SYSLIFTERS

Demo-Report: Web

Pentest für **Security Maximale GmbH**
2022-08-29
v 1.0

Kontakt:
Aron Molnar
+436609234060
aron@syslifters.com

Inhalt

Management Summary

Im Zuge der Sicherheitsprüfung war es uns möglich, administrative Konten, sowie Kundenkonten des Onlineshops zu übernehmen. Dadurch konnten wir Kundendaten inklusive Zahlungsinformationen, sowie Kundenbestellungen einsehen und bearbeiten.

Dies gelang über zwei Wege: Einerseits war die Funktion zum Zurücksetzen der Passwörter nicht ausreichend abgesichert und stand auch den Kunden des Shops zur Verfügung. Damit konnten Kunden die Passwörter von Administratoren zurücksetzen. Andererseits konnten über die Statusnachrichten der Kunden bösartige JavaScript-Funktionen eingeschleust werden, wodurch administrative Benutzersessions übernommen werden konnten.

Zudem konnten über die Funktion zur Auswahl der auszuliefernden Sprache beliebige Dateien vom Webserver ausgelesen werden. Darunter waren etwa Konfigurationsdateien mit Passwörtern.

Weitere, weniger kritische Schwachstellen, wie etwa die Weiterleitung fremder Benutzer auf nicht vertrauenswürdige Webseiten, die Preisgabe von internen Informationen, sowie Schwächen im Session-Management sollten in einem kontinuierlichen Verbesserungsprozess adressiert werden.

Hier ist der Bericht. Was jetzt?

Es ist uns sehr wichtig, dass ihr mit unserem Bericht arbeitet und Verbesserungsmaßnahmen daraus ableitet. Deshalb testen wir für euch behobene Schwachstellen kostenlos nach, wenn sie innerhalb von acht Wochen behoben werden!

In diesem Assessment haben wir Schwachstellen mit der Kritikalität `Critical` und `High` gefunden. Wir empfehlen, diese Schwachstellen vorrangig zu beheben.

Schwachstellen mit weniger komplexen Gegenmaßnahmen und Risiko `Medium` und darunter sollten nach unserer Empfehlung nach Aufwand priorisiert behoben werden. Alle anderen Schwachstellen sollten im Rahmen eines kontinuierlichen Verbesserungsprozesses adressiert werden.

Bitte stellt sicher, alle im Zuge des Pentests bereitgestellten Benutzer und Ressourcen zu deprovisionieren, sobald sie nicht mehr benötigt werden.

