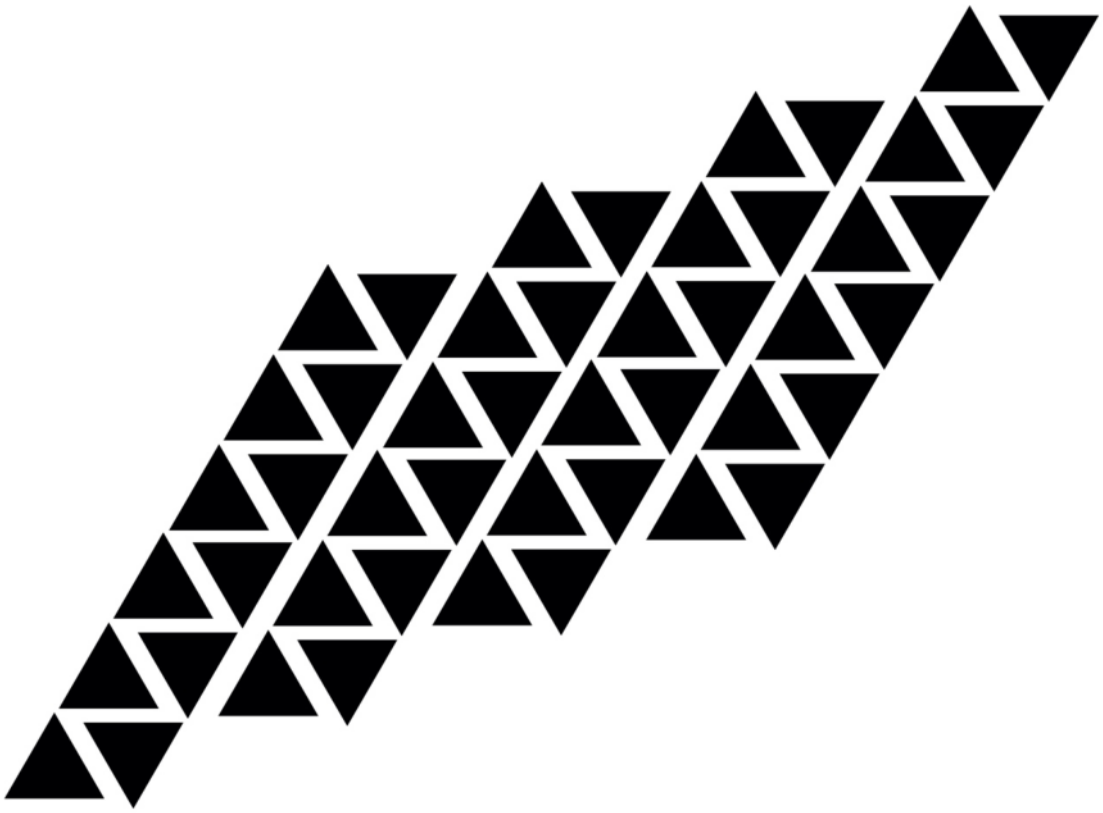


Sicherheitsanalyse

Abschlussbericht

SleepyLogistics GmbH

Version:	1.0
Stand:	DD.MM.JJJJ
Status:	Intern / Extern

KUNDE	EXEKUTIVE INSTANZ	PROJEKT-VOLUMEN	TEST 1	TEST 2	MITARBEITER
SleepyLogistics GmbH Musterstr. 13 37073 Göttingen	AWARE7 GmbH Munscheidstr. 14 45886 Gelsenkirchen	10 PT	29.06 – 07.07.	20.07 – 22.07	Marius Müller Michael Meier Jan Schneider Peter Müller

Der vorliegende Bericht wurde erstellt von:

Dr./M./B. Sc. Name, Vorname

DD.MM.JJJJ

Datum

Unterschrift
Projektverantwortlicher

Dokumenteninformationen

VERSION	DATUM	AUTOR	KOMMENTARE
0.1	29.06.20	Große-Kampmann, Matteo Projektverantwortlicher	Erstellung
0.2	03.07.20	Müller, Marius Senior Pentest Expert	Ergebnisse
1.0	21.07.20	Große-Kampmann, Matteo Projektverantwortlicher	Finales Dokument

