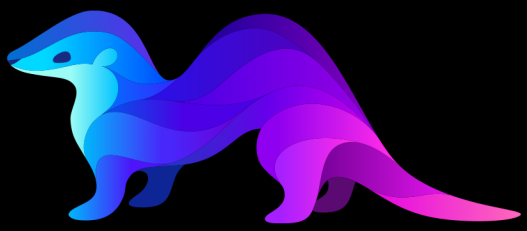

lutra security

ASSESSMENT REPORT

Beispielprojekt

Version 1.1

10.05.2023

Öffentlich

Inhaltsverzeichnis

Klassifikation: Öffentlich

Änderungsverlauf

Name	Datum	Version	Kommentar
Max Mustermann	25.04.2023	1.0	Initiale Berichtlieferung
Max Mustermann	10.05.2023	1.1	Anpassung nach Ergebnisbesprechung

Anhänge

Mit dem vorliegenden Dokument wurden weitere Dokumente geliefert:

Dateiname	Kommentar
scan_results.xlsx	Aggregierte Scanergebnisse

Kontakte

Name	Firma	Rolle	E-Mail
Erika Musterfrau	Example GmbH	CISO	erika.musterfrau@example.com
Max Mustermann	Lutra Security GmbH	Pentester	max.mustermann@lutrasecurity.com

1 EXECUTIVE SUMMARY

Example GmbH beauftragte Lutra Security mit der Durchführung eines Penetrationstests der DVWA-Webanwendung. Es handelte sich um einen Grey-Box-Ansatz, bei dem dem Tester nur begrenzte Informationen zur Verfügung gestellt wurden. Darüber hinaus wurden im Vorfeld Schwerpunkte festgelegt, auf die sich der Test konzentrieren sollte. Zudem sollte der Test beantworten, ob es einem Angreifer möglich ist, den darunterliegenden Server zu übernehmen.

In der getesteten Webanwendung konnten im vereinbarten Zeitraum insgesamt 9 Schwachstellen gefunden werden, von denen 6 mit einem hohen bzw. kritischen Risiko bewertet wurden.

So ist es möglich, dass eingeloggte Benutzer durch eine SQL-Injection die gesamte Datenbank der Anwendung auslesen können. Darüber hinaus erlauben weitere Schwachstellen das Ausführen beliebiger Befehle auf dem Server über Command Injection bzw. einen Upload schädlicher Dateien und damit die vollständige Kompromittierung des Servers.

Bei diesen drei Schwachstellen besteht dringender Handlungsbedarf, da sie einem normalen Benutzer die Übernahme der Datenbank oder des gesamten Servers ermöglichen.

Darüber hinaus ist es über eine Cross-Site-Scripting-Schwachstelle möglich, Benutzer der Anwendung anzugreifen, die auf einen vom Angreifer präparierten Link klicken. Dies kann dazu führen, dass der Angreifer Aktionen im Namen des Opfers ausführt oder Zugriff auf die Daten des betroffenen Benutzers erhält.

Um den Applikationsserver zusätzlich zu härten, wird empfohlen, unverschlüsselte Kommunikation zu unterbinden und unsichere Verschlüsselungstechnologien nicht zu verwenden.

Außerdem sollte die verwendete Software aktualisiert werden, um weitere öffentlich bekannte Sicherheitslücken zu schließen. Die Konfiguration des Webservers sollte ebenfalls angepasst werden, um Informationslecks und detaillierte Fehlermeldungen zu vermeiden. Dadurch wird verhindert, dass ein Angreifer Informationen über das System erhält, die ein normaler Benutzer für die normale Nutzung der Webanwendung nicht benötigt.

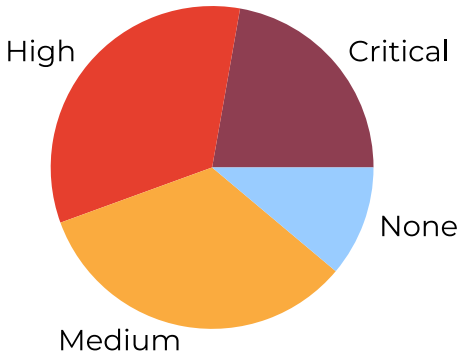

Figure 1: Verteilung der Risiken der einzelnen Befunde.

1.1 Scope

Example GmbH hat Lutra Security beauftragt, die Webanwendung DVWA (https://webapp.example.com/) auf Schwachstellen zu untersuchen. Folgende Bedingungen wurden vereinbart

Zeitraum	4 Tage
Testverfahren	Greybox
Schwachstellenscans	Aktiv
Portscans	Ja

Darüber hinaus wurde nach Absprache ein besonderer Schwerpunkt auf die Suche nach Injection-Schwachstellen wie SQL-Injections oder Command Injections gelegt.

1.2 Methodik

Beim Test von Webapplikationen und Webservices orientieren wir uns an etablierten Methodologien wie dem OWASP Web Security Testing Guide[1] und beziehen IT-Security Best Practices sowie interne Methodologien mit ein.

Dabei werden automatisierte Tools mit einer weitgehend manuellen Überprüfung kombiniert, um eine maximale Abdeckung bei gleichzeitig sehr hoher Effizienz zu gewährleisten.