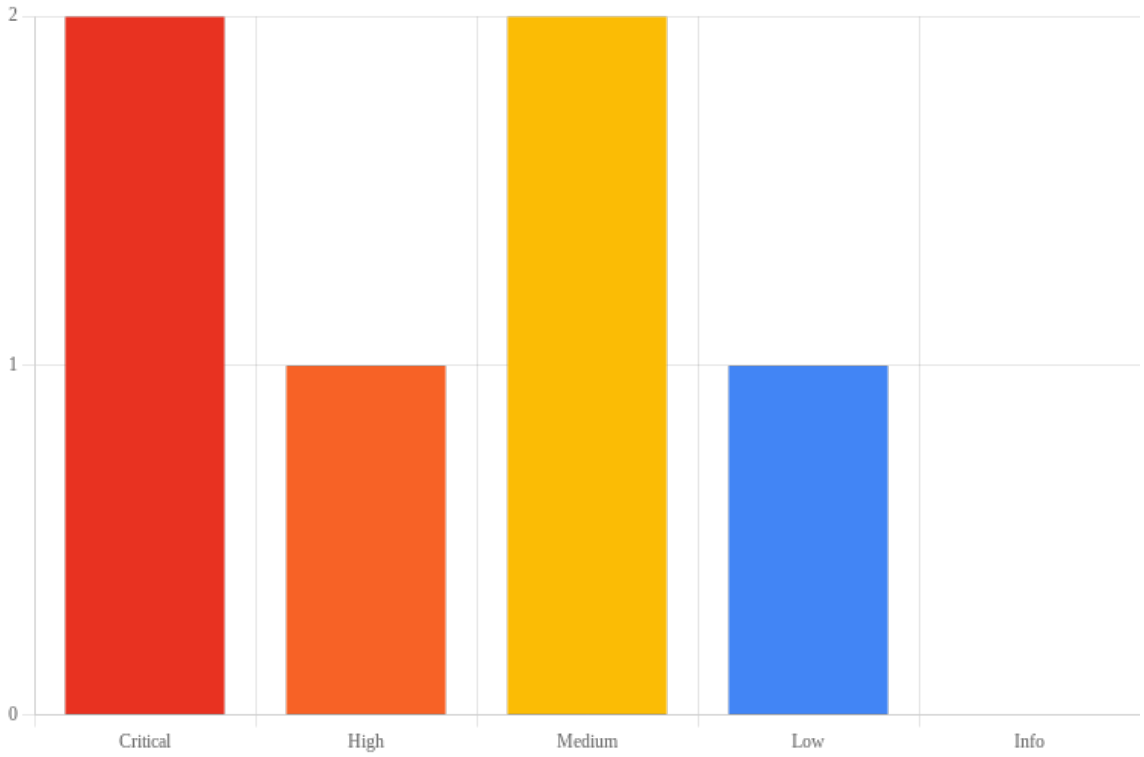
Scope und Dauer

Der Scope des Pentests umfasste das Onlineshop-System der Security Maximale GmbH unter https://www.example.com.

Der Penetration-Test erfolgte nach einem Time-Box-Ansatz und umfasste 10 Personentage.

Schwachstellenübersicht

Im Rahmen dieses Penetration Tests wurden **2 Critical** , **1 High** , **2 Medium** und **1 Low** Schwachstellen identifiziert:



Verteilung der gefundenen Schwachstellen

Eine tabellarische Übersicht aller gefundenen Schwachstellen:

Schwachstelle	Kritikalität
Übernahme von Benutzer-Accounts durch Passwort-Reset	Critical
Stored Cross-Site Scripting (XSS)	Critical
Offenlegung von Passwörtern über Path Traversal	High
Weiterleitung von Benutzern durch Open Redirect	Medium
Preisgabe von internen Informationen	Medium
Schwächen im Session-Management	Low

Eine Auflistung aller Schwachstellen inklusive Kurzbeschreibung:

1. Übernahme von Benutzer-Accounts durch Passwort-Reset (Critical: 9.9)
Betrifft: http://www.example.com/api/v2/users/change-password/admin

Im Zuge der Prüfung war es uns als Kunde des Onlineshops möglich, die Passwörter anderer Kunden, sowie der Shop-Administratoren auf beliebige Passwörter zu ändern. Dadurch konnten wir administrative Benutzerkonten übernehmen und damit das gesamte Shop-System kompromittieren.

2. Stored Cross-Site Scripting (XSS) (Critical: 9.3)
Betrifft: http://www.example.com/api/v2/users/status

Zum Zeitpunkt der Prüfung speicherte die Webanwendung Benutzereingaben ungeprüft und band diese später auf unsichere Weise in HTTP-Responses ein. Sie war damit anfällig für Stored Cross-Site Scripting Angriffe (XSS).
Dadurch könnten Angreifer die Benutzerkonten anderer Benutzer - darunter Administratoren - kompromittieren.

3. Offenlegung von Passwörtern über Path Traversal (High: 7.5)
Betrifft: http://www.example.com/?lang=de

Benutzer der Applikation konnten zum Zeitpunkt der Prüfung Zugriff auf Passwörter in Konfigurationsdateien erlangen. Dies gelang über einen Path Traversal-Angriff, weil die Web-Applikation HTTP-Parameter nicht ausreichend prüfte.

4. Weiterleitung von Benutzern durch Open Redirect (Medium: 6.1)

Über eine Open Redirect-Schwachstelle konnten Benutzer zum Zeitpunkt der Prüfung auf andere Webseiten weitergeleitet werden.
Dafür mussten die Benutzer auf einen manipulierten Link klicken. Der Link wirkt vertrauenswürdig, da er zur Web-Applikation zeigt, welche den Benutzer jedoch umgehend auf eine fremde Webseite weiterleitet.

5. Preisgabe von internen Informationen (Medium: 5.3)
Betrifft: http://www.example.com/.env

Die Webanwendung offenbarte interne Informationen, die potenziell für weiterführende Angriffe genutzt werden könnten. Die Offenlegung von Informationen, auch bekannt als Information Disclosure, liegt vor, wenn ein System unbeabsichtigt interne Informationen an ihre Nutzer weitergibt.

6. Schwächen im Session-Management (Low: 3.6)

Wir konnten Schwächen im Sessions-Management der Webapplikation identifizieren. Die Sessions der Benutzer waren ohne zeitliche Einschränkung nutzbar und erforderten daher zu keinem Zeitpunkt eine Neuauthentifizierung. Personen mit Zugriff auf ein Computersystem könnten diesen Sachverhalt ausnutzen, wenn sich ein anderer Benutzer zuvor bei der Anwendung nicht explizit abgemeldet hat.