Dieses Dokument spiegelt nur einen **Auszug** eines professionellen Pentest Berichts wider. Die Ergebnisse sind anonymisiert und mit dem Einverständnis der Unternehmen untergebracht. Bei Fragen können Sie uns gerne kontaktieren: **0209 8830 6760**

Abbildungsverzeichnis

1. Management Summary

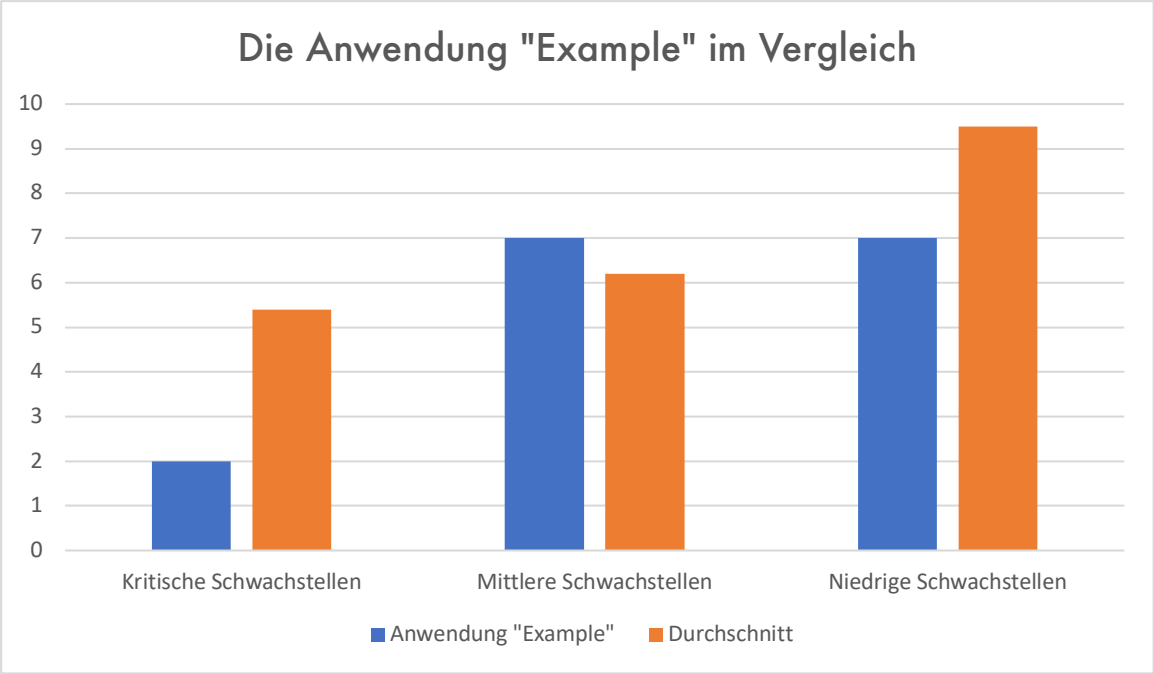
Die Beispiel GmbH beauftragte die AWARE7 GmbH, ihr Produkt “Example” zu testen. Details zu den bewerteten Systemen sind im Kapitel Allgemeine Informationen zur durchgeführten Sicherheitsbewertung zu finden. Die Untersuchung wurde nach dem aktuellen Stand der Technik und durch fachkundiges Personal durchgeführt. Die Tests wurden mit größtmöglicher Sorgfalt und nach bestem Gewissen durchgeführt. Als Testmethodik diente die aktuelle Version des Schwachstellenkatalogs vom Open Web Application Security Projekt (OWASP). Wenn für eine entsprechende Kategorie keine Bedrohung gefunden wird, haben wir die Kategorie nicht in den Bericht aufgenommen.

Während unserer Bewertung fanden wir zwei Schwachstellen in der API, die als "hoch" kategorisiert wurden. Mit Ausnahme der Schwachstelle "Privilege Escalation" sollten alle Schwachstellen ohne hohen Aufwand behoben werden können, da es sich um Konfigurationsprobleme handelt. Die Privilege Escalation muss mit mehr Aufwand mitigiert werden, da Berechtigungen auf einer hierarchischen Ebene überprüft und neu bewertet werden müssen.