1.3 Projekttimeline

Datum	Milestone
06.03.2023	Scoping
29.03.2023	Kickoff
17.04.2023 - 20.04.2023	Penetrationstest
25.04.2023	Berichtlieferung
08.05.2023	Ergebnisbesprechung

1.4 Disclaimer

Dieser Bericht stellt eine Momentaufnahme der Sicherheitslage des getesteten Systems dar und sollte auch als solche behandelt werden. Es werden jeden Tag neue Schwachstellen entdeckt und die Informationssicherheit als Ganzes entwickelt sich stetig weiter, weshalb auch nach dem Beheben der dokumentierten Befunde keine Garantie auf vollständige Sicherheit des Systems gegeben werden kann. Das Assessment und der vorliegende Bericht wurden nach bestem Wissen und Gewissen durchgeführt bzw. verfasst.

In der Regel gilt, dass innerhalb einer wirtschaftlichen Analyse keine “vollständige” Abdeckung erreicht werden kann. Lutra Security ist jedoch bestrebt, im Vorfeld alle möglichen Optionen zu berücksichtigen, um innerhalb eines begrenzten Zeitrahmens die größtmögliche Abdeckung zu gewährleisten.

2 BEFUNDE

Eine Übersicht über die gefundenen Schwachstellen finden Sie in der folgenden Tabelle:

Titel	CVSS-Score	Status
Command Injection	8.8	Offen
Upload schädlicher Dateien	8.8	Offen
SQL-Injection (UserID Search)	9.8	Offen
Reflected Cross-Site-Scripting	6.1	Offen
TLS-Konfiguration webapp.example.com:443	4.2	Offen
Unverschlüsselte Kommunikation über HTTP	8.1	Offen
Detaillierte Fehlermeldungen	4.3	Offen
Informationsleck (HTTP Header)	0.0	Offen
Veraltete Software: PHP & Apache	9.8	Offen

2.1 Command Injection

CVSS v3.1 Vektor	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
Impact Subscore	5.9
Exploitability Subscore	2.8
Status	Offen

8.8

Command Injection bezeichnet eine Schwachstelle in Software, die es einem Angreifer ermöglicht, Befehle an das System zu senden und auszuführen. Dadurch kann der Angreifer beispielsweise unberechtigten Zugriff auf das System erlangen, Daten manipulieren oder das System komplett übernehmen.

Über die Funktion *Network Ping* der Webanwendung können beliebige Befehle ausgeführt werden. Die entsprechende Funktionalität kann über den folgenden Endpunkt aufgerufen werden:

- http://webapp.example.com/vulnerabilities/exec/

Die *Network Ping* Funktion ermöglicht es Benutzern, jeden beliebigen Server zu pingen. Zum Beispiel kann der Server `8.8.8.8` gepingt werden, indem der folgende POST-Request an den Server gesendet wird:

```
1 POST /vulnerabilities/exec/ HTTP/1.1
2 Host: webapp.example.com
3 ...
4 Connection: close
5
6 ip=8.8.8.8&Submit=Submit
```

Der Parameter `ip` gibt den Server an, der angepingt werden soll (siehe fig. 2).

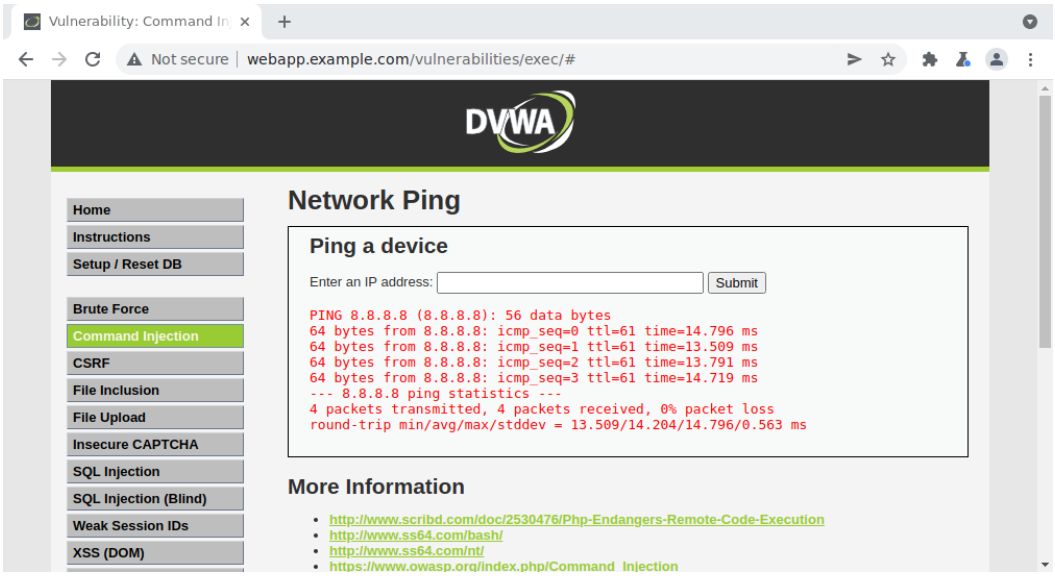

Figure 2: Der Server mit der IP Adresse `8.8.8.8` wurde angepingt.

Mit einem `;` im Feld der IP-Adresse können beliebige andere Befehle ausgeführt werden. Wird beispielsweise die Payload `; id` verwendet, antwortet der Server mit den Benutzerinformationen des Webservers (siehe fig. 3). Dies liegt daran, dass Unix/Linux-Systeme Semikolons verwenden, um Befehle nacheinander auszuführen, und dass die Benutzereingabe an das darunter liegende System weitergegeben wird.