Schwachstellendetails

1. Übernahme von Benutzer-Accounts durch Passwort-Reset

Kritikalität: Critical
CVSS-Score: 9.9
Betrifft: http://www.example.com/api/v2/users/change-password/admin
Empfehlung: Benutzer sollten die Funktion zur Änderung der Passwörter nicht verwenden dürfen.

Überblick

Im Zuge der Prüfung war es uns als Kunde des Onlineshops möglich, die Passwörter anderer Kunden, sowie der Shop-Administratoren auf beliebige Passwörter zu ändern. Dadurch konnten wir administrative Benutzerkonten übernehmen und damit das gesamte Shop-System kompromittieren.

Beschreibung

Administratoren des Onlineshops sollen in der Lage sein, Passwörter der Kunden zurückzusetzen.
Im Zuge der Prüfung konnten wir feststellen, dass diese Funktion nicht auf Administratoren eingeschränkt, sondern sämtlichen Benutzern, darunter registrierten Kunden, möglich war.

Der API-Endpunkt `POST /api/v2/users/change-password/{username}` überschreibt das Passwort des angegebenen Benutzers.

Überschreiben von Benutzer-Passwörtern via Change-Password API

Der Server antwortete mit dem Status-Code 200 "OK", und änderte das Passwort des Benutzerskontos "admin".

Empfehlung

Kunden und nicht administrative Benutzer des Onlineshops sollten die Funktion zur Änderung der Passwörter nicht verwenden dürfen.

2. Stored Cross-Site Scripting (XSS)

Kritikalität: Critical
CVSS-Score: 9.3
Betrifft: http://www.example.com/api/v2/users/status
Empfehlung: Benutzereingaben sollten validiert und gefiltert werden. Es sollte sichergestellt werden, dass Daten kontextabhängig richtig enkodiert werden, bevor sie in HTTP-Responses eingebunden werden.

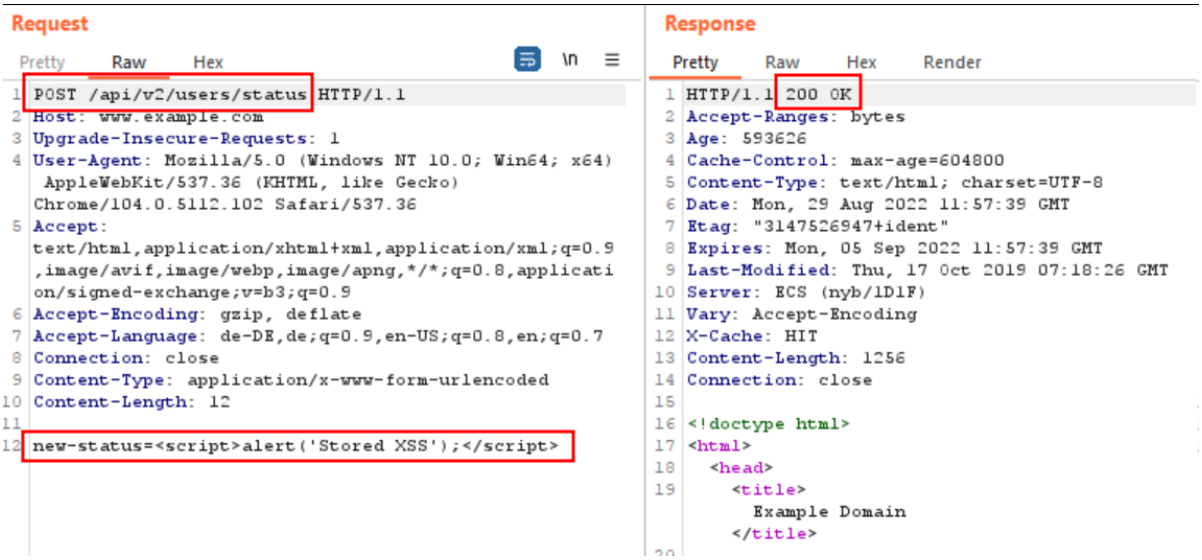
Überblick

Zum Zeitpunkt der Prüfung speicherte die Webanwendung Benutzereingaben ungeprüft und band diese später auf unsichere Weise in HTTP-Responses ein. Sie war damit anfällig für Stored Cross-Site Scripting Angriffe (XSS).
Dadurch könnten Angreifer die Benutzerkonten anderer Benutzer - darunter Administratoren - kompromittieren.

