Ergebnisbericht
Web Application Security Penetration Test

CIMS

Consline AG

16.09.2014, Version 1.0, Status: Final

Auftraggeber:
Dr. Heinz van Deelen

Autoren:
Sven Schleier, sven.schleier@securenet.de, +49 (89) 32133-663
Robert Kulzer, robert.kulzer@securenet.de, +49 (89) 32133-633

Inhalt vertraulich!

SecureNet GmbH

Frankfurter Ring 193a
80807 München

Tel.: +49/89/32133-600
Fax: +49/89/32133-699
E-Mail: info@securenet.de
http://www.securenet.de

A. Zusammenfassung und Bewertung

Sprungtabelle

Auf den folgenden Seiten sind die Ergebnisse der Sicherheitsanalyse zusammengefasst.

Alle Verweise in diesem Dokument sind aktive Links, d.h. können per Mausklick direkt angesprungen werden.

A1 Inhaltsverzeichnis

A2 Management Summary

Der vorliegende Bericht fasst die Ergebnisse der Untersuchung der Webanwendung von

https://cims1.consline.com

auf Ebene der Web Application Security und Netzwerkebene zusammen. Die Untersuchung erfolgte nach dem Blackbox-Verfahren, bei dem dem Tester keine zusätzlichen Informationen gegeben werden.

Die Tests fanden am 10.09.2014 und 11.09.2014 auf einem Testsystem statt und wurden über das Internet durchgeführt. Ebenso erfolgte ein Test am 15.09.2014 auf dem Produktivsystem.

Ergebnis

Die Anwendung weist

keine Sicherheitslücken

auf.

Schwachstellenübersicht

Die Untersuchung der Anwendung hat keine Sicherheitsmängel erbracht, insbesondere keine SQL- oder Command-Injection-Probleme, mit denen Zugriffe auf das Backend möglich wären.

Schwachstellen, die ein Eindringen in die Anwendung selbst, den Host, auf dem sie läuft oder das umgebende System erlauben, haben wir ebenfalls nicht nachweisen können.

Ein externer Angriff durch einen anonymen Benutzer ist nicht möglich. Ebenfalls sind unerlaubte Zugriffe zwischen den Organisationen, die gegen das Rollenkonzept verstoßen würden, nicht möglich.

Die (potentiellen) Schwachstellen, deren Gefahrenpotential mit [Info] bewertet worden sind, sollten vom Kunden einzeln dahingehend beurteilt werden, ob eine Behebung erforderlich ist.

Allgemeiner Hinweis zur Schwachstellenbehebung

Wir weisen darauf hin, dass es zur Bereinigung der Anwendung von Schwachstellen in der Regel nicht ausreichend ist, die in diesem Bericht aufgeführten Schwachstellen isoliert voneinander zu beheben. Vielmehr ist es erforderlich, dass die Anwendung mit generellen Konzepten zur Vermeidung von Schwachstellen ausgerüstet wird.

Die durchgeführten Tests und Ergebnisse werden ab Seite 17 dargestellt.

A3 Übersicht aller Findings

In der folgenden Übersicht sind die Bedrohungen und Schwachstellen aufgeführt, die das Gefährdungspotential info, niedrig, mittel, hoch oder kritisch besitzen.

Schwachstelle	[niedrig]	[mittel]	[hoch]
Datenvalidierung			
-			
Authentisierung			
-			
Session-Management			
-			
Zugriffsschutz			
-			
Konfiguration und Deployment			
-			
Logik			
-			
Informationsabfluss			
-			
Kryptographie			
-			

A4 Assessment Details

A4.1 Getestete Applikation

Die Sicherheitsüberprüfung erfolgte auf den folgenden Webapplikationen / der folgenden Webapplikation:

Akronym	URL	IP
de-Consline	https://cims1.consline.com	86.110.76.245

A4.2 Testzeitraum

Die Tests wurden am 10.09.2014, 11.09.2014 und 15.09.2014 durchgeführt.

A4.3 Testbenutzer

Die folgenden Testbenutzer wurden vom Kunden bereitgestellt:

Benutzername	Rolle
cu149014	Power user, CIMS
cu149043	Power user, CIMS, Status-Editor, Alerts
cu149072	Power user

Die Testbenutzer sollten nach dem Ende des Penetrationstestes wieder gelöscht oder deaktiviert werden. SecureNet übernimmt keine Verantwortung für Schäden, die durch weiterhin vorhandene Testbenutzer entstehen.

B. Methodik und Bewertung

B1.1 Beschreibung der Methodik

B1.1.1 Testverfahren (Blackboxtest)

Die Anwendung wurde sowohl ausgiebigen manuellen Tests, als auch automatischen Tests unterzogen. Zum Einsatz kamen verschiedene frei verfügbare Tools. Die Tests mit den Tools haben wir auf die Bereiche beschränkt, in denen die Stärken der Tools liegen. Die manuellen Tests erstrecken sich auf alle untersuchten Bereiche und erfassen insbesondere auch die Businesslogik der Anwendung.

B1.1.2 Art der Schwachstellenuntersuchung

Wir betrachten die im folgenden Bild gezeigten Ebenen (Aspekte) 0 bis 5 der Sicherheit der Webanwendungen:

	Ebene	Inhalt (Beispiele)
5	**Semantik**	Schutz vor Täuschung und Betrug - Informationen ermöglichen Social-Engineering-Angriffe - Gebrauch von Popups u. ä. erleichtern Phishing-Angriffe - Keine Absicherung für den Fall der Fälschung der Website
4	**Logik**	Absicherung von Prozessen und Workflows als Ganzes - Verwendung unsicherer E-Mail in einem ansonsten gesicherten Workflow - Angreifbarkeit des Passworts durch nachlässig gestaltete „Passwort vergessen"-Funktion - Die Verwendung sicherer Passwörter wird nicht erzwungen
3	**Implementierung**	Vermeiden von Programmierfehlern, die zu Schwachstellen führen - Cross-Site Scripting - SQL-Injection - CSRF (Session-Riding)
2	**Technologie**	Richtige Wahl und sicherer Einsatz von Methoden - unverschlüsselte Übertragung sensitiver Daten - Authentisierungsverfahren, die dem Schutzbedarf nicht angemessen sind - Ungenügende Entropie von Tokens

1	**System**	Absicherung der auf der Systemplattform eingesetzten Software - Fehler in der Konfiguration des Webservers - Bekannte Schwachstellen in den eingesetzten Softwareprodukten - Mangelnder Zugriffsschutz in der Datenbank
0	**Netzwerk & Host**	Absicherung von Host und Netzwerk

Abbildung B1.1.2: Ebenen der Web Application Security

Ebene 0 – Netzwerk und Host

Die Websicherheit steht auch in Abhängigkeit zur Sicherheit von Netzwerk, Hardware und Host. Wir betrachten die Berührungspunkte der beiden Ebenen, soweit dies erforderlich ist.

Ebene 1 – Systemebene

Ebene 1 beinhaltet all jene Software, die eine Webanwendung benötigt, um überhaupt laufen zu können. Dazu gehören der Webserver und der Applikationsserver, aber auch die Datenbank und beteiligte Backend-Systeme. Alle diese Komponenten müssen bei der Betrachtung der Sicherheit einer Webanwendung mit einbezogen werden. Eine Webanwendung A, die frei von Sicherheitsmängeln programmiert ist, ist trotzdem unsicher, wenn die von ihr verwendete Datenbank über einen anderen Kanal, etwa den nicht ausreichend abgesicherten und für ‚Innentäter' zugänglichen Direktzugriff per Database-Client manipulierbar ist und diese Manipulation im Web von einem Angreifer ausgenutzt werden kann.

Ein besonderes Feld sind die sog. Known-Vulnerabilities, die wir ebenfalls der Systemebene zuordnen.

Ebene 2 – Technik

In diesem Bereich geht es um die Auswahl der für den jeweiligen Zweck und Schutzbedarf richtigen Technik – und um deren richtigen Einsatz. Eine Webanwendung, die sensible Daten unverschlüsselt über das Internet transferiert, setzt nicht die richtige Technik ein. Und eine Webanwendung, die Passwörter zwar verschlüsselt, dies aber mit einem zu kurzen Schlüssel tut, setzt die richtige Technik falsch ein. Die erste Anwendung ist gegen Ausspähen auf dem Übertragungswege nicht geschützt, die andere nicht ausreichend gegen Passwort-Cracking. Beide sind also unsicher, auch wenn sie im Programmcode keine Sicherheitslücken enthalten.

Ebene 3 – Implementierung

Die Implementierungsebene ist die offensichtliche Ebene der Web Application Security. Dies ist der Bereich, in dem unbeabsichtigte Programmierfehler (Bugs) auftreten und zu Sicherheitsproblemen führen oder aber fehlerhafte Programmierung, wie nicht vorhandene oder ungenügende Datenvalidierung, stattfinden.

Logik (Ebene 4) und Semantik (Ebene 5)

Diese beiden Ebenen sind diejenigen, die gegenwärtig im Sicherheitskontext noch kaum beachtet werden. Dabei sind sie von großer Wichtigkeit – und werden es im Zeitalter von Phishing und Identitätsdiebstahl immer stärker werden – wenn man die Sicherheit von Webanwendungen nicht allein als den Schutz des Servers vor Eindringversuchen versteht, sondern sie umfassend begreift. Eine Webanwendung ist sicher, wenn sie selbst mitsamt dem sie beherbergenden System sicher ist, und wenn sie die Benutzer und deren vertrauenswürdigen Daten vor Schaden bewahrt. Der Anbieter einer Webanwendung trägt also nicht nur die Verantwortung für sein eigenes System, sondern auch für alle an der Nutzung Beteiligten. Auf den Ebenen der Logik und der Semantik kommt dieser letzte Aspekt ganz besonders zum Tragen.

Ebene 4 – Logik

Diese Ebene betrifft die Logik der Abläufe in einer Anwendung – die Anwendungs- und Business-Logik – mit dem Benutzer. Ist diese zu ‚zweckorientiert' implementiert, d. h. ist die Möglichkeit, dass sie anders als beabsichtigt genutzt wird, zu wenig berücksichtigt, dann ist häufig eine Angreifbarkeit gegeben. Sind z.B. Mechanismen eingebaut, welche bei unsachgemäßer Bedienung (Zustände, die nur unter Umgehung der Browserfunktionalität möglich sind und damit eine missbräuchliche Verwendung eindeutig anzeigen) den Benutzer ausloggen oder andere Formen des Schutzes betreiben?

