IT-SecLab: Vulnerability Assessment

Advisory zu der Anwendung Passierschein A38

Hans Lenard Lorenz Mollenkopf, Matr. 456308

3. Mai 2022

Inhaltsverzeichnis

1 Stored XSS über Abgabe eines Antrages

1.1 Zusammenfassung

`Schwere: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:N  Wertung: 9.3`

Bei der Abgabe eines Antrages kann ein Nutzer diesen Begründen. Ein Bedrohungsakteur kann die dafür vorgesehen Eingabe nutzen, um beliebigen Javascript Code in dem Browsers des bearbeitenden Sachbearbeiters auszuführen.

1.2 Beschreibung

Wenn ein Nutzer einen Antrag abgibt, werden die von ihm angebenden Daten in einer dazugehörigen Datei gespeichert, die angegebene Begründung wird dazu Url-codiert. Wenn nun ein Sachbearbeiter diesen Antrag sichtet, werden diese Daten wiederum für ihn ausgegeben, die Begründung wir dabei wieder Decodiert.
In keinem dieser Schritte wird sichergestellt, dass die Ausgabe dieser Information nicht dazu führen kann, dass diese von einem Browser als HTML interpretiert wird. Dies erlaubt es einem Bedrohungsakteur z.B. ein Script-Tag der dem Sachbearbeiter angezeigten Seite hinzuzufügen.

1.3 Wertung

Diese Schwachstelle kann über das Internet ausgenutzt werden, hat nur eine geringfügige Komplexität, benötigt keinerlei spezieller Berechtigungen und erlaubt es einem jegliche Daten die dem Sachbearbeiter angezeigt werden zu Modifizieren, sowie jegliche Aktion zu tätigen die der Sachbearbeiter tätigen kann. Damit ist dies eine schwere Lücke, dies spiegelt sich auch in der CVSS 3.0 Wertung von 9.3 wieder.
Diese Lücke fällt in den OWASP Top Ten in die Kategorie „Injection".

1.4 Lösung

Um die Schwachstelle (und einige ähnliche) zu beheben muss an jeder Stelle, an der Daten eines Nutzers ausgeben werden Sichergestellt werden, dass diese von dem Browser nur als Text interpretiert werden. PHP stellt dafür die Funktion `htmlspecialchars` bereit. Beispielhaft müsste der Code für die Ausgabe eines Antrages von

```
<?php
...
echo "<ul><li><B>Applicant:</B><ul><li>".$fname." ".$name."</li><li>".
    $str.", ".$plz." ".$ort."</li></ul></li><li><B>Reason for Application:</B>
    <ul><li>".urldecode($begr)."</li></ul></li><li>Current Processor:<ul>";
...
?>
```

zu

```
<?php
...
echo "<ul><li><B>Applicant:</B><ul><li>";
echo htmlspecialchars($fname." ".$name, ENT_QUOTES | ENT_HTML401);
echo "</li><li>";
echo htmlspecialchars($str.", ".$plz." ".$ort, ENT_QUOTES | ENT_HTML401);
echo "</li></ul></li><li><B>Reason for Application:</B><ul><li>";
echo htmlspecialchars(urldecode($begr), ENT_QUOTES | ENT_HTML401);
echo "</li></ul></li><li>Current Processor:<ul>";
...
?>
```

geändert werden.

2 Umgehung der Authentifizierung

2.1 Zusammenfassung

`Schwere: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N  Wertung: 9.1`

Durch Angabe eines nicht registriertem Benutzernamens und eines leerem Passworts ist es möglich auf das Backend der Anwendung zuzugreifen.

2.2 Beschreibung

Die registrierten Konten sind in der Anwendung in einem Array gespeichert. Bei dem Logins eines Nutzers wird überprüft, ob dieser einen Benutzernamen und ein Passwort angegeben hat, sofern dies gegeben ist, wird das an der Stelle des Benutzernamens im Array hinterlegte Passwort mit dem angegebenem verglichen. Stimmen diese Überein wird der Nutzer authentifiziert.
Dieses Array gibt aber falls, unter dem Benutzername kein Passwort gespeichert ist, `NULL` zurück. Da zu der Überprüfung der Gleichheit des Passwortes der Operator ,,`==`" verwendet wurde, wird dieser mit Type Juggling[1] durchgeführt. Somit ergibt der Vergleich zwischen dem leerem Passwort und `NULL` (`"" == NULL`) das diese gleich sind.
Da unter einem nicht registriertem Benutzernamen kein Passwort hinterlegt ist ist ein solcher Benutzername in Kombination mit einem leerem Passwort ausreichend um angemeldet zu werden.