Beschreibung

Wir konnten im Zuge der Tests eine Stored XSS-Schwachstelle in der Webanwendung identifizieren. Aufgrund fehlerhafter Validierung und Kodierung von Daten konnten wir als Kunde des Onlineshops bösartige Skripte einschleusen und dort persistent speichern.

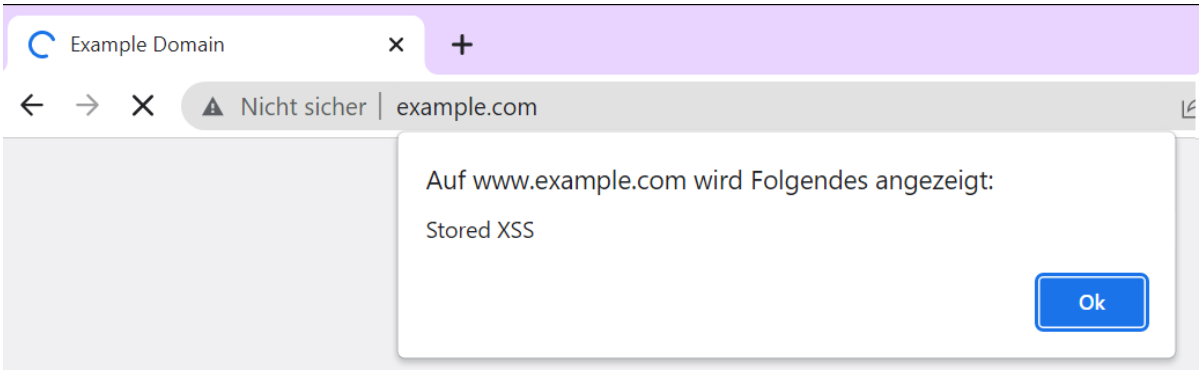
Benutzer können in der Applikation eine Status-Nachricht festlegen.



Festlegen der Status-Nachricht

Beim Anklicken des entsprechenden Benutzerprofils, sowie in der Benutzerübersicht im Backend wird die Status-Nachricht in das HTML-Dokument eingebettet. Handelt es sich

bei der Nachricht um JavaScript-Code, wird dieser schließlich im Browser eines anderen Benutzers zur Ausführung gebracht.



Ausführung des JavaScript-Codes im Backend

Dadurch könnten die Benutzerkonten anderer Benutzer, darunter auch jene der Administratoren, kompromittiert werden.

Cross-Site Scripting (XSS) ist eine häufig auftretende Websicherheitslücke, bei der bösartige Skripte in Webanwendungen aufgrund unzureichender Validierung oder Kodierung von Daten eingeschleust werden können. Bei XSS-Angriffen betten Angreifer JavaScript-Code in die von der anfälligen Webanwendung gelieferten Inhalte ein.

Das Ziel bei Stored XSS-Angriffen ist es, Script-Code auf Seiten zu platzieren, die von anderen Benutzern besucht werden. Das bloße Aufrufen der betroffenen Unterseite ist ausreichend, damit der Skript-Code im Webbrowser des Opfers ausgeführt wird.

Für einen Angriff werden bösartige Skripte durch den Angreifer in die Webanwendung eingeschleust und gespeichert und in späteren HTTP-Responses der Applikation eingebunden. Das bösartige Skript wird letztendlich im Webbrowser des Opfers ausgeführt und kann potenziell auf Cookies, Session-Tokens oder andere sensible Informationen zugreifen.

Bei einem erfolgreichen Angriff erhält ein Angreifer Kontrolle über Funktionen und Daten der Webanwendung im Kontext des Opfers. Wenn der betroffene Benutzer privilegierten Zugriff besitzt, kann ein Angreifer unter Umständen vollständige Kontrolle über die Webanwendung erhalten.

Empfehlung

- Es sollte sichergestellt werden, dass alle verarbeiteten Daten so streng wie möglich gefiltert werden. Es sollte auf Basis der erwarteten und gültigen Eingaben gefiltert und validiert werden.
- Daten sollten enkodiert werden, bevor die Webanwendung diese in HTTP-Responses einbindet. Die Kodierung sollte kontextabhängig geschehen, das heißt abhängig davon wo die Webanwendung Daten im HTML-Dokument einfügt, muss die entsprechende Kodierungssyntax berücksichtigt werden.