Ebene 5 – Semantik

Die semantische Ebene der Web Application Security umfasst inhaltliche und die Kommunikation betreffende Aspekte. Dies betrifft die Art der Informationen, die dem Benutzer gegeben werden, wie ihm Inhalte präsentiert werden und wie mit ihm umgegangen wird. Dieser Bereich kann in seiner Betrachtung selten auf eine einzelne Anwendung beschränkt bleiben. Er ist in der Regel vielmehr website- oder unternehmensübergreifend zu definieren und ähnlich, wie die Vorgabe einer CI von allen Kommunikationsmedien einzuhalten ist, von allen Anwendungen zu befolgen.

Ein fehlerhafter Umgang mit den semantischen Aspekten der Web Application Security erleichtert insbesondere Angriffe auf den Benutzer, was wiederum dem Vertrauen des Benutzers in die Anwendung und das Unternehmen sowie das Web insgesamt schadet. Zu derartigen Angriffen zählen Social-Engineering, Phishing, Identitätsdiebstahl, Täuschung, Fälschung, Betrug, sowie Aufbrechen des Datenschutzes und des Schutzes der Privatsphäre.

B1.2 Bewertungsschema

B1.2.1 Gefahrenpotential

Die durchgeführten Tests wurden jeweils mit einem Gefahrenpotential bewertet. Das Gefahrenpotential ist ein abstraktes Maß für die Klassifizierung einer Schwachstelle (Vulnerability) und der durch sie entstehenden Bedrohung (Threat).

Das Gefahrenpotential wird zu jeder getesteten Schwachstelle in der rechten Spalte genannt. Wir unterscheiden:

[OK**]** die genannte Schwachstelle liegt nicht vor

[kritisch] (siehe unten Gefährdungs-Matrix)

[hoch] (siehe unten Gefährdungs-Matrix)

[mittel] (siehe unten Gefährdungs-Matrix)

[niedrig] (siehe unten Gefährdungs-Matrix)

[Info] hierbei handelt es sich um Informationen, nicht um eine Schwachstelle

Das Gefahrenpotential ist kein Maß für das Risiko, das eine solche Schwachstelle darstellt. Das Risiko (= Schadenshöhe * Eintrittswahrscheinlichkeit), welches dieses Gefahrenpotential darstellt, wird hier nicht bewertet, da zu dessen Beurteilung weitere Informationen (z.B. Schadenspotential) vom Auftraggeber nötig sind.

B1.2.2 Eintrittswahrscheinlichkeit

Die Eintrittswahrscheinlichkeit ist im Wesentlichen von folgenden Faktoren bestimmt:

- wie einfach ist die Schwachstelle zu finden (Visibility)?
- wie einfach ist die Schwachstelle auszunutzen (Exploitability)?
- wie bekannt ist die Schwachstelle (Publicly Known)?
- welche technischen Kenntnisse sind zum Ausnutzen nötig?
- welche Zugriffsmöglichkeiten bestehen (remote vs. lokal)?
- welchen Typs ist die Schwachstelle (Vulnerability Type)?

Vereinfacht kann man sagen, dass die Eintrittswahrscheinlichkeit von der Schwierigkeit des Angriffs abhängt.

B1.2.3 Schadenspotenzial

Das Schadenspotenzial ist vor allem die Folge, die sich aus der Ausnutzung der Schwachstelle ergeben kann:

- Verlust der Verfügbarkeit (Availability)
- Verlust der Vertraulichkeit (Confidentiality)
- Verlust der Vollständigkeit, bzw. Unversehrtheit (Integrity)
- Verlust von Schutzmechanismen
- Erweiterung der Zugriffberechtigungen.

Das konkrete Schadenspotenzial kann i. A. nur vom Auftraggeber bestimmt werden.

B1.2.4 Risiko-/Gefährdungs-Matrix

Es ist allgemein üblich das Risiko nach der Formel:

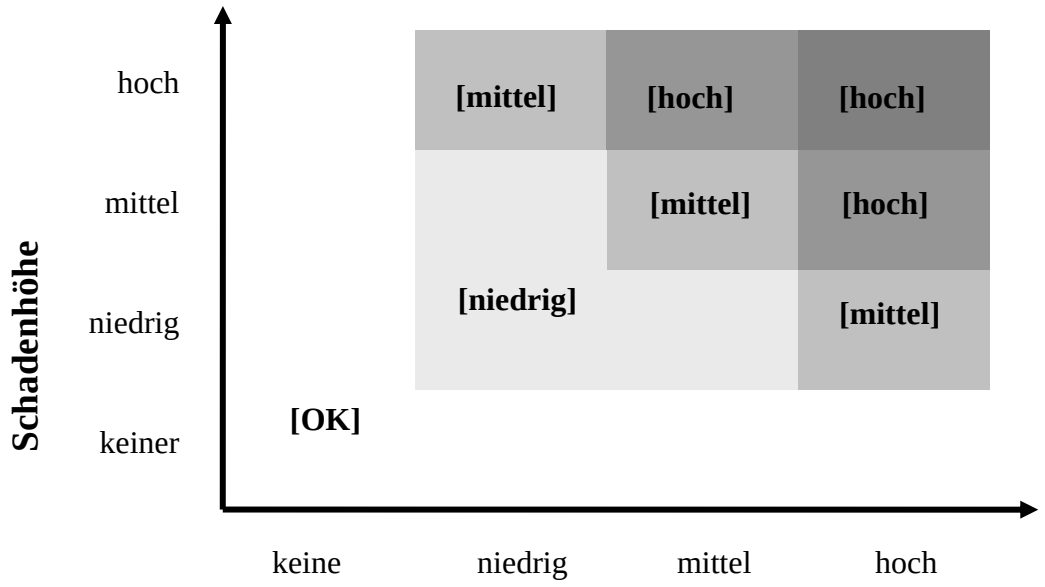
Risiko = Schadenspotenzial * Eintrittswahrscheinlichkeit

zu bestimmen. Aus den Beschreibungen zur Eintrittswahrscheinlichkeit und zum Schadenspotenzial wird ersichtlich, dass sich beide Faktoren ebenfalls aus mehreren Werten zusammensetzen. Z. B. hat eine Schwachstelle, die allgemein bekannt ist und für die bereits fertige Exploits existieren, eine hohe Eintrittswahrscheinlichkeit. Ist dadurch dann z.B. die Übernahme des Systems möglich oder können vertrauliche Daten eingesehen werden, dann ist auch das Schadenspotenzial hoch.

Um das Produkt aus Eintrittswahrscheinlichkeit und Schadenspotenzial mit ihren jeweils eigenen Faktoren zu berechnen, gibt es verschiedene Modelle (CVSS, DREAD, STRIDE, OCTAVE), die die einzelnen Faktoren unterschiedlich gewichten.

Um eine genaue Risikoabschätzung zu erhalten, müsste das Risiko mit der subjektiven Gewichtung des Auftraggebers nach den bekannten Modellen berechnet werden. Diese muss für jede in diesem Bericht aufgeführte Schwachstelle einzeln erfolgen.

Um die Schwachstellen einfach und schnell zu klassifizieren, wird für das Gefahrenpotential folgende vereinfachte Matrix verwendet:



Aus dieser Beurteilung kann auch eine Handlungspriorität abgeleitet werden:

[kritisch] – es besteht sofortiger Handlungsbedarf; der Applikation, der Datenbank oder dem System kann schwerwiegender Schaden zugefügt werden; ggf. ist das System abzuschalten

[hoch] – es besteht dringender Handlungsbedarf, die Webanwendung/das System und/oder die Daten sind bestandsgefährdet;

[mittel] – es besteht Handlungsbedarf, die Webanwendung/das System muss z.B. durch einspielen von Sicherheitsupdates oder durch zusätzliche Sicherungsmaßnahmen (z.B. WAF) abgesichert werden.

[niedrig] – der Handlungsbedarf liegt im Ermessen des Auftraggebers.

Hinweis

Obwohl wir bei der Bewertung objektive Maßstäbe bzw. als allgemeingültig anerkannte Grundsätze zugrunde gelegt haben und nach bestem Wissen und Gewissen versucht haben, die Vor- und Nachteile gegeneinander abzuwägen, kann eine solche Bewertung immer nur subjektiv sein. Uns nicht bekannte Details oder unterschiedliche Interpretation der Sicherheitsanforderungen können zu einer anderen als der hier jeweils abgegebenen Bewertung führen. Daher ist der Auftraggeber angehalten, anhand der gegebenen Informationen eine eigene Bewertung vorzunehmen, bzw. die Dringlichkeit der Behebung einzelner Mängel selbst zu beurteilen.

Gerne stehen wir dabei beratend zur Seite.

B1.3 Aufbau der Testergebnisse

Für jede Schwachstelle gibt es jeweils ein eigenes Unterkapitel, wobei deren Überschriften Auskunft über die Art der Schwachstelle geben.

Eine solche Überschrift sieht dann z.B. so aus:

- n.m Frame-Spoofing
- **n.m** ist die Unterkapitelnummer
- **Frame-Spoofing** ist die genaue Schwachstellenbezeichnung

Jedes dieser Unterkapitel ist dann gegliedert in:

- **Bedrohung**
  Die durch diese Schwachstelle vorhandene Bedrohung wird skizziert.
- **Angriffsszenario / Bedrohung** (optional)
  Falls erforderlich, schildern wir hier mögliche Szenarien der Ausnutzung der gefundenen Schwachstelle, um dem Leser die Möglichkeit zu geben, das Risiko zu bewerten. Hierbei legen wir einen Worst-Case Ansatz zugrunde, d. h. schildern im Zweifel das bedrohlichere Szenario. Diese Schilderung ist unabhängig von der Eintrittswahrscheinlichkeit. Eine Bewertung der Eintrittswahrscheinlichkeit muss separat erfolgen.
- **Testfälle**
  es folgen die durchgeführten Tests, die wie folgt beschrieben sind:

n.m.o Reflected Cross-Site-Scripting stark verbreitet [hoch]

- **Reflected Cross-Site-Scripting stark verbreitet** ist eine knappe Zusammenfassung der gefundenen Schwachstelle
- **hoch** ist die Bewertung des Gefahrenpotentials

Die Gliederung sieht hier wie folgt aus:

- **Beobachtung**
  Der Test und die Beobachtung der Reaktion der Anwendung werden

beschrieben. Damit soll (1) der Leser in die Lage versetzt werden, den Angriff zu verstehen, um das Risiko selbst besser beurteilen zu können, und (2) der Angriff nachvollziehbar werden. Es ist nicht immer möglich, (2) zu gewährleisten, da für das Nachstellen u. U. eine umfangreichere Vorbereitung oder zeitliche Korrelation mit anderen Bedingungen erfüllt sein muss.