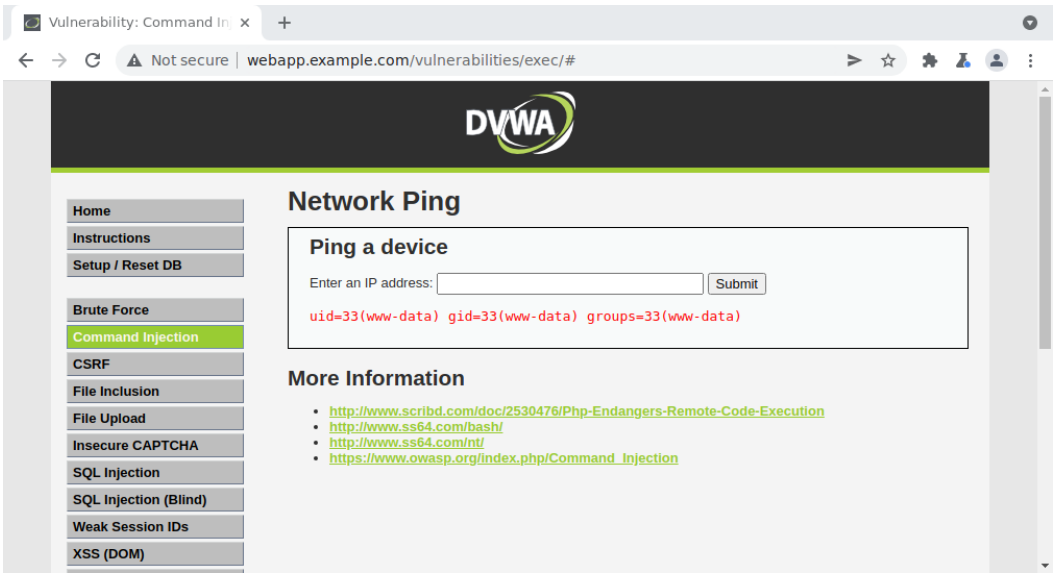

Figure 3: Antwort der Webseite auf die Payload `; id` im IP-Adressfeld. Das `;` in der Payload bewirkt, dass der nachfolgende Befehl `id` ausgeführt wird. Innerhalb der Webanwendung wird anschließend die Ausgabe des Befehls angezeigt.

2.1.1 Handlungsempfehlungen

- Stellen Sie sicher, dass der Parameter `ip` tatsächlich nur eine IP enthält. Eine Möglichkeit besteht darin, einen geeigneten regulären Ausdruck zu verwenden.
- Benutzereingaben sollten nicht ohne sorgfältige Validierung und Escaping in Befehlen verwendet werden.
- Wenn möglich, verwenden Sie eine erprobte Bibliothek, um diese Funktionalität zu implementieren, und vermeiden Sie Eigenentwicklungen, die Systembefehle verwenden.

2.1.2 Referenzen

- Command Injection - OWASP

2.2 Upload schädlicher Dateien

CVSS v3.1 Vektor	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
Impact Subscore	5.9
Exploitability Subscore	2.8
Status	Offen

8.8

Fehlende oder unzureichende Dateityp-Prüfungen beim Hochladen von Dateien ermöglichen es einem Angreifer, unerwünschte Dateien auf das System der Anwendung hochzuladen. Das Risiko besteht darin, dass der Angreifer dadurch Zugriff auf das System erhält, Daten stehlen, das System vollständig übernehmen oder diese schädlichen Dateien an Benutzer der Anwendung verteilen kann.

Die untersuchte Webanwendung verfügt über einen Datei-Upload, über den ausführbare Skripte hochgeladen und ausgeführt werden können.

Konkret ist folgende URL betroffen:

- http://webapp.example.com/vulnerabilities/upload/

Über diese Schnittstelle können beliebige Dateien hochgeladen werden (siehe fig. 4). Beispielsweise kann eine PHP-Webshell hochgeladen werden, die es einem Angreifer ermöglicht, beliebige Befehle auf dem System auszuführen (siehe fig. 5).

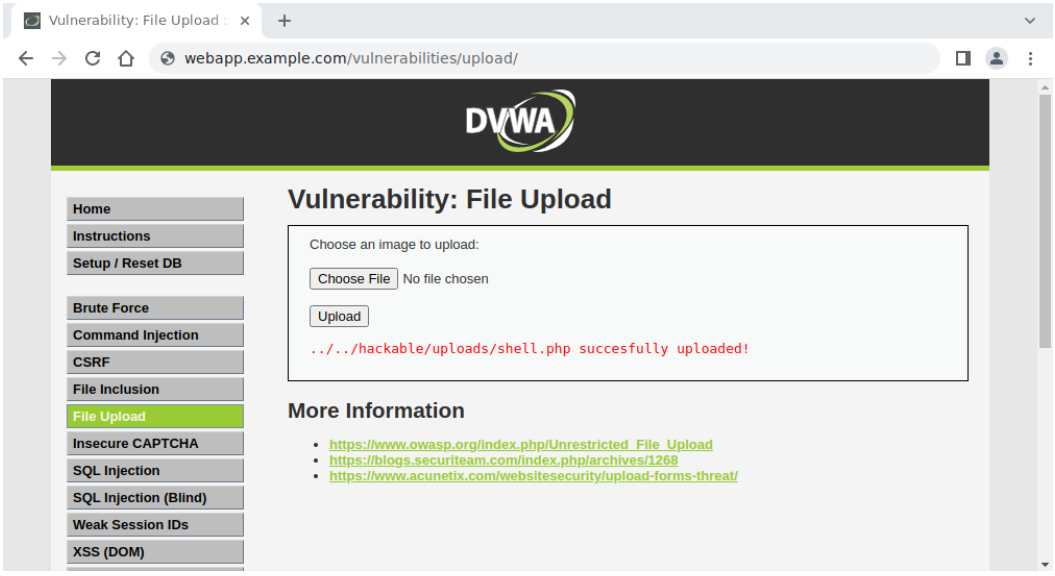

Figure 4: Der Upload der PHP-Datei ist erfolgreich.