- Die HTTP-Header `Content-Type` (z. B. `text/plain`) und `X-Content-Type-Options: nosniff` können für HTTP-Responses gesetzt werden, die kein HTML und kein JavaScript enthalten.
- Wir empfehlen, zusätzlich eine Content Security Policy (CSP) einzusetzen, um zu steuern, welche clientseitigen Skripte erlaubt und welche verboten sind.
- Detaillierte Informationen und Hilfestellungen zur Verhinderung von XSS sind im verlinkten Cross-Site Scripting Prevention Cheat Sheet von OWASP zu finden.

Weiterführende Informationen

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

3. Offenlegung von Passwörtern über Path Traversal

Kritikalität: High
CVSS-Score: 7.5
Betrifft: http://www.example.com/?lang=de
Empfehlung: Verzichtet nach Möglichkeit auf benutzerdefinierte Dateinamens- und Pfadangaben in der Webanwendung.

Überblick

Benutzer der Applikation konnten zum Zeitpunkt der Prüfung Zugriff auf Passwörter in Konfigurationsdateien erlangen. Dies gelang über einen Path Traversal-Angriff, weil die Web-Applikation HTTP-Parameter nicht ausreichend prüfte.

Beschreibung

Wir konnten im Zuge der Prüfung eine Path Traversal-Schwachstelle in der Webanwendung identifizieren. Dadurch konnten wir auf sensible Dateien auf dem Serversystem zugreifen.

Der `lang`-Parameter, über den Benutzer eine individuelle Sprache auswählen können, war der Name eines Ordners, in dem Sprachinformationen abgelegt und dynamisch aufgerufen wurden.

Durch die Manipulation des Parameters konnten wir beliebige Dateien auf dem Webserver lesen, darunter etwa Konfigurationsdateien mit Passwörtern.

Auslesen des Datenbankpassworts der Web-Applikation

Path Traversal ist eine Websicherheitslücke, mit der ein Angreifer auf Dateien und Verzeichnisse des zugrundeliegenden Webservers einer Webanwendung zugreifen kann. Der Zugriff auf Dateien und Verzeichnisse ist bei einem Path Traversal Angriff ausschließlich durch die vorhandenen Zugriffskontrollen des zugrundeliegenden Betriebssystems eingeschränkt.

Die meisten Webserver beschränken den Zugriff dabei auf einen bestimmten Teil des Dateisystems, der üblicherweise als "Web Root" bezeichnet wird. Dieses Verzeichnis enthält alle Dateien, die für die Funktionalität der Webanwendung erforderlich sind. Um mittels Path Traversal Angriff aus dem Web-Root-Ordner auszubrechen, verwenden Angreifer spezielle Sonderzeichenfolgen in Pfadangaben.

In der einfachsten Form verwendet ein Angreifer die Sonderzeichenfolge "../", um den im Parameter angeforderten Speicherort der Ressource zu ändern. Bei dieser Zeichenfolge handelt es sich um eine relative Pfadangabe. Sie bezieht sich konkret auf das übergeordnete Verzeichnis des aktuellen Arbeitsverzeichnisses. Um gegebenenfalls vorhandene Sicherheitsfilter zu umgehen, verwenden Angreifer auch oft alternative Kodierungen der "../"-Sequenz. Zu diesen Methoden gehören etwa gültige und ungültige Unicode-kodierte ("..%u2216" oder "..%c0%af"), URL-kodierte ("%2e%2e%2f") und doppelte URL-kodierte Zeichen ("..%255c") des Backslash-Zeichens. Bei fortgeschrittenen Techniken werden häufig auch zusätzliche Sonderzeichen wie der Punkt ".", um auf das aktuelle Arbeitsverzeichnis zu verweisen, oder das "%00" NULL-Zeichen, um rudimentäre Dateiendungsprüfungen zu umgehen, verwendet.

Bei einem erfolgreichen Angriff kann ein Angreifer durch Path Traversal auf beliebige Dateien und Verzeichnisse auf dem anfälligen System zugreifen. Dazu können sensible Betriebssystemdateien, Anwendungscode oder Konfigurationsdateien gehören. In bestimmten Fällen kann es auch möglich sein, dass durch Path Traversal schreibend auf Dateien und Verzeichnisse zugegriffen werden kann. Unter diesen Umständen ist es möglich, dass ein Angreifer Code Execution und somit vollständige Kontrolle über den Webserver erhält.