— **Beispiel** (optional)
Falls es mehrere Möglichkeiten gibt diese Schwachstelle auszunutzen, dann werden diese hier detailliert aufgelistet, während unter Beobachtung (siehe oben) dies nur allgemein beschrieben ist.

— **Bemerkung** (optional)
Hier werden Besonderheiten und/oder bestimmte Bedingungen zur gefundenen Schwachstelle beschrieben, z.B. wenn diese nicht reproduzierbar auftrat oder nur mit bestimmten Browsern usw.. Oft sind solche Anmerkungen zur Beurteilung der Bewertung (siehe oben) wichtig.

— **Maßnahme** (optional)
Es ist zwischen grundlegenden Maßnahmen und individuellen Maßnahmen zu unterscheiden. Grundlegende Maßnahmen sind meistens bedrohungsbezogen, wohingegen individuelle Maßnahmen eher schwachstellenbezogen sind. Grundlegende Maßnahmen lassen sich unabhängig von der jeweiligen Anwendung formulieren und haben universelle Gültigkeit.

Bei Webanwendungen, die ohne Berücksichtigung der grundlegenden Schutzmaßnahmen entwickelt worden sind, kann die nachträgliche Anwendung grundlegender Maßnahmen zu unvertretbar hohem Aufwand führen. Hinzu kommt, dass eine Anwendung, die die beschriebene Maßnahme nicht umgesetzt hat, trotzdem nicht zwingend anfällig für einen Angriff sein muss, weil dies bewusst durch andere Maßnahmen sichergestellt worden ist oder weil es sich eher zufällig so verhält. In diesem Fall wäre es nicht angebracht, die Umsetzung der genannten Maßnahme zu fordern.

Maßnahmen aus Sicht des Penetrationstests sind daher zumeist individuelle schwachstellenbezogene Maßnahmen, die erst bei Entdeckung einer Schwachstelle relevant werden. Dabei kommt häufig zur Behebung der Schwachstelle nicht genau eine Maßnahme in Betracht, sondern eine ganze Palette von möglichen Maßnahmen. Die Entscheidung über die richtige Maßnahme ist für jeden Anwendungsfall individuell zu treffen und abhängig von den softwaretechnischen Randbedingungen, der verwendeten Technik, der Art der Realisierung, den Realisierungskosten usw..

Wir nennen an dieser Stelle die grundlegenden Maßnahmen. Sie gelten i. d. R. für alle im Folgenden aufgeführten Schwachstellen.

— Eine tatsächliche Bewertung und Entscheidung darüber, wie eine Schwachstelle zu beseitigen ist, muss der Auftraggeber treffen oder sie werden in einem abschließenden Workshop zusammen mit den Entwicklern bzw. Anwendungsverantwortlichen getroffen.

B1.4 Urheberrecht

Die in diesem Dokument enthaltenen allgemeinen Beschreibungen von Schwachstellen, Angriffsszenarien, Maßnahmen und Bewertungsverfahren sowie die

C. Testergebnisse im Detail

C1 Datenvalidierung

Die meisten Schwachstellen in Webanwendungen haben eine ungenügende Datenvalidierung (oft als Input-Datenvalidierung bezeichnet) als Ursache.

Unter Datenvalidierung sind alle Daten und zugehörigen Prüfungen zu verstehen, die die Daten bearbeiten, welche von außen in ein System kommen. Dazu gehören nicht nur die unmittelbaren Benutzereingaben, sondern auch die Daten, die von Drittsystemen (z.B. aus Datenbanken) kommen. Weiter muss jedes System sicherstellen, dass die Daten, die an Drittsysteme zur Verarbeitung weitergegeben werden, keinen schadhaften Code enthalten.

Korrekterweise muss daher zwischen Input- und Output-Datenvalidierung unterschieden werden. Bei den folgenden einzelnen Schwachstellen und Bedrohungen wird ggf. auf diesen Unterschied aufmerksam gemacht.

C1.1 Cross-Site-Scripting

Bedrohung

Cross-Site-Scripting (XSS) bezeichnet allgemein das Ausnutzen einer Schwachstelle, indem Informationen aus einem Kontext, in dem sie nicht vertrauenswürdig sind, in einen anderen Kontext eingefügt werden, in dem sie als vertrauenswürdig eingestuft sind.

XSS-Schwachstellen können vielfältig ausgenutzt werden, z.B.:

- Session-Hijacking (Eindringen in den Account eines anderen Benutzers unter Umgehung des Logins).
- Website-Spoofing (Fälschung einer Website) zum Phishing (Stehlen von Logindaten und persönlichen Daten) oder anderen Täuschungs- und Betrugsformen. Hierbei wird weiterhin die original URL der Website angezeigt und auch das Zertifikat bei einer https-Verbindung bezeugt weiterhin die Echtheit der Site – es bleibt unverletzt.
- Ausspionieren oder Manipulation der Daten auf dem System des Benutzers.
- Angriff auf den Browser um die vollständige Kontrolle des Clientsystems zu erhalten durch frei verfügbare Angriffstools (z.B. BeEF).

Die Ausführung eines Angriffs geschieht am einfachsten durch Versenden einer E-Mail. Bei bestimmten Einstellungen des E-Mail-Clients braucht der Benutzer dabei noch nicht einmal auf einen Link zu klicken.

Wegen der aktuellen Verbreitung von Phishing-Angriffen und der Möglichkeit des Session-Hijacking bewerten wir XSS-Schwachstellen in der Regel mit dem Gefahrenpotential [hoch]. Wenn die Ausnutzung jedoch nur unter bestimmten Umständen möglich ist, wird das Gefahrenpotential reduziert. Eine Behebung ist trotzdem dringend anzuraten, da nicht ausgeschlossen werden kann, dass es doch Möglichkeiten für eine einfache Ausnutzung gibt.

Beobachtung

Hinweise auf Cross-Site-Scripting wurden nicht gefunden.

C1.2 Website-Spoofing

Bedrohung

Website-Spoofing bezeichnet allgemein die Manipulation einer Webseite. Dies kann eine Manipulation der Seite auf dem Webserver selbst sein, oder eine Seite in einem vorgelagerten Caching-/Proxy-Server, oder mittels HTML-Injection oder Cross-Site-Scripting im Browser.

Das wichtigste Angriffsszenario für Website-Spoofing ist das Phishing. Mit den hier beschriebenen Problemen ist es möglich, dem Benutzer eine gefälschte Loginseite zu präsentieren, die identisch mit der echten ist, die Logindaten jedoch an den Angreifer schickt. Dabei bleibt der Servername im URL (Adresszeile des Browsers) erhalten und auch das Zertifikat, welches die Authentizität des Servers bezeugt, bleibt unverletzt.

Beobachtung

Hinweise auf Website-Spoofing wurden nicht gefunden.

C1.3 SQL-Injection

Bedrohung

SQL-Injection bezeichnet das Ausnutzen einer Schwachstelle, indem SQL-Abfragen über Eingabeparameter so manipuliert werden, dass ein unberechtigter Zugriff auf eine Datenbank oder über die Datenbank auf das System entsteht.

Eine ausnutzbare SQL-Injection Schwachstelle liegt häufig auch dann vor, wenn die Anwendung die ausgelesenen Daten gar nicht in die Seite weitereicht, sondern lediglich unterschiedlich reagiert, abhängig davon, ob die Eingabe intern einen Fehler erzeugt hat oder eine bestimmte Bedingung erfüllt wurde. Mit geeigneten Tools kann dann im Brute-Force-Verfahren auf die Daten geschlossen werden und mittels Durchführung sehr vieler systematischer Requests Inhalte der Datenbank, bis hin zu Benutzernamen und Passworten, ermittelt werden.

Beobachtung

Hinweise auf SQL-Injection wurden nicht gefunden.

C1.4 Command-Injection

Bedrohung

Command-Injection bezeichnet das Ausnutzen einer Schwachstelle, indem Parameter bei System- und Shell-Aufrufen so manipuliert werden, dass ein unberechtigter Zugriff auf das Betriebssystem und/oder dessen Dateien entsteht.

Beobachtung

Hinweise auf Command-Injection wurden nicht gefunden.

C1.5 Path-Traversal

Bedrohung

Angriffstechnik, die durch Manipulation des Pfades in der URL versucht auf Verzeichnisse und/oder Seiten (Dateien) zuzugreifen, die außerhalb des öffentlichen Bereichs des Webservers (DocumentRoot) liegen.

Siehe auch C5.4 Directory-Indexing.

Beobachtung

Hinweise auf Path-Traversal wurden nicht gefunden.

C1.6 Parameter-Manipulation/Tampering

Bedrohung

Parameter-Tampering bezeichnet allgemein die Manipulation von Parametern, die zum Server gesendet werden.

Viele der in diesem Dokument getesteten Schwachstellen können mittels Parameter-Tampering ausgenutzt werden, wir haben diese jeweils unter dem spezielleren Begriff eingeordnet. In diesem Abschnitt werden nur noch die Beobachtungen aufgeführt, die sich nicht weiter einordnen lassen.

Beobachtung

Hinweise auf weitere Parametermanipulationen wurden nicht gefunden.

C2 Authentisierung

C2.1 Enumeration von Benutzername oder Passwort

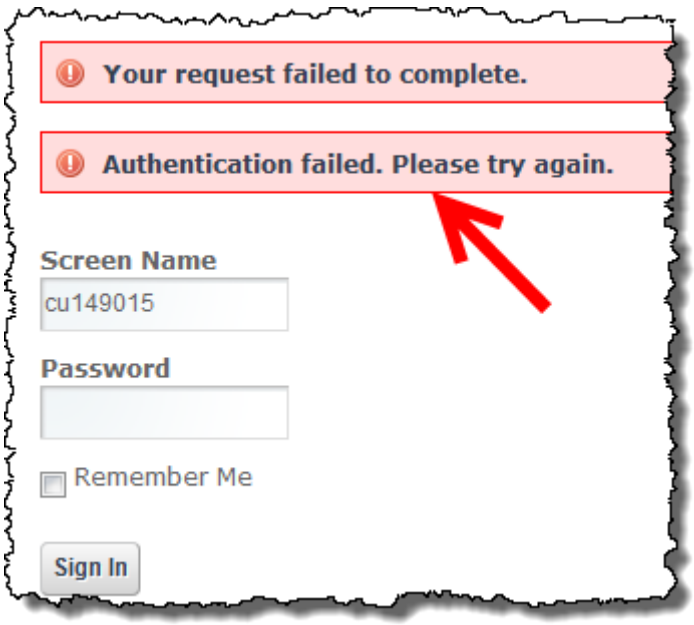
Bedrohung

Ein Benutzername oder ein Passwort könnte durch einfaches Durchprobieren aller möglichen Kombinationen der für Benutzernamen und Passwort erlaubten Zeichen erraten werden.