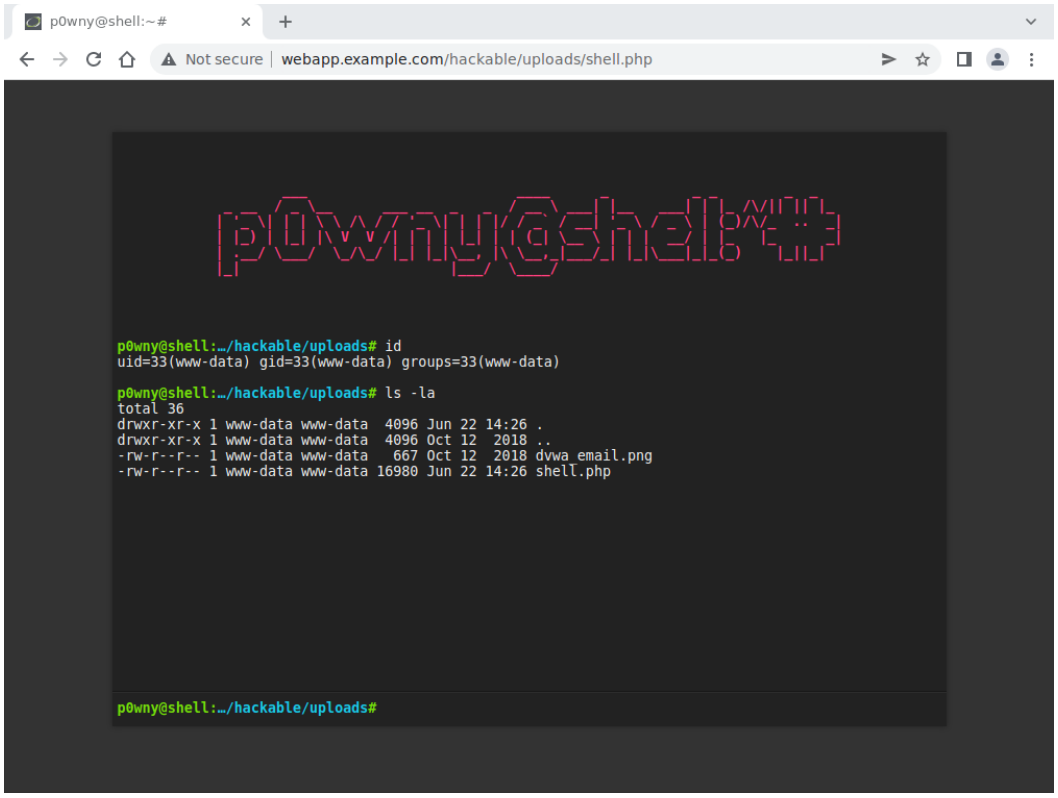

Figure 5: Die hochgeladene PHP-Webshell kann verwendet werden, um Befehle auf dem Server auszuführen.

2.2.1 Handlungsempfehlungen

- Stellen Sie serverseitig sicher, dass nur die gewünschten Dateitypen hochgeladen werden können. Dazu müssen u.A. die Dateiendung, die Magic Bytes und der Content-Type-Header überprüft werden.
- Die hochgeladenen Dateien dürfen nicht auf dem Server ausgeführt werden. Daher sollte die Ausführung von Code im Upload-Ordner unterbunden werden.

2.2.2 Referenzen

- Unrestricted File Upload - OWASP

2.3 SQL-Injection (UserID Search)

CVSS v3.1 Vektor	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Impact Subscore	5.9
Exploitability Subscore	3.9
Status	Offen

9.8

SQL-Injection tritt auf, wenn Benutzereingaben unzureichend überprüft werden, wodurch ein Angreifer schädlichen Code in die Datenbank der Anwendung einschleusen kann. Diese Schwachstelle kann zu einer vollständigen Kompromittierung der Datenbank und der darin gespeicherten Informationen führen. Unter Umständen kann eine SQL-Injection auch zur Kompromittierung des zugrundeliegenden Servers führen.

Konkret betroffen sind die *UserID Search* innerhalb der Anwendung sowie der Login der Anwendung. Letzteres ermöglicht es einem nicht authentifizierten Angreifer, die Schwachstelle auszunutzen.

Im Folgenden soll die SQL-Injection-Schwachstelle am Beispiel der *UserID Search* veranschaulicht werden:

- http://webapp.example.com/vulnerabilities/sqli/?id=2&Submit=Submit#

Die *UserID Search* kann verwendet werden, um nach einem Benutzer mit einer UserID zu suchen (siehe fig. 6). Der verwendete Parameter `id` ist anfällig für SQL-Injection.