Empfehlung

- Die effektivste Methode zur Vermeidung von Path Traversal, ist der Verzicht auf benutzerdefinierte Dateinamens- und Pfadangaben in der Webanwendung. Wenn das nicht möglich ist, verwendet alternativ Indizes anstelle konkreter Wertangaben (z. B. Index: 5 entspricht der Spracheinstellung "Deutsch").
- Validiert grundsätzlich immer alle Benutzereingaben. Stellt sicher, dass nur für die Anwendung erwartete und gültige Eingaben akzeptiert werden. Potenziell bösartige Eingaben sollten verworfen werden.
- Bedenkt bei der Normalisierung von Pfadangaben, dass Zeichen einfach oder mehrfach enkodiert sein könnten (etwa URL-enkodiert, z. B. `%20` oder `%2520` anstatt eines Leerzeichens).

4. Weiterleitung von Benutzern durch Open Redirect

Kritikalität: Medium
CVSS-Score: 6.1
Empfehlung: Lasst keine benutzerdefinierten URLs für Um- oder Weiterleitungen zu. Geht davon aus, dass alle Eingaben bösartig sind. Verfolgt daher einen Allow-Listing Ansatz, welcher eine Liste akzeptabler Eingaben definiert.

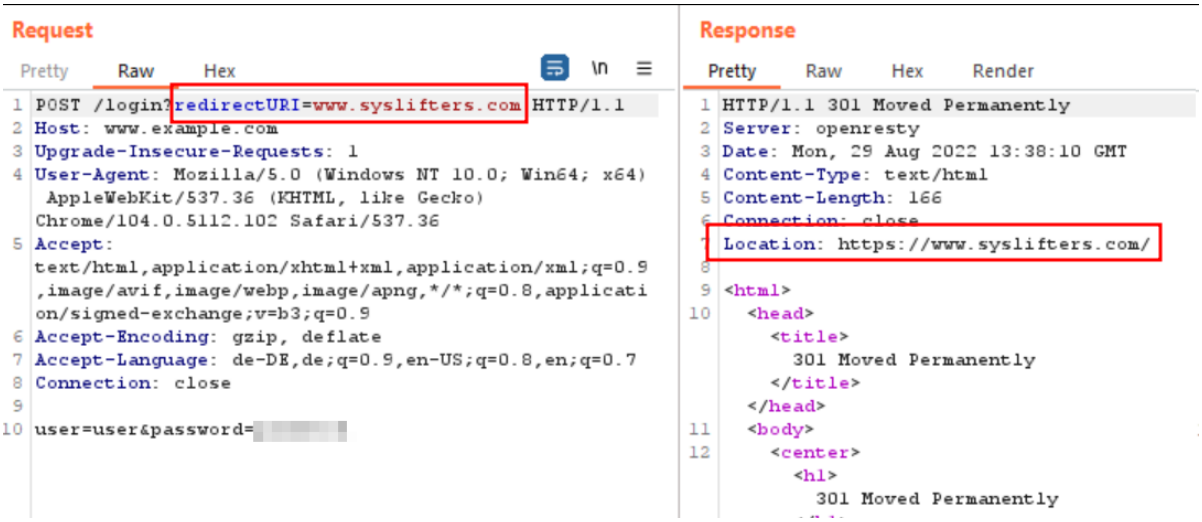
Überblick

Über eine Open Redirect-Schwachstelle konnten Benutzer zum Zeitpunkt der Prüfung auf andere Webseiten weitergeleitet werden.
Dafür mussten die Benutzer auf einen manipulierten Link klicken. Der Link wirkt vertrauenswürdig, da er zur Web-Applikation zeigt, welche den Benutzer jedoch umgehend auf eine fremde Webseite weiterleitet.

Beschreibung

Bei der Login-Funktionalität zum Onlineshop kann mittels des Parameters `redirectURI` angegeben werden, auf welche Seite Benutzer nach erfolgter Anmeldung weitergeleitet werden sollen. Hierbei können jedoch beliebige URIs angegeben werden.

Angreifer könnten dadurch Benutzer der Webseite mittels dieser Open Redirect-Schwachstelle auf nicht vertrauenswürdige Webseiten weiterleiten.



Weiterleitung des Benutzers auf www.syslifters.com

Diese Schwachstelle könnte etwa im Zuge von Phishing-Angriffen missbraucht werden.