Wenn eine Anwendung bei Angabe von falschen Anmeldedaten (Username / Passwort) in der Antwortseite mitteilt, welcher der eingegebenen Daten falsch ist, dann erleichtert sie solche Brute-Force-Angriffe, d.h. erhöht die Chance durch systematisches Durchprobieren diese zu ermitteln.

Beobachtung

Die Anwendung gibt bei fehlerhafter Anmeldung keine Informationen darüber, ob der Benutzername oder das Passwort falsch sind. Es kommt immer die gleiche Fehlermeldung.



C2.2 Passwort-Cracking

Bedrohung

Wenn die Anwendung keinen Mechanismus zur Überwachung fehlerhafter Login-Versuche vorsieht, hat ein Angreifer die Möglichkeit, potentiell beliebig viele Login-Versuche durchzuführen. Er wird dann nicht daran gehindert, mithilfe entsprechender Tools systematisches Passwort-Cracking zu betreiben.

Eine strenge Passwort-Policy alleine ist in der Regel kein ausreichender Schutz, da Benutzer dazu tendieren, keine wirklich zufälligen Passwörter zu verwenden, sondern

leicht zu merkende Zeichenketten. Das verlangte Sonderzeichen oder die Zahl wird dann einfach noch hinten angestellt. Die Entropie solcher Passwörter ist dann um einige Größenordnungen kleiner als die theoretisch berechnete und Wörterbuch-basierte Angriffstechniken (Verwendung von Wortlisten) werden anwendbar.

Beobachtung

Die Anwendung verhindert Passwort-Cracking durch Sperrung des betroffenen Benutzers nach dem dritten Fehlversuch, siehe auch C6.3.

C2.3 Passwort-Struktur, -Qualität

Bedrohungen

Ein Passwort darf nicht nach einer automatisch ermittelbaren Struktur aufgebaut sein.

Triviale Passwörter sind nicht zu erlauben. Dazu zählen auch solche, die sich in einschlägigen Wörterbüchern häufig verwendeter Passwörter befinden.

Beobachtung

Die Passwort-Policy der Anwendung ist als ausreichend anzusehen. Unsere Prüfungen haben ergeben, dass diese bei der Vergabe eines neuen Passworts durch den Benutzer auch überprüft wird.

C3 Session-Management

Session-Management beschreibt allgemein die technischen und logischen Voraussetzungen, die notwendig sind, um Sitzungen, Aktionen und Transaktionen in einer Umgebung mit zustandslosen Verbindungen zu betreiben.

C3.1 Transportmedium

Bedrohung

Daten wie die SessionID unterliegen einem erhöhten Schutzbedarf, da sie meist die einzige Information sind, mit der die Anwendung erkennen kann, mit welchem Benutzer sie es zu tun hat. Insbesondere ist mit der SessionID meist die Authentisierungsinformation verknüpft, d. h. dass jeder, der in den Besitz der SessionID gelangt, diese nutzen kann, um sich in der Session, also innerhalb des Benutzerkontos, zu bewegen.

Für den Transport der SessionID stehen prinzipiell folgende Wege zur Verfügung:

- Cookie: SessionID ist ein Cookiewert, das Cookie wird im HTTP-Header übertragen
- URL-Rewriting: SessionID steht im Link
- Form-Parameter: SessionID steht in einer Form-Variablen (HIDDEN-Field)

Darüber hinaus kann die Authentisierungsinformation nach dem Login des Benutzers vermittels dieser Techniken aufrechterhalten werden:

- Basic Authentication, Digest Authentication
- SSL-Clientzertifikat

Am häufigsten sind Cookies anzutreffen, gefolgt von URL-Rewriting. Beides wird von den gängigen Application-Frameworks unterstützt. Die Form-Parameter haben an Bedeutung verloren, da sie von vielen Frameworks nicht unterstützt werden.

Abhängig vom Transportverfahren sind unterschiedliche Angriffe möglich und dementsprechend auch unterschiedliche Sicherheitsmaßnahmen zu treffen.

Beobachtung

Die getestete Anwendung verwendet als Transportmedium Cookies.

Es wurde stichprobenartig geprüft, ob der Anwendung URL-Rewriting durch Einfügen der SessionID in den URL aufgezwungen werden kann. Dies war nicht der Fall.

C3.2 Logout-Funktionalität

Bedrohung

Eine Anwendung, die mit benutzerbezogenen Sessions arbeitet, muss eine Logout-Funktion anbieten, so dass der Benutzer seine Session jederzeit beenden kann. Fehlt eine solche Funktion, erhöht sich die Anfälligkeit der Anwendung für eine Reihe von Angriffen auf die Session, da das Zeitfenster für einen Angriff nicht minimal gehalten wird.

Beobachtung

Die Anwendung besitzt eine „Logout"-Funktion, diese invalidiert die Session auf Serverseite und löscht den Session-Cookie Inhalt auf Clientseite.

C3.3 Logout durch Timeout bei Inaktivität

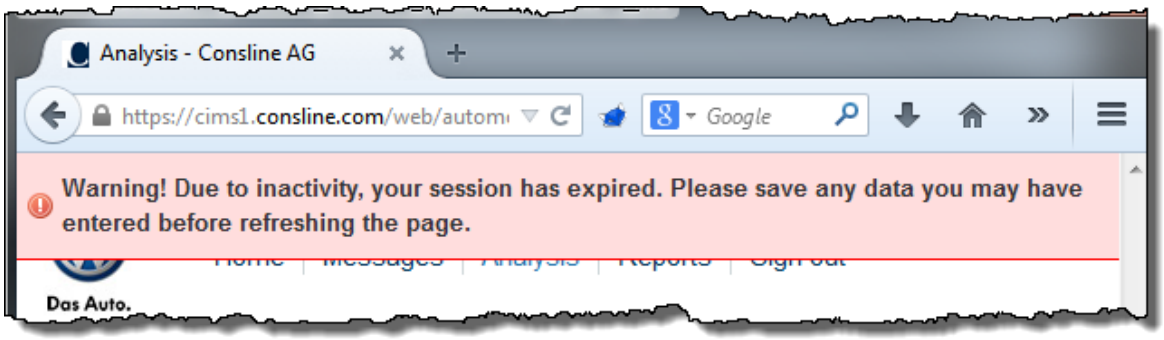
Bedrohung

Wenn Anwendungen die Session bei Inaktivität nicht beenden, erhöht sich potentiell die Anfälligkeit für Angriffe auf die Session (Hijacking, Replay, Riding) und unbeabsichtigte Ausspähung der Daten am Client durch Dritte, da das Zeitfenster für einen Angriff nicht minimal gehalten wird.

Beobachtung

Die Anwendung hat ein automatisches Logout nach einer hinreichend kurzen Zeit der Inaktivität implementiert.

Die Timeout-Zeit wurde nicht ermittelt.



C3.4 Session-Fixation

Bedrohung

Session-Fixation ist eine Angriffstechnik, die dem Benutzer einer Anwendung eine bestehende und vom Angreifer initiierte Session vorgibt. Der Benutzer authentisiert dann diese vorgegebene Session und verschafft dem Angreifer damit Zugang zu seinem Benutzerkonto.

Beobachtung

Das Session-Cookie wird nach dem Login und Logout von der Anwendung im Browser des Clients invalidiert. Somit ist die Anwendung gegen Session-Fixation geschützt.

C3.5 CSRF

Bedrohung

Als Cross-Site Request Forgery (CSRF) oder Session-Riding bezeichnet man die Möglichkeit, Aktionen in einer Anwendung (in einer Session eines legitimierten Benutzers) auszuführen, ohne dass der Benutzer diese selbst aktiv oder bewusst veranlasst.

Solche Links können damit auf irgendeiner Webseite einem Benutzer untergeschoben werden. Klickt der Benutzer auf diesen Link oder wird der Link automatisch ausgeführt und ist der Benutzer in diesem Moment in der Webanwendung noch eingeloggt, so wird der entsprechende Befehl ohne Wissen des Benutzers ausgeführt. Einzige Voraussetzung für das Funktionieren des Links ist eine gültige Session des Benutzers.

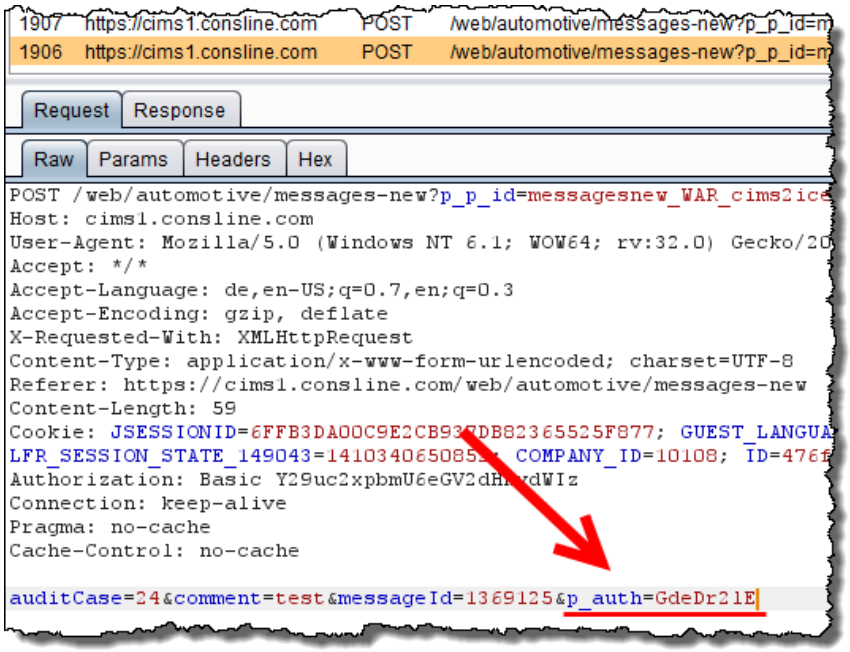
Mögliches Angriffsszenario

Ein Angreifer sendet einem Benutzer eine E-Mail mit den oben beschriebenen Links. Klickt der Benutzer auf den Link, so kommt der untergeschobene Request dadurch entweder direkt zur Ausführung oder der Benutzer wird auf eine Seite geführt, auf der der Request dann indirekt zur Ausführung gelangt.

Eine ausführliche Beschreibung dieser Schwachstelle findet sich in /23/ und /24/.