Die Anwendung an sich enthält zwei Schwachstellen, die wir als "hoch" kategorisiert haben. Beide Schwachstellen sind aus unserer Sicht ohne großen Aufwand zu beheben, da es sich um ebenfalls Konfigurationsprobleme handelt. Der Ressourcen-Endpunkt enthält eine Schwachstelle mit der Kategorisierung "Mittel". Ein Angreifer kann drei "niedrig" kategorisierte Schwachstellen zu einer "mittleren" Schwachstelle verketten. Dies ist ausschließlich ein serverseitiges Konfigurationsproblem und sollte mit nicht allzu großem Aufwand gepatcht werden.

2. Die Sicherheit Ihrer Anwendung im Benachmarking

Die Anwendung „Example“ verfügt über ein hohes Sicherheitsniveau. Im Vergleich zu vorherigen Untersuchungen wurden nicht nur weniger Schwachstellen generell, sondern auch im speziellen weniger kritische Sicherheitslücken entdeckt.



Es konnten keine Schwachstellen identifiziert werden, welche den produktiven Betrieb der Plattform ad-hoc außer Kraft setzen könnten. Die zwei identifizierten, als hoch klassifizierten Schwachstellen sollten schnellstmöglich geschlossen werden.

(…)

3. Zielsetzung

Die SleepyLogistics GmbH beauftragte die AWARE7 GmbH mit der Aufgabe ihre grundlegende Webanwendung, bestehend aus nachfolgenden Systemen, auf Sicherheitslücken zu untersuchen. Der Umfang des Tests bestand darin, diese drei Systeme, die drei hierarchische Rollen anbieten, zu untersuchen. Der Angriff wurde von extern auf die unten aufgeführten Systeme durchgeführt. Jedes andere System, das auf der Domäne "example.com" gehostet wird, ist nicht Teil der Untersuchung.

System	Kommentar
api.example.com	RESTful API
www.example.com	Webseite
Example.example.com	Resource Provider

Die Priorisierung der Maßnahmen leitet sich direkt aus der Kritikalität der Schwachstelle und dem identifizierten Risiko ab. Der zeitliche Horizont zum Beseitigen der Schwachstellen wird wie folgt definiert:

zeitnah	Reaktion unmittelbar
kurzfristig	Reaktion innerhalb von 4 Wochen
mittelfristig	Reaktion innerhalb von 3 Monaten
langfristig	Reaktion innerhalb von 6 Monaten

Falls verfügbar werden zu den identifizierten Schwachstellen die zugehörigen Common Vulnerabilities and Exposure IDs aufgeführt. Durch den Industriestandard ist ein Austausch der Informationen zwischen diversen Datenbanken problemlos möglich.

Eine technische Sicherheitsanalyse muss immer als Betrachtung zu einem Stichtag aufgefasst werden. Die zu diesem Tag bekannten Schwachstellen und Angriffstechniken werden genutzt. **Aussagen über die Sicherheit eines Systems für die Zukunft lassen sich nicht treffen**. Änderungen an den Systemeinstellungen können negative wie auch positive Auswirkungen auf die Systemsicherheit haben.

4. Ergebnisse Port-Scans

Die folgenden Tabellen zeigen die Ergebnisse des Port-Scans gegen die beauftragten Systeme:

4.1.1. Port-Scan von api.example.com

PORT	PROTOKOLL	SERVICE
80	tcp	HTTP
443	tcp	HTTPS
53	tcp	DNS

Schwachstellen am Port 443/tcp
Der verwendete HTTPS-Server fordert keine HSTS Header
Bedrohung Integrität: **niedrig**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit **mittel**

Ein Angreifer könnte dies ausnutzen, um Downgrading oder SSL-Stripping MITM-Angriffe durchzuführen.
Wahrscheinlichkeit: **niedrig**

Der Server sollte so konfiguriert werden, dass HSTS-Header genutzt werden.
Realisierungszeitraum: **langfristig**

(…)

4.1.2. Port-Scan von www.example.com

(…)

4.1.3.Port-Scan von example.example.com

(…)

5. Ergebnisse – www.example.com

Dieses Kapitel fasst die Testergebnisse für die Anwendung api.example.com zusammen. Es wurden nur uns bekannte Systeme im Rahmen dieses Tests getestet.