Ein Open Redirect ist eine Websicherheitslücke, die entsteht, wenn eine Anwendung Benutzereingaben auf unsichere Weise in das Ziel einer Weiterleitung einbindet. Eine Weiterleitung erfolgt, wenn eine Webanwendung die URL ändert, auf die der Client aktuell zugreift. Das Backend hat verschiedene Möglichkeiten Weiterleitungen durchzuführen, wie z.B. über entsprechende HTTP-Header (Status-Codes 30x) oder via JavaScript. Eine Anwendung ist anfällig für Open Redirects, wenn ein Angreifer in der Lage ist, innerhalb der Anwendung eine URL zu konstruieren, die eine Weiterleitung auf eine beliebige externe Domäne bewirkt.

Durch eine Open Redirect Schwachstelle kann ein Angreifer eine authentische Anwendungs-URL im Zuge eines Angriffs verwenden, welche die legitime Domäne referenziert und ein gültiges SSL-Zertifikat besitzt. Dieses Verhalten verleiht einem Phishing-Angriff mehr Glaubwürdigkeit und erleichtert damit die Durchführung von Social Engineering Angriffen gegen Benutzer. Open Redirects ermöglichen viele weitere, darauf aufbauende Angriffe.

Empfehlung

- Verzichtet wenn möglich auf Um- und Weiterleitungen in der Webanwendung. Falls ihr die Funktionalität benötigt, lasst keine benutzerdefinierten URLs zu.
- Wenn das Ziel einer Um- oder Weiterleitung von einem Benutzer bestimmt werden muss, verwendet alternativ Indizes anstatt vollständiger URL-Angaben (z. B. einen kurzen Namen, eine ID oder ein Token).
- Erzwingt, dass alle Weiterleitungen zu externen Ressourcen zunächst eine Seite durchlaufen, die den Benutzer darüber informiert, dass er Ihre Website verlässt. Diese Seite zeigt das Ziel deutlich an und der Benutzer muss zur Bestätigung auf einen Link klicken.
- Validiert alle Benutzereingaben und stellt sicher, dass nur für die Anwendung erwartete und gültige Eingaben akzeptiert werden. Potenziell bösartige Eingaben sollten verworfen werden.
- Akzeptiert nach Möglichkeit keine vollständigen URLs. URLs sind schwer zu validieren und der URL-Parser könnte je nach verwendeter Technologie missbraucht werden. Wenn dies dennoch notwendig ist, stellt sicher, dass nur gültige IP-Adressen oder Domänennamen akzeptiert werden.
- Verwendet den Allow-Listing Ansatz, wenn die Webanwendung nur zu identifizierten und vertrauenswürdigen URLs weiterleitet. Bei diesem Ansatz wird eine Liste von erlaubten IP-Adressen und Domänen auf Applikationsebene definiert. Die Webanwendung darf nach Validierung der Eingaben und Abgleich mit der Liste nur an jene Ressourcen weiterleiten, die auch in der Liste enthalten sind. Weiterleitungen zu Ressourcen außerhalb der Liste werden bei diesem Ansatz blockiert.
- Detaillierte Informationen und Hilfestellungen wie ihr Open Redirects verhindern könnt, findet ihr im verlinkten "Unvalidated Redirects and Forwards Cheat Sheet" von OWASP.

Weiterführende Informationen

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

5. Preisgabe von internen Informationen

Kritikalität: Medium
CVSS-Score: 5.3
Betrifft: http://www.example.com/.env
Empfehlung: Konfigurationsdateien sollten nicht öffentlich zugänglich sein. Die Datei `.env` sollte auf einen Ort außerhalb des Web-Roots verschoben werden.