Beobachtung

Die Anwendung ist vor Cross-Site Request Forgery (CSRF) Angriffen geschützt. Die einzige Zustandsändernde Operation, das Hinterlassen eines Kommentares, verwendet zusätzlich den Parameter `p_auth`, der ein Token mitliefert.



C3.6 Zufälligkeit der SessionID

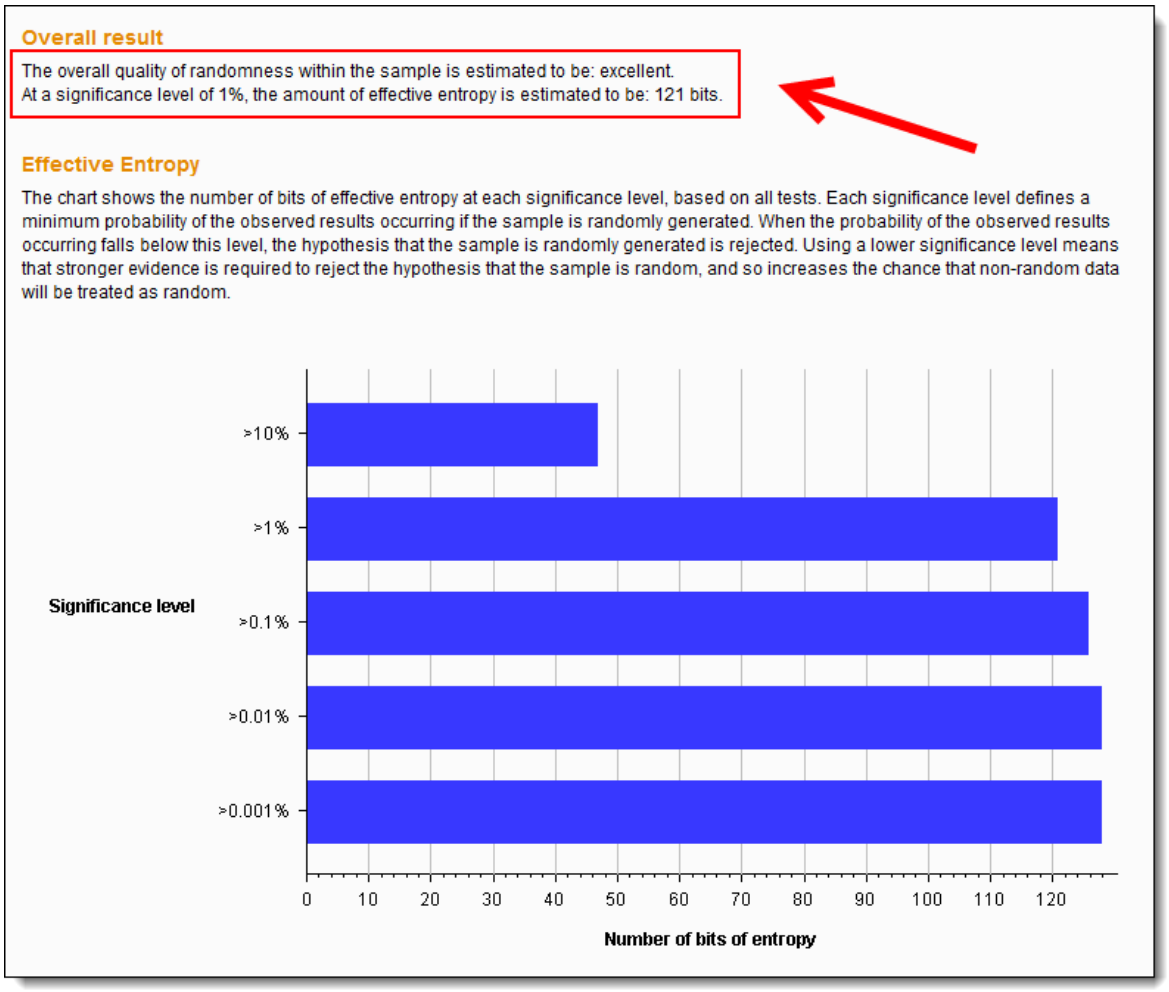
Bedrohung

Wenn die SessionID die einzige Information ist, anhand derer die Anwendung erkennt, mit welchem Benutzer sie es zu tun hat, dann unterliegt sie einem erhöhten Schutzbedarf. Insbesondere ist mit der SessionID meist die Authentifizierungsinformation verknüpft, d.h. dass jeder, der in den Besitz der SessionID kommt, diese nutzen kann, um eine Session fortzusetzen.

Daher muss die SessionID eine dem Schutzbedarf der Anwendung entsprechende Güte besitzen. SessionIDs dürfen z.B. nicht einfach „hochgezählt“ werden, oder nach einem leicht zu durchschauenden Schema aufgebaut sein.

Beobachtung

Die Zufälligkeit der SessionID wurde mit einem Tool überprüft und ist ausreichend.



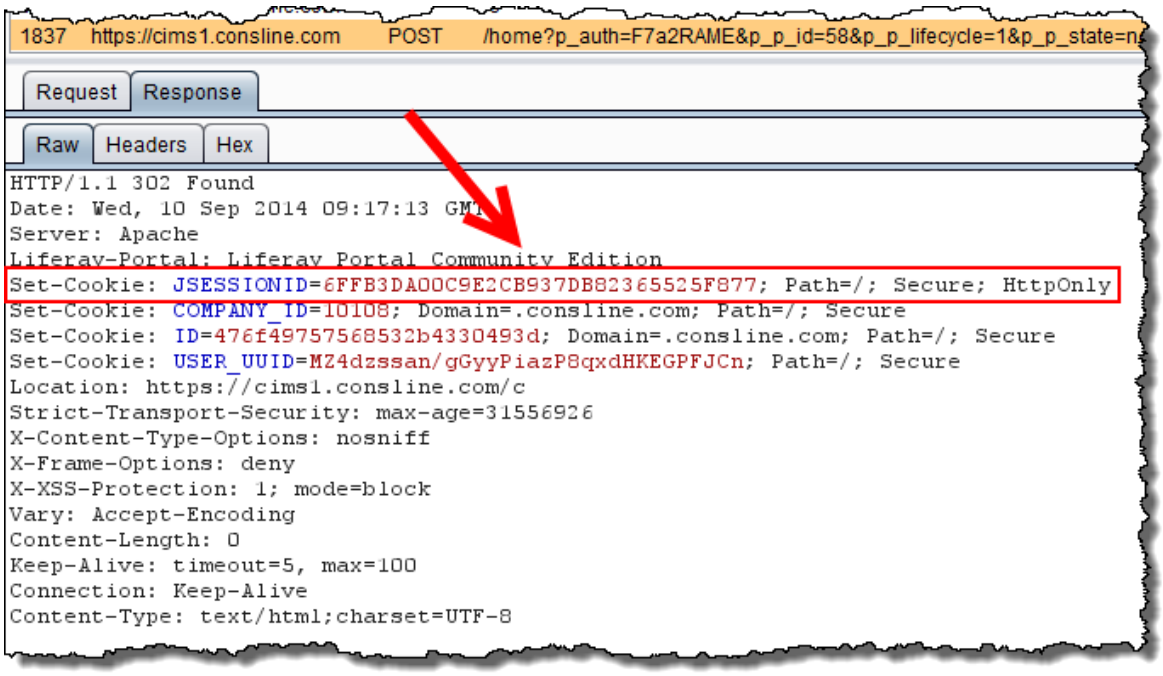
C3.7 Gültigkeit des SessionID-Cookies

Bedrohung

Cookies besitzen einen bestimmten Gültigkeitsbereich. Dieser wird beim Setzen des Cookies definiert und dem Browser in Form von Cookie-Parametern (`domain`, `path`, `expire`, `httponly`, `secure`) mitgeteilt. Der Browser entscheidet dann anhand dieser Parameter, ob das entsprechende Cookie an die Anwendung geschickt wird oder nicht. Die Anwendung hat keine Möglichkeit zu erkennen, unter welchen Bedingungen ein Cookie gesendet wurde.

Beobachtung

Es konnten keine Hinweise auf eine unzureichende Cookie-Konfiguration gefunden werden. Die Flags `secure` und `httpOnly` sind beide für das Session-Cookie gesetzt.


1837 https://cims1.consline.com POST /home?p_auth=F7a2RAME&p_p_id=58&p_p_lifecycle=1&p_p_state=n
Request
Response
Raw
Headers
Hex
HTTP/1.1 302 Found
Date: Wed, 10 Sep 2014 09:17:13 GM
Server: Apache
Liferay-Portal: Liferay Portal Community Edition
Set-Cookie: JSESSIONID=6FFB3DA00C9E2CB937DB82365525F877; Path=/; Secure; HttpOnly
Set-Cookie: COMPANY_ID=10108; Domain=.consline.com; Path=/; Secure
Set-Cookie: ID=476f49757568532b4330493d; Domain=.consline.com; Path=/; Secure
Set-Cookie: USER_UUID=MZ4dzssan/gGyyPiazP8qxdHKEGPFJCn; Path=/; Secure
Location: https://cims1.consline.com/c
Strict-Transport-Security: max-age=31556926
X-Content-Type-Options: nosniff
X-Frame-Options: deny
X-XSS-Protection: 1; mode=block
Vary: Accept-Encoding
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=UTF-8

C4 Zugriffsschutz

C4.1 Privilegienerweiterung

Bedrohung

Die Erweiterung von Privilegien kann in unterschiedlichen Formen auftreten, z.B.:

- Vertikale Erweiterung von Privilegien bezeichnet die Möglichkeit, dass ein angemeldeter Benutzer die Rechte/Rollen eines anderen Benutzers für einzelne oder alle Funktionen an- oder übernehmen kann.
- Horizontale Erweiterung von Privilegien bezeichnet die Möglichkeit, dass ein angemeldeter Benutzer Rechte erlangen kann, die ihm durch normale Bedienung nicht gegeben sind.

Beobachtung

Hinweise auf die Erweiterung von Privilegien wurden nicht gefunden.

C5 Konfiguration und Deployment

Unter Konfiguration und Deployment sind die Schwachstellen zusammengefasst, die ihre Ursache in den verwendeten Techniken und Methoden der Implementierung der Webanwendung haben.

C5.1 GET vs. POST Servlet-Schnittstelle

Bedrohung

Die Übertragung von Daten vom Client zum Server kann mittels der GET oder der POST-Methode erfolgen. In der Regel ist für sensitive Daten POST vorzuziehen, da dies schwieriger zu automatisieren ist und die Daten dann auch nicht in der Adressleiste des Clients stehen.