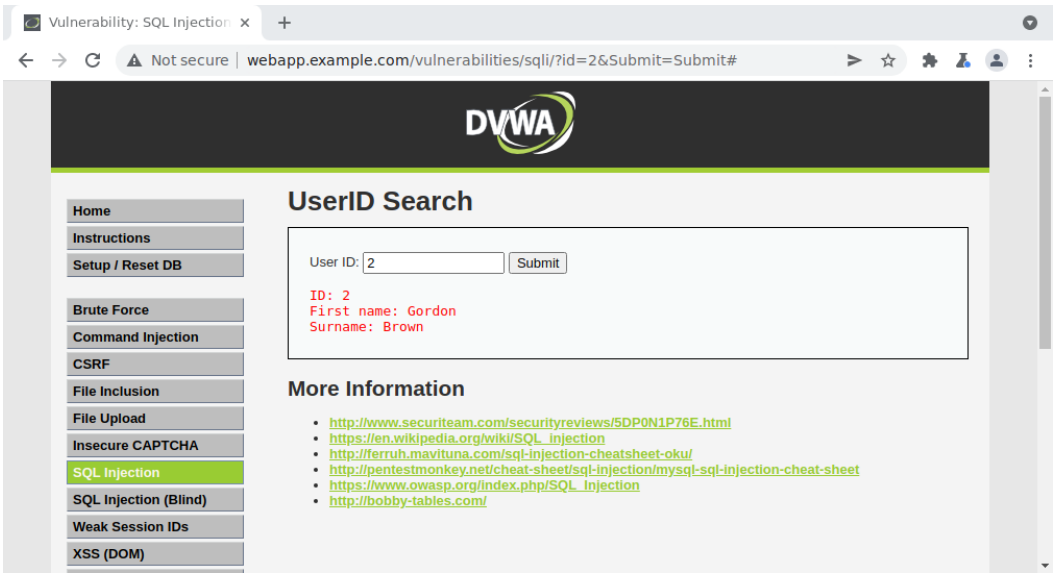

Figure 6: Suche nach dem Benutzer mit der ID `2`.

Wird im Feld UserID ein `'` angegeben, reagiert die Anwendung mit einer SQL-Fehlermeldung: `You have an error in your SQL syntax`. Dies ist ein erster Hinweis darauf, dass die Schnittstelle anfällig für SQL Injection ist.

Um zu verifizieren, dass es sich um eine SQL-Injection handelt, wird die Payload `X' OR 1=1;-- -` verwendet, die eine wahre Anweisung (`1=1`) enthält. Wie in fig. 7 zu sehen ist, gibt die Anwendung die gesamte Benutzerdatenbank aus. Wird hingegen die Payload `X' OR 1=0;-- -` verwendet, gibt die Anwendung keinen Benutzer zurück (siehe fig. 8). Aus diesem Verhalten kann geschlossen werden, dass der Endpunkt anfällig für SQL Injection ist.

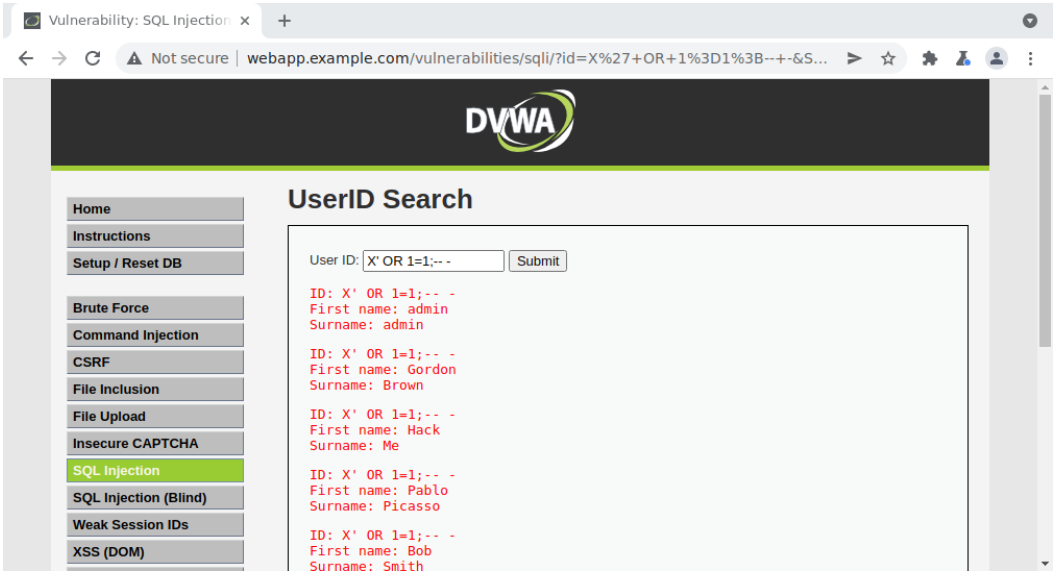

Figure 7: Wenn die Payload `X' OR 1=1;-- -` verwendet wird, gibt die Anwendung die gesamte Benutzerdatenbank aus, da `OR 1=1` in jedem Fall als wahr ausgewertet wird.