5.1.1.Exponierte Session Variablen

Da Session-Variablen die Authentisierung eines Benutzers beinhalten, muss geprüft werden, ob diese Daten im Laufe der Sitzung über Parameter, unverschlüsselte Verbindungen oder andere Fehlkonfigurationen einem Angreifer bekannt werden.

Die Session-Variable wird in der URL gespeichert.
Bedrohung Integrität: **niedrig**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit: **niedrig**

Für Session-IDs, die in der URL übertragen werden, besteht ein erhöhtes Risiko, dass diese im Browser oder in Proxies zwischengespeichert werden, oder auf anderem Wege in falsche Hände gelangen. Die Session-ID ermöglicht entsprechenden Zugriff auf die in der Plattform angegebenen, sensiblen Informationen.
Wahrscheinlichkeit: **gering**

Die Session-ID sollte in einem sicher konfigurierten Cookie oder dem Session-Store des Browsers gespeichert werden und nur eine begrenzte Gültigkeit besitzen.
Realisierungszeitraum: **mittelfristig**

5.1.2.Web-Applikations-Fingerprinting

In diesem Test wird versucht zu identifizieren, welche Software bzw. Applikation genutzt wird, um das Portal zu betreiben. Zu den bekannten Web-Applikationen gehören z.B. Wordpress oder phpBB. Basierend auf diesen Informationen lassen sich weitere Angriffe besser planen.

Über Web-Applikation Fingerprinting konnten folgende Informationen gewonnen werden:

Auszug HTML-Quelltext:

```
<! - - Example Software Web CRM System – Darkhouse Studio San Francisco 9_2_0 - - >
<! - - Copyright Darkhouse Software, Inc. 1998 – 2020 –
www.example.com – USA – (000)  444 – 8888 -- >
```

5.1.3.Passwort-Speicherung

Zahlreiche Portal erlauben es Benutzern, das Login-Passwort im Browser zu speichern. Dies ermöglicht dem Browser beim nächsten Besuch der Web-Applikation das Passwort bereits vorauszufüllen. Zusätzlich bieten einige Web-Applikationen eine „Eingeloggt bleiben" Funktionalität an. Benutzer müssen dann für einen gewissen Zeitraum keine Passworteingabe mehr vornehmen. In diesem Test wird die Web-Applikation auf schwache Passwort-Speicherungsmethoden untersucht.

Im Portal sind keine Maßnahmen etabliert, die eine lokale Speicherung der eingegebenen Passwörter verhindern.

Das Portal lässt die lokale Speicherung der Passwörter-Formulare zu.
Bedrohung Integrität: **niedrig**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit: **mittel**

Ein Angreifer mit lokalem Zugriff auf einen Client-PC könnte die Informationen aus dem Browser auslesen und diese für einen weiteren Datendiebstahl aus dem Portal missbrauchen.
Wahrscheinlichkeit: **mittel**

Alle Formulare, die Passwort-Informationen verarbeiten, sollten um das Autocomplete-Attribut erweitert werden:

```
<INPUT TYPE=”password“ AUTOCOMPLETE=“off“>
```

Realisierungszeitraum: mittel

5.1.4.SSL/TLS Protokollunterstützung. Algorithmen und bekannte Schwachstellen

Verfahren zur Verschlüsselung von Inhalten veralten und müssen an neue Gegebenheiten angepasst werden. In diesem Test wird die Plattform daraufhin untersucht, ob ausschließlich anerkannte und zum Zeitpunkt der Untersuchung sichere Algorithmen angeboten und verwendet werden.

Der Server akzeptiert alle Signatur-Algorithmen, so dass auch unsichere Algorithmen akzeptiert werden und Daten unsicher übertragen werden. Der Server

- api.example.com akzeptiert alle Signatur-Algorithmen

Abbildung 1: Der untersuchte Server akzeptiert alle Signatur-Algorithmen

Die Kommunikation mit dem Server kann durch einen veralteten Algorithmus verifiziert werden. Dies könnte es Angreifern erleichtern, eine Signatur zu fälschen.