Mit GET kann ein Angreifer die Requests sehr leicht automatisieren, indem einfach die Änderungen im GET-Request übertragen werden. Außerdem ist es so möglich, die URL einfacher in E-Mails zu übertragen und ggf. automatisch auszulösen.

C5.1.1 Requests als GET und POST möglich

Risiko: Info

Die Anwendung verwendet teilweise POST-Requests, die auch als GET-Request abgesetzt werden können.

Beobachtung

Die Anwendung verwendet die POST-Methode in den Formularen. Es ist aber genauso möglich die Formulardaten mit der GET-Methode zu senden. Dies ist z.B. beim Erstellen eines Kommentares möglich, der sowohl als POST als auch als GET-Request abgesetzt werden kann.

Ausnutzbarkeit

Dieses Verhalten kann potentiell die Grundlage für die Ausnutzbarkeit anderer Schwachstellen sein. Es sind keine solchen Schwachstellen bekannt.

Maßnahme

Die Anwendung sollte den POST-Request erzwingen und GET-Requests ablehnen. Das Servlet muss die Werte der Parameter explizit vom entsprechenden POST-Objekt lesen. Werte im GET-Objekt sind zu ignorieren.

C5.2 HTTP vs. HTTPS

Bedrohung

Alle Daten können auf dem Übertragungsweg an vielen Stellen mitgelesen werden. Sensible Daten sind daher auf dem Übertragungsweg vor Ausspähung und Manipulation durch Verschlüsselung und Signatur zu schützen. Dies betrifft auch `FRAMES/FRAMESETs` als auch `IFRAMEs`.

Bei gemischten (HTTP und HTTPS) Seiten kann ein Benutzer nicht erkennen welche Teile per SSL geschützt sind und welche nicht, was Phishing-Angriffe erleichtert.

Dies betrifft auch den HTTP-Header, der z.B. die Cookies mit der SessionID enthält, sowie die vertrauliche Authentifizierungsinformation Benutzername und Passwort.

Beobachtung

Die Anwendung benutzt ausschließlich HTTPS für die Kommunikation zwischen Browser und Server. Der Zugriff per HTTP wird abgelehnt, da Port 80 auf dem Server nicht verfügbar ist.

C5.3 Benennung des Content-Type

Bedrohung

Durch das falsche Setzen des Content-Types in der Response können Sicherheitslücken entstehen, die es z.B. ermöglichen Cross-Site-Scripting Angriffe auszuführen.

Beobachtung

Es konnte keine Response mit falschem Content-Type entdeckt werden.

C5.4 Directory-Indexing

Bedrohung

Als Directory-Indexing bezeichnet man die Eigenschaft des Webservers, bei Angabe einer URL ohne Dateinamen das entsprechende Verzeichnislisting anzuzeigen.

Oft sind in den Verzeichnissen auch Dateien, die z.B. zur Konfiguration der Anwendung dienen, oder von anderen Dateien inkludiert werden. Diese dürfen i.d.R. nicht öffentlich zugänglich sein.

Direcotry-Indexing ist auch eine Form von Information-Disclosure.

Beobachtung

Hinweise auf Directory-Indexing wurden nicht gefunden.

C5.5 Beispieldateien

Bedrohung

Bei vielen Standard-Installationen von Web- und Applikationsservern werden Beispieldateien mit installiert, die z.B. dem Administrator bei seinen ersten Schritten helfen sollen.

Solche Beispielseiten können zum einen Installationsdetails oder sonstige Informationen über das System oder gar selbst Schwachstellen jeder Art enthalten (diese Seiten sind ja explizit nicht für den produktiven Betrieb gedacht).

Die in den Seiten enthaltenen Informationen können einem Angreifer helfen oder ihn ermutigen gezielt nach weiteren Schwachstellen zu suchen.

Beobachtung

Beispiel-Dateien wurde nicht gefunden.

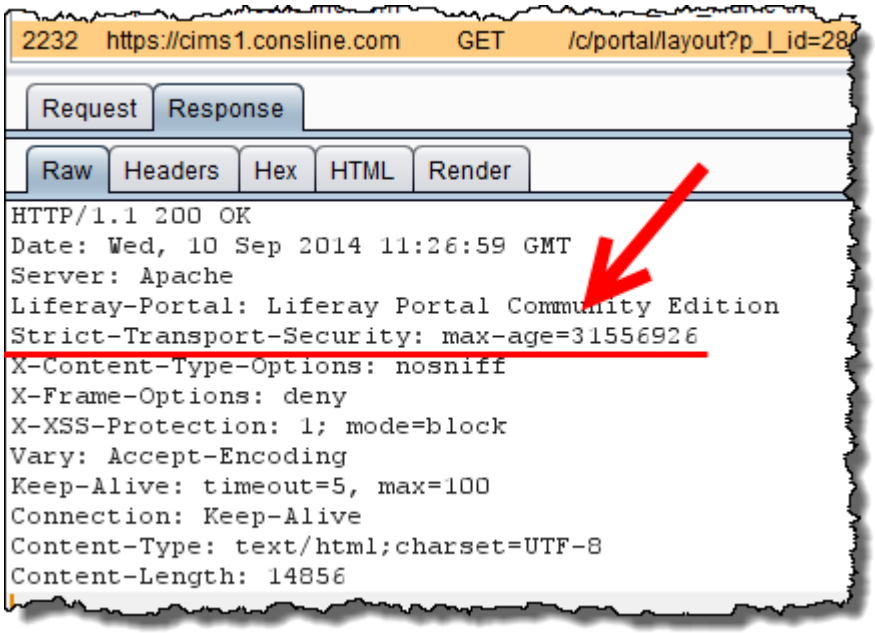
C5.6 Serverkonfiguration für HSTS (HTTP Strict Transport Security)

Bedrohung

Hat ein Angreifer eine Man-in-the-middle Position zwischen Client und Zielserver, kann er durch den Einsatz von z.B. SSL-Strip gezielt den Aufbau einer verschlüsselten Kommunikation verhindern. Der Angreifer ersetzt dabei die eigentlichen HTTPS Links in der Webseite durch unverschlüsselte HTTP Links („stripping“). Dadurch wird beim Aufruf von vermeintlich sicheren Links im Client eine unverschlüsselte HTTP Verbindung zum Angreifer aufgebaut, der seinerseits eine korrekte HTTPS Verbindung zum Zielserver aufbaut. Die unverschlüsselte Verbindung ist für den Client nur schwer zu erkennen und dem Angreifer ist es möglich die komplette Kommunikation im Klartext aufzuzeichnen und zu analysieren.

Beobachtung

Ein HTTPS-Request an den Webserver enthält im Response Header das Attribut `Strict-Transport-Security` mit dem Wert max-age=31556926 und ist damit für ein Jahr gültig.



C5.7 Serverkonfiguration für MIME-Sniffing

Bedrohung

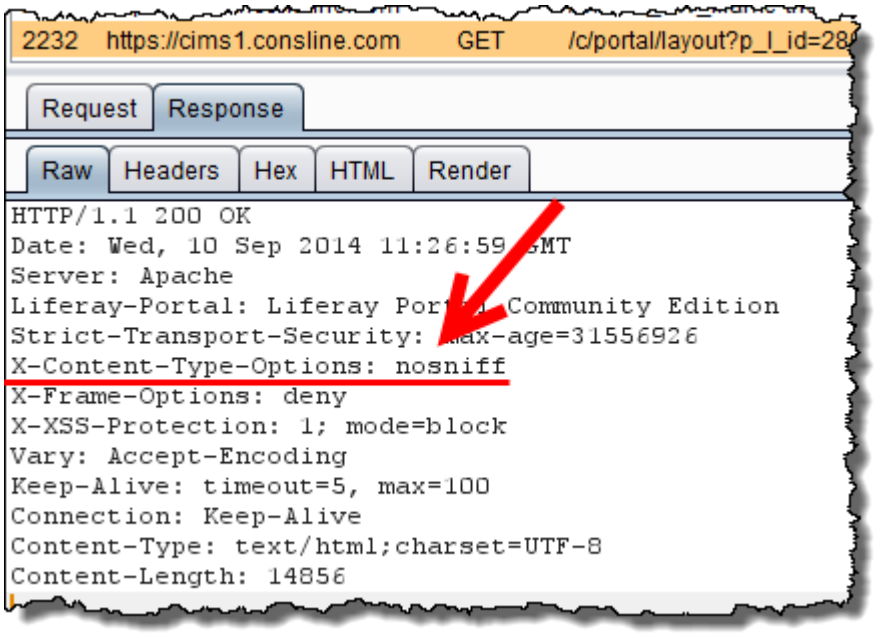
Der `X-Content-Type-Options` HTTP-Header stellt sicher, dass der vom Webserver ausgelieferte Content-Type auch im Browser als solcher umgesetzt wird.

Der Microsoft Internet Explorer implementiert eine Technologie, welche es dem Browser ermöglicht den MIME-Type der aufgerufenen Ressource zu bestimmen. Diese Technologie ist als MIME-Sniffing bekannt und kann von einem Angreifer als „drive-by download“ Möglichkeit ausgenutzt werden. Hierbei nutzt der Angreifer eine Datei-Upload-Funktionalität in der Anwendung um beispielsweise eine dynamische HTML-Datei hochzuladen, anstelle der vom Webserver erwarteten Bilddatei. Dies kann er durch z.B. eine Veränderung der Dateiendung erreichen. Wird nun diese Datei von der Anwendung an den Browser ausgeliefert, so kann es in einigen Browser-Versionen vorkommen, dass durch MIME-Sniffing der Inhalt der Bilddatei als HTML gerendert wird. Dies hat zur Folge, dass der dynamische HTML-Inhalt im Browser des Anwenders zur Ausführung kommt.

Ist der X-Content-Type-Options HTTP-Header gesetzt, so wird Browser angewiesen, den im HTTP-Header gesetzten Content-Type als MIME-TYPE zu verwenden.

Beobachtung

Ein HTTPS-Request an den Webserver enthält im Response Header das Attribut `X-Content-Type-Options: nosniff`. Damit erfolgt kein MIME-Sniffing durch Browser, die diese Direktive auswerten können.



