art. 143bis Abs. 1 StGB² ist das «unbefugte Eindringen in ein Datenverarbeitungssystem» («Hacken») strafbar und kein Kavaliersdelikt.**

1. https://www.owasp.org/index.php/OWASP_Proactive_Controls

2. <https://www.admin.ch/opc/de/classified-compilation/19370083/index.html#a143bis>

OWASP Top 10 – Wie kann ich auf die Anfälligkeit testen?

| No. | automatisch | manuell |
|--|---|----------------------|
| A1 Injection | (✓) | ✓ source code review |
| A2 Broken Authentication | | ✓ |
| A3 Sensitive Data Exposure | | ✓ |
| A4 XML External Entities (XXE) | ✓ | ✓ |
| A5 Broken Access control | (✓) fehlende Authentisierung wird erkannt | ✓ |
| A6 Security Misconfiguration | ✓ | ✓ |
| A7 Cross-Site Scripting (XSS) | ✓ | ✓ |
| A8 Insecure Deserialization | (teilweise) | ✓ |
| A9 Using Components with Known Vulnerabilities | (teilweise) | ✓ |
| A10 Insufficient Logging & Monitoring | | ✓ |

Online-Übungen

- Die WebGoat-Übungen sind auch online verfügbar.
<https://webgoat.cnlab.ch/WebGoat>
username: webgoat
password: cnlab

Weitere Informationen

- OWASP Top 10 für Entwickler:
https://www.owasp.org/index.php/Category:OWASP_Top_10_fuer_Entwickler
- OWASP Cheat Sheets:
https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series
- OWASP Dependency Check (A9):
https://www.owasp.org/index.php/OWASP_Dependency_Check
- OWASP Proactive Controls:
https://www.owasp.org/index.php/OWASP_Proactive_Controls
- OWASP Application Security Verification Standard Project:
https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

Danke

Thomas Lüthi

thomas.luethi@cnlab.ch

+41 55 214 33 41

Stephan Verbücheln

stephan.verbuecheln@cnlab.ch

+41 55 214 33 36

Martina Minges

martina.minges@cnlab.ch

+41 55 214 33 42