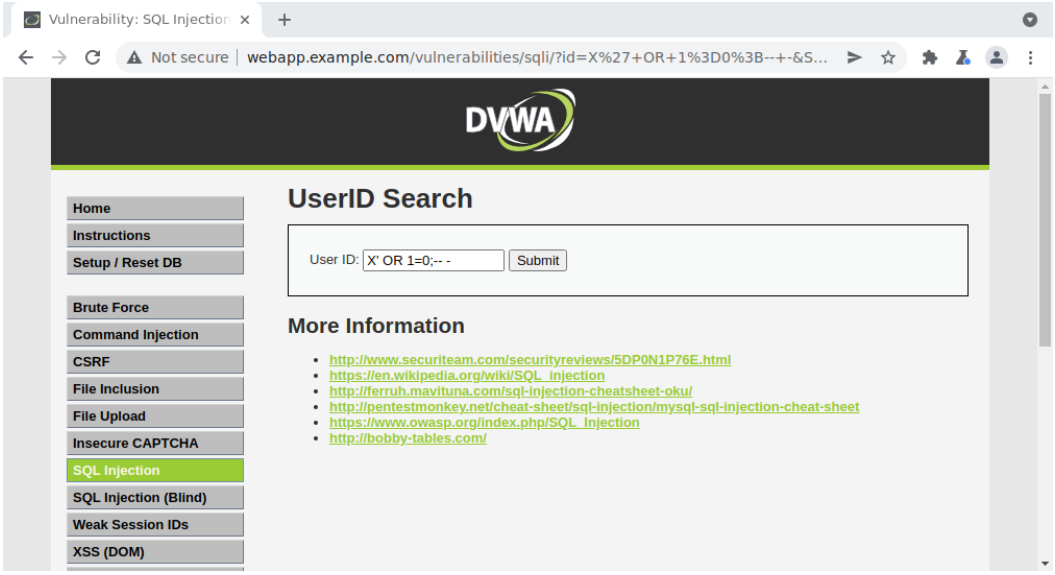

Figure 8: Wenn die Payload `X' OR 1=0;-- -` verwendet wird, gibt die Anwendung keinen Benutzer zurück, da `1=0` ein falscher Ausdruck ist.

Als weiterer Proof-of-Concept wurde der Datenbank-Banner extrahiert, der den verwendeten Datenbankserver und die eingesetzte Version enthält: `10.1.26-MariaDB-0+deb9u1`. Dazu wurde das Programm `sqlmap`[2] verwendet.

Alternativ kann der folgende String in das Feld UserID eingegeben werden: `' UNION SELECT NULL, @@version;-- -`.

2.3.1 Handlungsempfehlungen

- Verwenden Sie Prepared Statements und Parameter Binding anstelle von direkten SQL-Abfragen, um sicherzustellen, dass Benutzereingaben sicher in SQL-Anweisungen eingebettet sind.
- Prüfen und filtern Sie Benutzereingaben, um sicherzustellen, dass sie dem erwarteten Format und Datentyp entsprechen und keine schädlichen Zeichen enthalten.

2.3.2 Referenzen

- SQL Injection - OWASP
- SQL Injection Prevention - OWASP

2.4 Reflected Cross-Site-Scripting

CVSS v3.1 Vektor	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
Impact Subscore	2.7
Exploitability Subscore	2.8
Status	Offen

6.1

Cross-Site-Scripting ist ein Angriff auf den Benutzer einer Webanwendung. Dabei wird JavaScript vom Angreifer in eine Webanwendung eingefügt und im Browser des Opfers ausgeführt. Dadurch hat der Angreifer vollen Zugriff auf die Webanwendung als der betroffene Benutzer und kann in diesem Rahmen beliebige Aktionen ausführen oder Daten auslesen.

Ein Cross-Site-Scripting Angriff ist über die folgende URL möglich:

- http://webapp.example.com/vulnerabilities/xss_r/?name=test

Der Inhalt des Parameters `name` wird unverändert auf der Webseite ausgegeben. Dies ermöglicht es einem Angreifer, JavaScript in den Parameter einzufügen, das im Browser des Opfers ausgeführt wird. Beispielsweise kann mit der Payload `test%3Cscript%3Ealert%28%27reflected+XSS%27%29%3C%2Fscript%3E` ein Warnfenster angezeigt werden (siehe fig. 9).

Das Problem kann mit Hilfe der folgenden URL reproduziert werden:

http://webapp.example.com/vulnerabilities/xss_r/?name=test%3Cscript%3Ealert%28%27reflected+XSS%27%29%3C%2Fscript%3E#

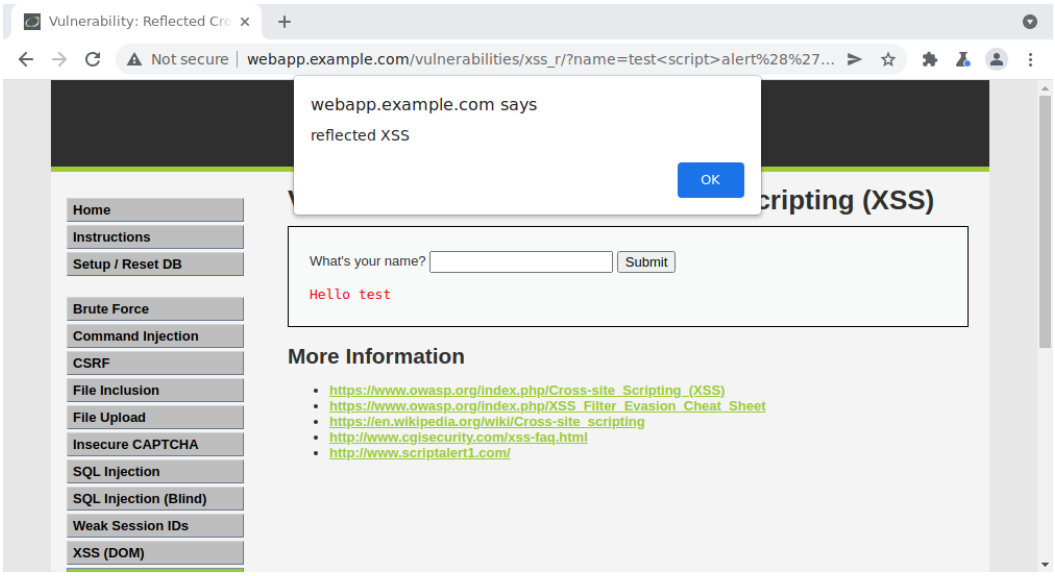

Figure 9: Der JavaScript-Code aus der Payload wird im Browser ausgeführt und ein Alarmfenster erscheint.