Der Server unterstützt TLS 1.0 als sicheres Kommunikationsprotokoll und unterstützt nicht das TLS Fallback SCSV. Zusätzlich werden unsichere Chiffren, die aus DES und SHA bestehen, unterstützt. Insbesondere die Chiffre DES-CBC3-SHA ist bedenklich.

Der Screenshort zeigt, dass der Server TLS 1.0 unterstützt. In Kombination mit dem fehlenden TLS Fallback SCSV und der Unterstützung unsicherer Chiffren, bewerten wir das Risiko größtenteils im mittleren Gefahrensektor, da ein Angreifer TLS 1.0 absichtlich verwenden kann.

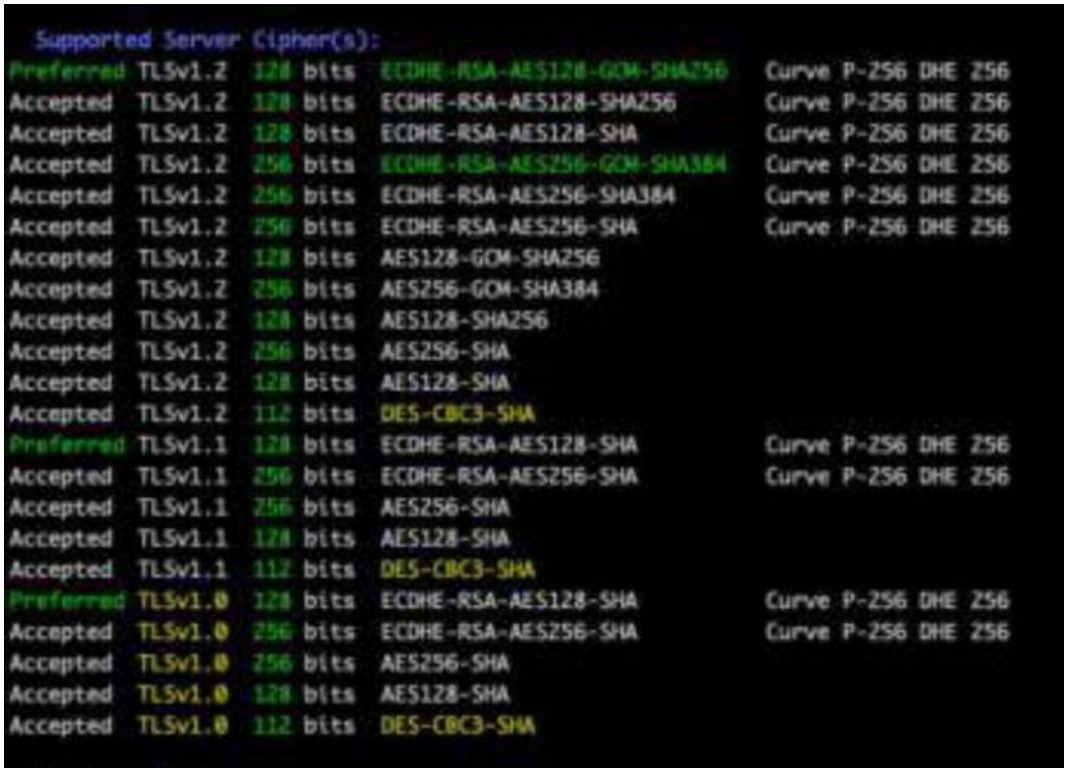


Abbildung 2: Der Server akzeptiert mittlerweile als unsicher geltende Signatur-Algorithmen

TLS 1.0 ist eine veraltete Version des Transport Layer Security Protocols. Es gibt mehrere Schwachstellen im Protokoll, die es einem Angreifer ermöglichen die Vertraulichkeit, Integrität und Verfügbarkeit von Servern, die TLS 1.0 unterstützen, zu brechen. Insbesondere wenn es mit einem fehlenden Fallback-Schutz gekoppelt ist und schwache Verschlüsselungs-Suites unterstützt werden. Ein Angreifer kann willkürlich den schwächsten, möglichen Verschlüsselungsmechanismus wählen und diesen brechen.