C5.8 Serverkonfiguration für Anti-Clickjacking

Bedrohung

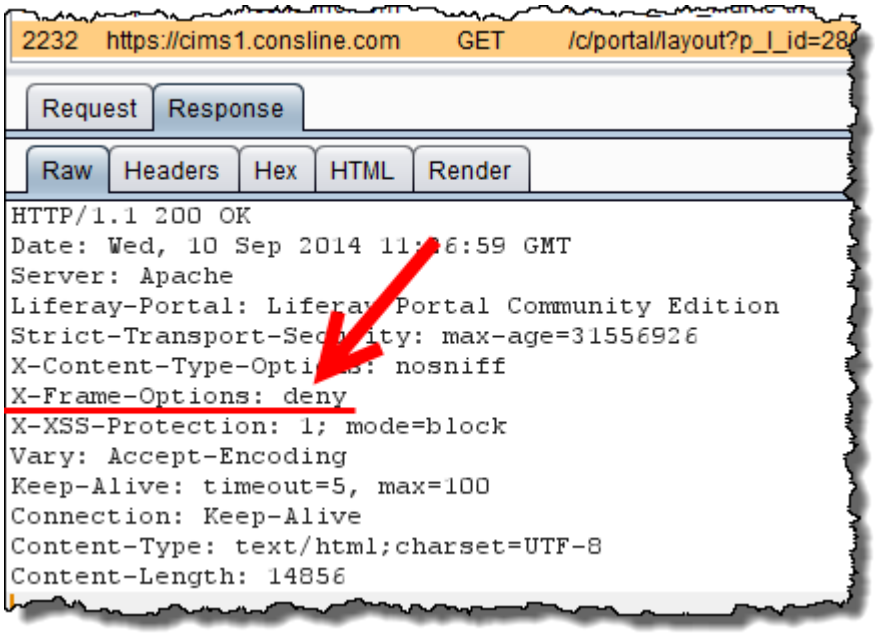
Der `X-Frame-Options` HTTP-Header legt fest, ob eine vom Browser angeforderte Webseite in einem Frame dargestellt werden darf.

Frames werden beispielsweise in Clickjacking-Angriffen genutzt, um den eigentlichen schadhaften Inhalt einer Anwendung durch eine legitim erscheinende Anwendung auf einem transparenten Layer zu überdecken. Benutzereingaben werden nun nicht wie beabsichtigt an die überlagernde Anwendung gesandt, sondern an die darunterliegende vom Angreifer kontrollierte Webseite.

Der `X-Frame-Options` HTTP-Header begegnet diesem Angriffs-Vektor, indem er definiert, dass die eigene Anwendung nicht im Kontext eines Frames verwendet werden darf.

Beobachtung

Ein HTTPS-Request an den Webserver enthält im Response den Header `X-Frame-Options` mit dem Wert `deny`.



C5.9 Serverkonfiguration für XSS-Schutz

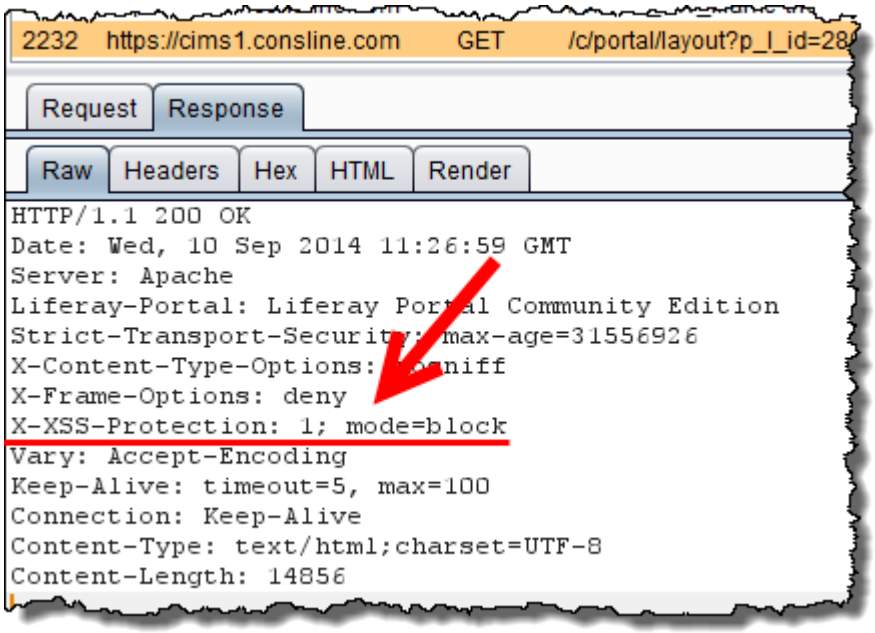
Bedrohung

Der `X-XSS-Protection` HTTP-Header reduziert die Ausnutzbarkeit von Reflected Cross-Site-Scripting-Angriffen in Browsern. Er wird in den aktuellen Versionen des Microsoft Internet Explorers verwendet, um den vom Browser implementierten XSS-Schutz explizit zu aktivieren. Der XSS-Schutz ist standardmäßig aktiviert, kann jedoch vom Benutzer deaktiviert werden. Dieser Header stellt sicher, dass für die Anwendung der XSS-Schutz aktiviert wird.

Neben dem Internet Explorer unterstützen auch noch WebKit-basierte Browser diesen HTTP-Header.

Beobachtung

Ein HTTP-Request an den Webserver enthält im Response Header das Attribut `X-XSS-Protection` mit dem Wert `1; mode=block`.



C5.10 Port-Scanning

Bedrohung

Als Port-Scanning bezeichnet man die systematische Überprüfung eines Servers auf die Erreichbarkeit von Ports und der darüber verfügbaren Dienste. Findet ein Angreifer offene Ports auf einem Server, stellt dies die Grundlage für weitere, spezielle Angriffe dar.

Eine große Gefahr geht dabei von offenen Ports und Diensten aus, die zum einen auf dem Server gar nicht aktiv sein sollten, oder durch eine falsche Konfiguration über das Internet erreichbar sind. Oft werden Dienste auch standardmäßig nach einer Installation des Betriebssystems gestartet und sind damit in einer Default-Konfiguration erreichbar, die damit leicht angreifbar ist.

Oft ermöglicht das Auffinden von offenen Ports einem Angreifer, an grundlegende Informationen des Servers zu gelangen und ist damit auch eine Form von Informationsabfluss.

Beobachtung

Der folgende Port wurde auf dem Host gefunden und ist über das Internet erreichbar:

Port	Service	Version
443	HTTPS	Apache

Der Host ist damit auf die benötigten Ports eingeschränkt und stellt keine zusätzliche Angriffsfläche bereit.

C6 Logik

C6.1 „Passwort vergessen“-Funktion

Bedrohung

Hier wird die "Passwort vergessen"-Funktion auf korrekte Implementierung untersucht.

Die „Passwort vergessen“-Funktion ermöglicht es einem Nutzer den Zugang zu seinem Benutzerkonto wiederherzustellen. Im Falle eines Passwortverlusts, egal ob durch Eigenverschulden oder Fremdeinwirkung kann dieser Vorgang auf unterschiedliche Art vollzogen werden, sei es beispielsweise durch eine Passwort-Reset E-Mail an die hinterlegte E-Mail-Adresse oder durch definierte Sicherheitsfragen des Nutzers.

Die Sicherheit dieser Verfahren ist unter anderem abhängig von der korrekten und robusten Implementierung der „Passwort vergessen“-Funktion.

Beobachtung

Die Anwendung hat keine „Passwort vergessen“-Funktion implementiert.

C6.2 „Passwort ändern“-Möglichkeit

Bedrohungen

Dem Benutzer sollte eine einfache Möglichkeit zur selbständigen Änderung seines Passwortes zur Verfügung stehen. Fehlt eine solche Möglichkeit, kann sich der Benutzer beim Bekanntwerden seines Passwortes nicht vor Missbrauch schützen und der allgemeinen Schutzempfehlung, ein Passwort regelmäßig zu ändern, nicht nachkommen.

C6.2.1 Keine Möglichkeit zur Änderung des Passwortes

Risiko: Info

Die Anwendung bietet dem Benutzer keine Möglichkeit zur Änderung seines Passworts.

Beobachtung

Der Anwender kann weder in der Applikation noch über einen Passwort Reset sein Passwort selbstständig ändern.

Ausnutzbarkeit

Kommt ein Angreifer in den Besitz eines Passwortes von einem Benutzerkonto, kann der Benutzer dies nicht selbständig zurücksetzen. Der Benutzer muss den Support kontaktieren, der das Passwort manuell zurücksetzt. Ein Angreifer hat damit unter Umständen ein längeres Zeitfenster um das Konto zu missbrauchen.

Maßnahme

Ein Benutzer sollte eine einfache Möglichkeit haben, sein Passwort selbständig zu ändern.

C6.3 Denial-of-Service durch Benutzersperrung

Bedrohung

Erfolgt bei mehrfachen fehlerhaften Loginversuchen eine Sperrung des Benutzers, so kann ein Dritter dies ausnutzen, um andere Benutzer missbräuchlich auszusperren. Durch entsprechende Automatisierung kann ein Angreifer dafür sorgen, dass ein Benutzer, der freigeschaltet worden ist, sofort wieder ausgesperrt wird. Auch ist es häufig möglich, eine große Anzahl von Benutzern in kurzer Zeit zu sperren. Der Schaden ist umso größer, je einfacher Benutzernamen erraten oder systematisch durchlaufen werden können. Darüber hinaus können auf diese Weise Social Engineering-Angriffe erleichtert werden.

Ebenso finden sich weiterführende Informationen unter /31/ und /32/.

Beobachtung

Eine der Anforderungen laut der Ausschreibung "Permanentes Internet Monitoring für die Bereich After Sales und Marketing des Kunden, 4.1.3. Authentifizierung / Login" ist, dass bei mehrfacher Eingabe einer falschen Kombination von Benutzername und Passwort eine sofortige Sperrung des Accounts stattfinden muss (jederzeit wieder freischaltbar).

C7 Informationsabfluss

C7.1 Bekanntgabe von unterschiedlichen Systeminformationen

Bedrohung

Als Information-Disclosure bezeichnet man die Tatsache, dass eine Anwendung Informationen preis gibt, die für die Bedienung nicht erforderlich sind, aber Rückschlüsse auf die Arbeitsweise oder die verwendeten Techniken zulassen.

Wenn solche Informationen nicht für den Zugriff von außen bestimmt sind, dann kann dies auch ein Datensicherheitsproblem darstellen. Alle Informationen, die für den Betrieb der Anwendung durch den Benutzer nicht notwendig sind, können einem Angreifer auch helfen in ein System einzudringen.

Das Vorhandensein einer einzigen solcher Schwachstelle ist dabei häufig nicht als Problem anzusehen. Kommen jedoch mehrere solcher Schwachstellen zusammen, dann lässt sich daraus ein Gesamtbild konstruieren, das eine Bedrohung darstellt.