2.4.1 Handlungsempfehlungen

- Verwenden Sie HTML-Sanitization, um unerwünschte oder schädliche HTML-Tags, Attribute und Eigenschaften aus Benutzereingaben zu entfernen, bevor diese auf der Seite dargestellt werden.
- Verwenden Sie Content Security Policies (CSPs), um festzulegen, welche Ressourcen von welchen Quellen auf eine Seite geladen werden dürfen.

2.4.2 Referenzen

- Cross Site Scripting (XSS) - OWASP
- Cross Site Scripting Prevention: Output Encoding - OWASP

2.5 TLS-Konfiguration webapp.example.com:443

CVSS v3.1 Vektor	AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N
Impact Subscore	2.5
Exploitability Subscore	1.6
Status	Offen

4.2

Die Verschlüsselung von HTTP-Datenverkehr mittels TLS gewährleistet sowohl Integrität als auch Vertraulichkeit der Kommunikation und ist damit eine der grundlegenden Sicherheitsvorkehrungen für jede Webanwendung.

Eine Analyse der unterstützten Cipher Suites ergab Sicherheitsbedenken in den folgenden Fällen:

Table 1: Unsichere Cipher Suites für webapp.example.com:443

Cipher Suite	Problem
TLS_RSA_WITH_AES_256_CBC_SHA	Keine PFS
TLS_RSA_WITH_AES_128_GCM_SHA256	Keine PFS
TLS_RSA_WITH_AES_128_CBC_SHA	Keine PFS
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	CBC
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	CBC
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	CBC
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	CBC

2.5.1 Handlungsempfehlungen

- Deaktivieren Sie die in tbl. 1 aufgeführten Cipher Suites.

2.5.2 Referenzen

- CWE-1240: Use of a Cryptographic Primitive with a Risky Implementation

2.5.3 Glossar

CBC

CBC (Cipher Block Chaining) ist ein Betriebsmodus, der z. B. anfällig für Padding-Oracle-Angriffe ist. Modernere Betriebsmodi, wie z. B. GCM, bieten einen besseren Schutz und zusätzliche Sicherheitsmechanismen, wie AEAD.

Keine PFS

Perfect Forward Secrecy (PFS) ist eine Eigenschaft eines Schlüsselaustauschs, die sicherstellt, dass die Kompromittierung eines Schlüssels in der Zukunft die Vertraulichkeit der Kommunikation in der Gegenwart nicht gefährden kann.

2.6 Unverschlüsselte Kommunikation über HTTP

CVSS v3.1 Vektor	AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N
Impact Subscore	5.2
Exploitability Subscore	2.8
Status	Offen

8.1

Eine unverschlüsselte Verbindung über HTTP ermöglicht es einem Angreifer, den Datenverkehr abzufangen, zu manipulieren und vertrauliche Informationen wie Passwörter oder Kreditkarteninformationen zu stehlen. Die Verschlüsselung des HTTP-Datenverkehrs mit TLS gewährleistet sowohl Integrität als auch Vertraulichkeit und ist daher eine der grundlegenden Sicherheitsmaßnahmen für jede Webanwendung.

Bei der Überprüfung der Webanwendung konnten folgende Webseiten identifiziert werden, die nur unverschlüsselt über HTTP erreichbar sind:

- http://example.com
- http://example.org
- http://example.net
- http://www.example.com
- http://www.example.org
- http://www.example.net

Die genannten Domains werden im Rahmen der Webanwendung und API zur Verarbeitung von Kundendaten verwendet und sollten auf keinen Fall unverschlüsselt kommunizieren.

2.6.1 Handlungsempfehlungen

- Stellen Sie sicher, dass alle Webanwendungen über HTTPS erreichbar sind.
- Richten Sie eine Weiterleitung von HTTP auf HTTPS ein oder deaktivieren Sie den unverschlüsselten Zugriff über HTTP.
- Setzen Sie den HSTS-Header, der den Browser anweist, die Verbindung künftig immer über HTTPS aufzubauen.

2.6.2 Referenzen

- CWE-319: Cleartext Transmission of Sensitive Information

2.7 Detaillierte Fehlermeldungen

CVSS v3.1 Vektor	AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N
Impact Subscore	1.4
Exploitability Subscore	2.8
Status	Offen

4.3

Detaillierte Fehlermeldungen sind Fehlermeldungen, die mehr Informationen enthalten, als ein Benutzer für den normalen Gebrauch der Anwendung benötigt. Solche Fehlermeldungen enthalten oft sensible Daten, die es einem Angreifer erleichtern können, bestimmte Schwachstellen zu identifizieren und auszunutzen. Darüber hinaus können sie dem Angreifer wertvolle Informationen über das System liefern, die im weiteren Verlauf des Angriffs genutzt werden können.