Bedrohung Integrität: **mittel**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit: **mittel**

Die Unterstützung für TLS 1.0 sollte eingestellt werden. Chrome markiert Webseiten als unsicher, die TLS 1.0 noch unterstützen und deren PCI DSS-Konformitätsfrist im Juni 2018 ablief. Zusätzlich sollte der TLS-Fallback aktiv sein, damit die stärkste TLS-Version gewählt wird wie möglich. Es wird empfohlen, nur die neusten Signatur-Algorithmen zuzulassen und regelmäßig zu überprüfen, ob der aktuelle Patch-Status den aktuellen Empfehlungen und dem Stand der Technik entspricht. DES-CBC3-SHA sollte nicht als Cipher-Suite unterstützt werden.
Realisierungszeitraum: **mittelfristig**

5.1.5. Stored Cross-Site-Scripting

Benutzereingaben des Portals werden in vielen Bereichen genutzt, um dargestellte Informationen anzupassen. Das kann sich z.B. auf die Darstellung des Benutzernamens beziehen. Enthalten diese Benutzereingaben Script-Code, kann dieser durch den Browser unbeabsichtigt zur Ausführung gebracht werden.

Die Webanwendung sowie die API ist anfällig für stored Cross-Site-Scripting. Dies kommt hauptsächlich durch eine fehlende Eingabevalidierung.

- https://api.example.com/example-api/event
  - title

- https://api.example.com/example-api/user/3425?_param=1sa24519
  - param

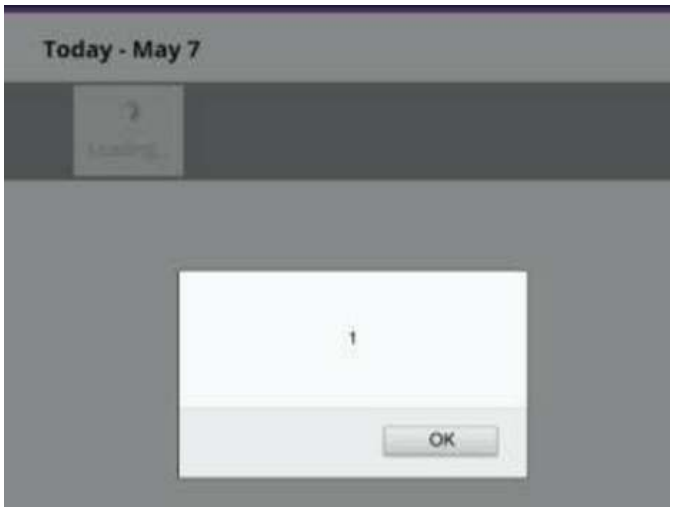


Abbildung 3: PoC der stored Cross-Site-Scripting Schwachstelle in der Webanwendung

Bedrohung Integrität: **hoch**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit: **hoch**

XSS-Fehler treten immer dann auf, wenn eine Anwendung nicht vertrauenswürdige Daten an einen Webbrowser sendet, ohne diese ordnungsgemäß zu validieren oder zu verwerfen. Eine XSS Schwachstelle ermöglicht es Angreifern Skripte im Browser des

Opfers auszuführen, die Sitzungen zu übernehmen, Webseiten Inhalte zu manipulieren oder den Benutzer auf bösartige Webseiten umzuleiten.

Um die Anwendung vor Cross-Site-Scripting zu schützen, müssen alle Benutzereingaben gefiltert werden. Es ist wichtig sicherzustellen, dass Benutzerdaten- und Befehle getrennt behandelt werden. Zum Filtern der Benutzereingaben wird empfohlen, eine API (wie z.B. OWASP ESAPI) zu verwenden. Die vorgestellte API bietet verschiedene Schnittstellen für verschiedene Arten von Eingaben wie z.B. Strings oder ganze Zahlen). Ebenfalls gefiltert werden alle HTML-Zeichen und Funktionen von z.B. PHP.
Realisierungszeitraum: **kurzfristig**

5.1.6. Session Management Schema

Session-IDs sind (nach Benutzername und Passwort) das zentrale Authentisierungsmerkmal bei der Benutzung eines Web-Portals. Entsprechend sicher muss die Initiierung und der Transport von Session-IDs realisiert werden. Im vorliegenden Test wird daher das Session-Management des Portals überprüft.