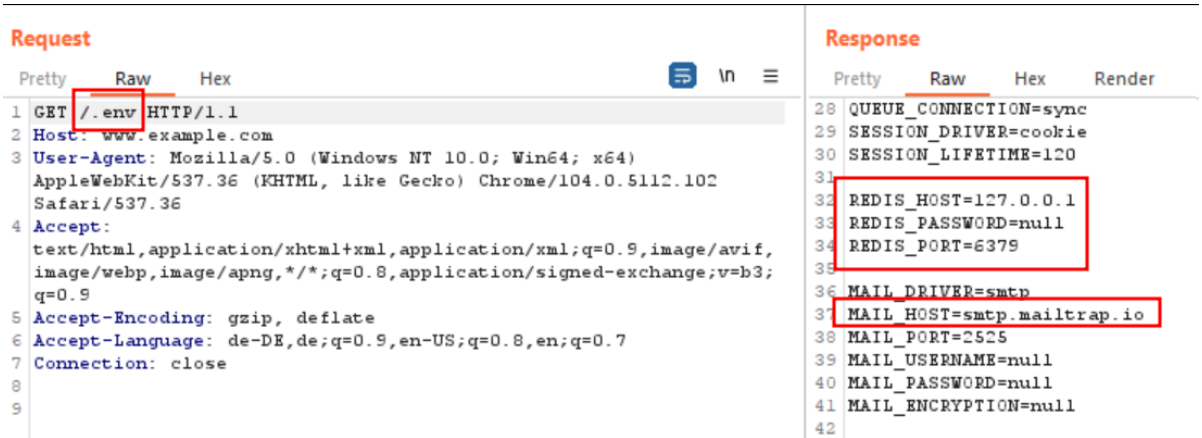
Überblick

Die Webanwendung offenbarte interne Informationen, die potenziell für weiterführende Angriffe genutzt werden könnten. Die Offenlegung von Informationen, auch bekannt als Information Disclosure, liegt vor, wenn ein System unbeabsichtigt interne Informationen an ihre Nutzer weitergibt.

Beschreibung

Wir konnten im Zuge der Tests eine Konfigurationsdatei mit internen Informationen herunterladen. Diese Informationen könnten für Folgeangriffe genutzt werden.

Im Web-Root der Web-Applikation war die Datei `.env` lesbar. Diese beinhaltete Informationen, wie etwa zum Redis-Server oder E-Mail-Gateway der Applikation.



Zugriff auf die Datei `.env`

Die Preisgabe dieser Informationen hat eher geringe Auswirkungen auf die Sicherheit der Applikation. Allerdings könnten sie wichtige Schlüsselinformationen für zielgerichtete Folgeangriffe eines Angreifers darstellen.

Empfehlung

- Konfigurationsdateien sollten nicht öffentlich zugänglich sein.
- Die Datei `.env` sollte auf einen Ort außerhalb des Web-Roots verschoben werden.

6. Schwächen im Session-Management

Kritikalität: Low
CVSS-Score: 3.6
Empfehlung: Benutzer sollten nach einer gewissen Zeit der Inaktivität automatisch abgemeldet werden.

Überblick

Wir konnten Schwächen im Sessions-Management der Webapplikation identifizieren. Die Sessions der Benutzer waren ohne zeitliche Einschränkung nutzbar und erforderten daher zu keinem Zeitpunkt eine Neuauthentifizierung. Personen mit Zugriff auf ein Computersystem könnten diesen Sachverhalt ausnutzen, wenn sich ein anderer Benutzer zuvor bei der Anwendung nicht explizit abgemeldet hat.

Beschreibung

Wir konnten feststellen, dass Benutzersessions ohne zeitliche Einschränkung nutzbar waren. Dies könnte es Angreifern erlauben, Benutzersessions zu übernehmen, die zuvor nicht explizit abgemeldet wurden.

Dies könnte etwa möglich sein, indem eine dritte Person den Computer eines Benutzers bedienen kann, in welchem eine Session weiterhin aktiv ist. Darüber hinaus könnte es für Angreifer möglich sein, beim Bekanntwerden von Session-Tokens (etwa über Log-Dateien; lokal oder etwa auf Proxy-Servern, etc) diese wiederzuverwenden.

Empfehlung

- Benutzersessions in Webapplikationen sollten nach einer bestimmten Zeit der Inaktivität automatisch ablaufen.
- Je nach Kritikalität der Benutzerberechtigung und der Applikation könnte das Timeout etwa zwischen einer Stunde und einem Tag liegen.

Änderungsverzeichnis

Version	Datum	Beschreibung	Autor
0.9	2022-08-26	Initiale Erstellung	Aron Molnar
1.0	2022-08-29	Review und Freigabe	Christoph Mahrl

Disclaimer

Wir können nicht garantieren, dass alle vorhandenen Schwachstellen und Sicherheitsrisiken tatsächlich entdeckt wurden. Das ist den beschränkten Zeitressourcen und dem limitierten Wissen der Pentester über die IT-Infrastruktur, Software, Source-Code, Benutzer etc. geschuldet. Eine umfassende Zusammenarbeit zwischen Auftraggeber und Penetration Testern erhöht die Effizienz des Penetration Tests. Das umfasst zum Beispiel die Offenlegung von Details interner Systeme oder die Provisionierung von Test-Benutzern.

Dieser Penetration Test stellt eine Momentaufnahme zum Zeitpunkt der Prüfung dar. Es lassen sich keine zukünftigen Sicherheitsrisiken davon ableiten.

Impressum

syslifters.com | **Dedicated to Pentests.**
Syslifters GmbH | Eitzersthal 75 | 2013 Göllersdorf
FN 578505 v | Bezirksgericht Hollabrunn