Informationen wie Serverbanner, Footprints von Modulen oder Software-Komponenten und sogar Hinweise auf Autoren einer Software ermöglichen einem Angreifer gezielt nach Schwachstellen zu suchen. Eine solche Suche umfasst z.B. die Known-Vulnerabilities (siehe 0). Ein Angreifer kann dann weitere Stellen in der Website suchen, die die gleichen Eigenschaften und damit Angriffspunkte haben.

Mögliche Maßnahme

Zur Einschränkung von Information-Disclosure sollte ein gewisser Aufwand getrieben werden. Einige Ansatzpunkte sind:

- Kommentare aus HTML-Seiten entfernen
- Server-Banner entfernen oder neutral formulieren (für Apache in der httpd.conf `ServerToken prod` setzen)
- Fingerprints, Kommentare und andere Module entfernen
- typische Merkmale wie `META`-Tags, Formvariablen usw. neutral umbenennen
- Fehlermeldungen abfangen und dafür neutrale Fehlerseiten liefern
- Serverkonfiguration überprüfen (z.B. Indexing abschalten, siehe C5.4)
- Enumeration verhindern

Beobachtung

Es konnten keine Systeminformationen identifiziert werden.

C7.2 Fehlerseiten

Bedrohung

Fehlerseiten enthalten oft interne Information, z.B. Statusmeldung der Datenbank, Java-Stacktraces oder Serverbanner, die dazu benutzt werden können nach weiteren Schwachstellen zu suchen. Eine weitere häufig anzutreffende Schwachstelle ist, dass die Benutzereingaben ungefiltert in die Fehlerseite übernommen werden, was dann zu allen Formen von Cross-Site-Scripting führen kann.

Die in den Seiten enthaltenen Informationen können einem Angreifer helfen oder ihn ermutigen gezielt nach weiteren Schwachstellen zu suchen.

Beobachtung

Fehlerseiten, die auf Konfigurationsfehler hindeuten, wurden nicht gefunden.

C8 Kryptographie

C8.1 SSL-Verschlüsselung

Bedrohung

Beim TSL/SSL-Verbindungsaufbau einigen sich Client und Server auf eine Cipher-Suite, die beide verstehen. Bietet der Server hier einen schwachen, möglicherweise brechbaren Verschlüsselungsalgorithmus an, so kann ein bösartiger Client durch geschicktes "Verhandeln" eine schwache Verschlüsselung provozieren.

Siehe auch SSL Best Practices /33/.

Beobachtung

Der Host `cims1.consline.com` wurde hinsichtlich der angebotenen Verschlüsselungsverfahren untersucht.

Chiffre	Sicherheit	SSL/TLS	Bits	Bewertung
ECDHE-RSA-AES256-SHA	very strong	SSLv3	256	[OK]
DHE-RSA-AES256-SHA	very strong	SSLv3	256	[OK]
DHE-RSA-CAMELLIA256-SHA	very strong	SSLv3	256	[OK]
AES256-SHA	very strong	SSLv3	256	[OK]
CAMELLIA256-SHA	very strong	SSLv3	256	[OK]
ECDHE-RSA-DES-CBC3-SHA	strong	SSLv3	168	[OK]
EDH-RSA-DES-CBC3-SHA	strong	SSLv3	168	[OK]
DES-CBC3-SHA	strong	SSLv3	168	[OK]
ECDHE-RSA-AES128-SHA	strong	SSLv3	128	[OK]
DHE-RSA-AES128-SHA	strong	SSLv3	128	[OK]
DHE-RSA-SEED-SHA	strong	SSLv3	128	[OK]
DHE-RSA-CAMELLIA128-SHA	strong	SSLv3	128	[OK]
AES128-SHA	strong	SSLv3	128	[OK]
CAMELLIA128-SHA	strong	SSLv3	128	[OK]
ECDHE-RSA-AES256-SHA	very_strong	TLSv1	256	[OK]
DHE-RSA-AES256-SHA	very_strong	TLSv1	256	[OK]
DHE-RSA-CAMELLIA256-SHA	very_strong	TLSv1	256	[OK]
AES256-SHA	very_strong	TLSv1	256	[OK]
CAMELLIA256-SHA	very_strong	TLSv1	256	[OK]
ECDHE-RSA-DES-CBC3-SHA	strong	TLSv1	168	[OK]
EDH-RSA-DES-CBC3-SHA	strong	TLSv1	168	[OK]
DES-CBC3-SHA	strong	TLSv1	168	[OK]
ECDHE-RSA-AES128-SHA	strong	TLSv1	128	[OK]
DHE-RSA-AES128-SHA	strong	TLSv1	128	[OK]
DHE-RSA-SEED-SHA	strong	TLSv1	128	[OK]
DHE-RSA-CAMELLIA128-SHA	strong	TLSv1	128	[OK]

AES128-SHA	strong	TLSv1	128	[OK]
CAMELLIA128-SHA	strong	TLSv1	128	[OK]

Bemerkung

Es konnten keine zu kurzen Schlüssellängen oder schwache Ciphers identifiziert werden.

Hinweis

Neben SSLv3 wird auch TLSv1.0, TLSv1.1 und TLSv1.2 angeboten.

D. Automatische Tests

Die Anwendung wurde mit den Tools *nessus* und *nikto* ausgiebig getestet. Alle Ergebnisse wurden manuell nachgeprüft und sind in den obigen Bericht eingeflossen.

D1.1 Überprüfung mit Nessus

Bedrohung

Durch falsche Konfigurationen, alte Softwarestände oder ungeschützte Services können Sicherheitslücken nicht nur in der Webapplikation, sondern auch im Webserver und an anderen Stellen enthalten sein.

D1.1.1 Test [Info]

Beobachtung

Der Nessus Vulnerability Scanner hat den über das Internet erreichbaren Host `cims1.consline.com` auf Schwachstellen überprüft.

Dabei konnten keine weiteren Schwachstellen identifiziert werden.

D1.2 Überprüfung mit nikto

Bedrohung

Durch falsche Konfigurationen, veraltete Software oder ungeschützte Services können Sicherheitslücken nicht nur in der eigentlichen Webapplikation, sondern auch in anderen Stellen enthalten sein.

Beobachtung

Es konnten durch den Einsatz von nikto keine weiteren Schwachstellen identifiziert werden.

E. Referenzen

/1/ Input- und Output-Datenvalidierung in Webanwendungen
https://cwe.mitre.org/data/definitions/79.html

/2/ Übersicht zu Datenvalidierung
https://www.owasp.org/index.php/Data_Validation

/3/ Cheat Sheet zu Datenvalidierung
https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet

/4/ Übersicht zur XSS-Prävention
https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

/5/ Unvalidierte Redirects und Forwards
https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet

/6/ Validierung von Rich-Content
https://www.owasp.org/index.php/Secure_Coding_Cheat_Sheet#Validating_Rich_User_Content

/7/ Übersicht zu SQL-Injection
https://www.owasp.org/index.php/SQL_Injection

/8/ Übersicht zu Blind-SQL-Injection
https://www.owasp.org/index.php/Blind_SQL_Injection

/9/ SQL-Injection-Prävention
https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet

/10/ Best Practice Maßnahmen zur Auditierung und Härtung von Datenbanken
http://benchmarks.cisecurity.org/downloads/browse/index.cfm?category=benchmarks.servers.database

/11/ System-Kommandos aufrufen
http://cwe.mitre.org/data/definitions/78.html

/12/ Enumeration von Ressourcen
https://www.owasp.org/index.php/Forced_browsing

/13/ Path-Traversal
https://www.owasp.org/index.php/Path_Traversal

/14/ Buffer-Overflow-Angriff
https://www.owasp.org/index.php/Buffer_overflow_attack

/15/ Format-String-Angriff
https://www.owasp.org/index.php/Format_string_attack

/16/ Benutzer-Enumeration
https://www.owasp.org/index.php/Testing_for_user_enumeration_(OWASP-AT-002)

/17/ OWASP - Blocking Brute Force Attacks
https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

/18/ Länge und Komplexität von Passwörtern
https://www.owasp.org/index.php/Password_length_&_complexity

/19/ Übersicht zu Authentication
https://www.owasp.org/index.php/Authentication_Cheat_Sheet

/20/ Übersicht zu Session-Management
https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

/21/ Überwindung des Session-Kontroll-Mechanismus
https://www.owasp.org/index.php/Session_hijacking_attack

/22/ Session-Fixation-Schwachstelle
http://www.acros.si/papers/session_fixation.pdf

/23/ Einführung in CSRF bzw. Session Riding
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

/24/ CSRF-Prävention
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet

/25/ Privilegienerweiterung
http://cwe.mitre.org/data/definitions/269.html

/26/ Zugriffskontrolle
https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

/27/ Sichere Passwort-Verwaltung
https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

/28/ Sichere Datenspeicherung
https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

/29/ Best Practices for a Secure “Forgot Password” Feature
http://www.fishnetsecurity.com/sites/default/files/media/10WP0003_BestPractices_SecureForgotPassword%5B1%5D_0.pdf

/30/ Fehlerhafte Benutzer-Authentisierung und Session-Verwaltung
https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management

/31/ Denial of Service
https://www.owasp.org/index.php/Denial_of_Service

/32/ CAPTCHA: Telling Humans and Computers Apart Automatically
http://www.captcha.net/

/33/ SSL Best Practices
https://www.ssllabs.com/projects/best-practices/index.html

/34/ HTTP-Request-Smuggling-Angriff
https://www.owasp.org/index.php/HTTP_Request_Smuggling

/35/ Übersicht von öffentlich bekannten Schwachstellen
http://nvd.nist.gov/

/36/ Secure Coding of CORS
http://www.andlabs.org/html5/rejectCOR.php

/37/ Informationen über HSTS
http://blog.securenet.de/2012/11/02/ssl-stripping-die-ignorierte-gefahr/

/38/ OWASP Cheat Sheet Series
https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series

/39/ CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')
http://cwe.mitre.org/data/definitions/77.html

/40/ OWASP Content Security Policy
https://www.owasp.org/index.php/Content_Security_Policy

/41/ OWASP Clickjacking
https://www.owasp.org/index.php/Clickjacking

/42/ OWASP HTML5 Security Cheat Sheet
https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet

/43/ OWASP HTTP Headers
https://www.owasp.org/index.php/List_of_useful_HTTP_headers

/44/ Unrestricted Upload of File with Dangerous Type
http://cwe.mitre.org/data/definitions/434.html

/45/ Unrestricted File Upload
https://www.owasp.org/index.php/Unrestricted_File_Upload

/46/ BeEF Project
http://beefproject.com/