Die genutzte Session-ID hid_s sind so zufällig, dass nicht davon ausgegangen werden kann, dass ein Angreifer einen Cookie erfolgreich erraten kann.

Die statistische Qualität der Session Cookies ist: exzellent:

- hid_s hat einen exzellenten Entropie-Wert
  (133 Bit bei 11000 Tokens und 1% Signifikanz)

5.1.7. Privilege Escalation

Benutzern werden innerhalb des Portals bei einem Login-Vorgang Rollen und zugehörige Rechte zugewiesen. Es wird daher geprüft, ob ein Benutzer die eigene Rolle verlassen und Befehle für andere Benutzer (mit einer anderen, höheren Rolle) ausführen kann.

Es wurde festgestellt, dass es für einen authentifizierten Benutzer möglich ist, mehr Privilegien zu erhalten und Root-Adminstrator zu werden. Ein authentifizierter Benutzer ist in der Lage, auf das folgende API-Backend und die folgenden Funktionen zuzugreifen:

- https://api.example.com/example-api/user

Der Benutzer ist in der Lage, eine manipulierte http-POST-Anfrage mit einem JSON-Body zu erstellen, der zur Generierung eines neuen Benutzers verwendet werden kann. Wenn ein neuer Benutzer erzeugt wird, enthält dieses .json-Dokument einen spezifischen „Role“ Parameter. Es ist möglich, dass ein authentifizierter Benutzer mit der Rolle „User“ oder „Manager“ einen neuen Benutzer mit der Rolle „Root“ erstellt.

Eine solche Privilege Escalation Schwachstelle ermöglicht es einem Benutzer mit geringen Privilegien Zugang zu Funktionen und Sensiblen Daten zu erhalten, zu deren Abfrage und Interaktion er nicht berechtigt ist.

Bedrohung Integrität: **hoch**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit: **hoch**

Es sollte für jede Anfrage geprüft werden, dass der Benutzer, der die jeweilige Aktion ausführen möchte, berechtigt ist auf die entsprechenden Daten zuzugreifen oder sie zu bearbeiten.
Realisierungszeitraum: **kurzfristig**

5.1.8.Clickjacking

Im Rahmen von clientseitigen Clickjacking-Angriffen wird der Nutzer durch Überlagerung von Elementen auf der Webseite dazu motiviert, Aktionen auf der Seite auszuführen, die er willentlich nicht ausführen wollte. Dieser Angriff basiert auf Schwachstellen in den clientseitigen Browsern und findet deswegen vor allem bei Tests der Clients statt.

Der Server ist hinsichtlich Clickjacking anfällig.
Bedrohung Integrität: **niedrig**
Bedrohung Verfügbarkeit: **niedrig**
Bedrohung Vertraulichkeit: **mittel**

Ein Angreifer könnte versuchen, Authentisierungsinformationen vom Benutzer zu erlangen.
Wahrscheinlichkeit **gering**

Der X-Frame-Schutz sollte direkt in der Konfiguration des Web-Servers aktiviert werden. Im Apache Web-Server wäre dies die Datei

```
httpd.conf
```

Alternativ kann die Option

```
Header append X-FRAME-OPTIONS “SAMEORIGIN”
```

Hinzugefügt warden.
Realisierungszeitraum: **mittelfristig**

7. Typische Schwachstellen auf Web-Portalen kurz erklärt

7.1. Fehler in der Authentisierung und dem Session Management

7.1.1. Session-Fixiation

Können die authentifizierenden Sessions nicht nur vom Web-Portal sondern auch von einem Benutzer erzeugt werden, wird von einer Session-Fixiation Schwachstelle bzw. Angriff gesprochen. Dabei wird versucht, auf dem System eines Opfers eine Session-ID zu hinterlegen, die zum einen vom Web-Portal akzeptiert wird und zum anderen dem Angreifer bekannt ist. Der Angreifer kann dann mit der bekannten, dem Opfer untergeschobenen Session-ID eine Anmeldesitzung am Portal übernehmen, während das Opfer angemeldet ist.

7.1.2. Session-Reuse

(…)

7.1.3. Session-Switching

(…)

9. Abschluss & Fazit der IT-Sicherheitsuntersuchung

(...)