Detaillierte Fehlermeldungen werden anwendungsweit angezeigt. Das Problem wird am Beispiel des folgenden Endpunkts erläutert:

- http://webapp.example.com/vulnerabilities/sqli/

Fehlermeldungen für diesen Endpunkt können z.B. durch Senden des folgenden HTTP-Requests an den Server hervorgerufen werden:

```
GET /vulnerabilities/sqli/?id=1%27&Submit=Submit HTTP/1.1
Host: webapp.example.com
...
Connection: close
```

Im Parameter `id` wurde ein Hochkomma `'` (im HTTP-Request als `%27` kodiert) übergeben, was zu folgender Fehlermeldung führt:

```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server ver
        sion for the right syntax to use near ''1''' at line 1
```

Ein Angreifer kann dies beispielsweise nutzen, um Informationen über die verwendete SQL Query zu erhalten und damit die vorhandene SQL Injection leichter auszunutzen.

2.7.1 Handlungsempfehlungen

- Verwenden Sie eine allgemeine Fehlermeldung, die den Benutzer darauf hinweist, dass ein Fehler aufgetreten ist, ohne unnötige Einzelheiten über die Ursache oder den Ort des Fehlers preiszugeben. Auf diese Weise kann der Benutzer

das Problem identifizieren, ohne das potentielle Schwachstellen oder vertrauliche Informationen preisgegeben werden.
- Validieren und filtern Sie alle Benutzereingaben sorgfältig, um sicherzustellen, dass unerwartete Eingaben oder bösartiger Code nicht von der Anwendung weiterverarbeitet werden. Wenn die Anwendung Eingaben ordnungsgemäß validiert, kann verhindert werden, dass ein Angreifer die Anwendung ausnutzt, um detaillierte Fehlermeldungen zu erzeugen.

2.7.2 Referenzen

- CWE-209: Generation of Error Message Containing Sensitive Information

2.8 Informationsleck (HTTP Header)

CVSS v3.1 Vektor	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Impact Subscore	0.0
Exploitability Subscore	3.9
Status	Offen

0.0

Wenn vertrauliche oder geschützte Informationen durch eine Anwendung oder ein System entweder absichtlich oder versehentlich offengelegt werden, spricht man von einem Informationsleck. Diese Informationen können dann in nachgelagerten Angriffen von einem Angreifer genutzt werden, um beispielsweise gezielt nach Exploits für die verwendeten Softwarekomponenten zu suchen.

Der Webserver offenbart durch den `Server` sowie den `X-Powered-By` Header die verwendeten Softwareversionen. Der folgende Webserver ist betroffen:

- http://webapp.example.com/

Wenn die URL http://webapp.example.com/ aufgerufen wird, antwortet der Server unter anderem mit folgenden Headern:

```
HTTP/1.1 200 OK
Date: Wed, 22 Jun 2022 10:20:30 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.1.6
...
```

Die HTTP-Header `Server: Apache/2.2.3 (CentOS)` und `X-Powered-By: PHP/5.1.6` offenbaren, welche Software in welcher Version verwendet wird. Sowohl Apache als auch PHP werden in einer stark veralteten Version verwendet.

2.8.1 Handlungsempfehlungen

- Vermeiden Sie es, dem Benutzer Informationen über die verwendete Software oder sogar Softwareversionen zu liefern. Stattdessen sollte ein generischer Wert, z.B. `Server: webserver` verwendet werden oder der Header ganz weggelassen werden.
- Im Falle von Apache kann dies durch Setzen der `ServerTokens` Direktive auf `Prod` erreicht werden, siehe https://httpd.apache.org/docs/2.4/mod/core.html#servertokens

2.8.2 Referenzen

- Information Disclosure Vulnerabilities - Portswigger
- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

2.9 Veraltete Software: PHP & Apache

9.8

CVSS v3.1 Vektor	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Impact Subscore	5.9
Exploitability Subscore	3.9
Status	Offen

Legacy-Systeme stellen für viele Unternehmen ein erhebliches Sicherheitsrisiko dar. Diese Systeme verfügen häufig über veraltete Hardware oder Software. Viele enthalten auch bekannte Schwachstellen, die sie anfällig für Hackerangriffe machen. Die Situation verschärft sich oft noch, wenn die Hersteller den Support für diese Systeme einstellen.

Für Unternehmen ist es daher wichtig, die potenziellen Sicherheitsrisiken ihrer Systeme zu bewerten und Patch-Strategien oder Maßnahmen zur Risikominderung zu implementieren.

Durch ein Informationsleck konnten die folgenden Softwareversionen identifiziert werden:

- `PHP/5.1.6` (Releasedatum: 24.08.2006): Bekannte Schwachstellen
- `Apache/2.2.3` (Releasedatum: 28.07.2006): Bekannte Schwachstellen

Beide Versionen sind stark veraltet (End-of-Life) und enthalten zum Teil kritische Schwachstellen, weshalb eine rasche Aktualisierung empfohlen wird.

2.9.1 Handlungsempfehlungen

- Aktualisieren Sie Apache auf die neueste Version.
- Aktualisieren Sie PHP auf die neueste Version.

2.9.2 Referenzen

- Vulnerable and Outdated Components - OWASP Top 10
- CWE-1104: Use of Unmaintained Third Party Components

3 REFERENZEN

1. Der Web Security Testing Guide ist ein umfassender Leitfaden zum Testen der Sicherheit von Webanwendungen und Web-Services. Siehe https://owasp.org/www-project-web-security-testing-guide/.↩

2. Sqlmap ist ein Open-Source-Penetrationstest-Tool, das den Prozess der Erkennung und Ausnutzung von SQL-Injection-Fehlern und der Übernahme von Datenbankservern automatisiert. Siehe auch https://sqlmap.org/↩

https://lutrasecurity.com
hello@lutrasecurity.com
+49 89 2152 5883-0

lutra security

NACHHALTIG SICHERER