2.3 Wertung

Diese Schwachstelle kann über das Internet ausgenutzt werden, hat nur eine geringfügige Komplexität, benötigt keinerlei spezieller Berechtigungen und erlaubt jegliche Aktion zu tätigen die ein Sachbearbeiter tätigen kann. Damit ist dies eine schwere Lücke, dies in der CVSS 3.0 Wertung von 9.1 wieder.
Diese Lücke fällt in den OWASP Top Ten in die Kategorie ,,Identification and Authentication Failures".

2.4 Lösung

Um die Schwachstelle zu beheben muss der Vergleichs des Passworts ohne Type Juggling durchgeführt werden, d.h. der Operator ,,`===`" ist zu verwenden. Im übrigen sollte jeder Operator ,,`==`" durch den Operator ,,`===`" ersetzt werden, um ähnlichen Schwachstellen vorzubeugen.

[1] `https://www.php.net/manual/de/language.types.type-juggling.php`

3 Anträge können von nicht zuständigen verarbeitet werden

3.1 Zusammenfassung

`Schwere: CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:H/A:N  Wertung: 7.7`

Sofern der Name eines Antrages bekannt ist, kann jeder Authentifizierte Nutzer diesen bearbeiten, unabhängig davon, ob er für diesen zuständig ist.

3.2 Beschreibung

Wenn ein Authentifizierte Nutzer eine Anfrage an `backend.php` sendet, in der je ein Wert für `antrag` und `forward` enthalten ist, wird der angegebene Antrag verarbeitet, dabei wird nicht überprüft, ob der Nutzer für diesen zuständig ist, also ob er für diesen Vorgang die Autorität besitzt.
Der dafür benötigten Namen des Antrages wird ihm zwar nicht angezeigt, da diese Namen aber aus Vor- und Nachnamen des Antragsteller bestehen und neue Anträge für jeden sichtbar sind, ist es ein leichtes diese zu erraten oder sich schlichtweg zu Merken.

3.3 Wertung

Diese Schwachstelle kann über das Internet ausgenutzt werden, hat nur eine geringfügige Komplexität, benötigt Authentifizierte und erlaubt jegliche Aktion an einem Antrag zu tätigen, unabhängig davon ob man für diesen zuständig ist.
Da diese Lücke nur ausnutzbar ist, wenn man bereits die Autorität hat andere Anträge zu Bearbeiten ist sie im Kontext einer Behörde nicht so schwerwiegend, wie die vorhergehenden Lücken. Ist aber die Auswirkung, wie die CVSS 3.0 Wertung von 7.7 zeigt, nicht zu vernachlässigen, es wird nämlich u.A. das ,,principle of least privilege" gebrochen.
Diese Lücke fällt in den OWASP Top Ten in die Kategorie ,,Broken Access Control".

3.4 Lösung

Um die Schwachstelle zu beheben muss genauso wie bei der Ausgabe der zu bearbeitenden Anträge geprüft werden, ob der Aktuelle Nutzer für diesen zuständig ist:

```
<?php
...
if (isset($_REQUEST["antrag"]) && isset($_REQUEST["forward"])) {
        $fw = fopen("antraege/".$_REQUEST["antrag"], "r");
        while (!feof($file) && (($temp = trim(fgets($file))) != "")) {
            $ma = $temp;
        }
        if($ma == $user || $ma == "New"){
```

```
                // Weiterleiten & Genehmigen (user=accepted)
                $file = fopen("antraege/".$_REQUEST["antrag"], "a");
                fwrite($file,$_REQUEST["forward"]."\r\n");
            fclose($file);
    }
}
...
?>
```

4 Verwendung von festen Passwörtern

4.1 Zusammenfassung

`Schwere: CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N  Wertung: 7.4`

Die Passwörter für das Backend sind fest in dessen Code integriert.

4.2 Beschreibung

Da die Passwörter und Benutzernamen als Teil des Codes fest sind, ist eine Änderung dessen für Leihen erschwert. Dies kann zur Folge haben, dass Passwortänderungen nicht durchgeführt werden oder z.B. alte Nutzer und Tests Accounts erhalten belieben und somit auch noch im Produktiven betrieb verwendbar sind.
Diese Punkte führen dazu, dass ein Bedrohungsakteur bessere Chancen hat, Passwort und Benutzernamen zu erraten oder die eines eines (ehemalig) autorisierten Nutzers zu stehlen.

4.3 Wertung

Diese Schwachstelle kann über das Internet ausgenutzt werden, hat nur eine geringfügige Komplexität, benötigt keinerlei spezieller Berechtigungen und bringt einem die vollen Fähigkeiten eines Authentifizierten Nutzers. Da aber das Ausnutzen dieser Lücke ein bekannten Passwort und Benutzernamen benötigt, welche lediglich veraltet oder die eines Test-Accounts sind, ist die Ausnutzung weitaus schwerer als bei den Vorherigen Schwachstellen. Da aber je nach Konfiguration z.B. Test-Accounts nur sehr schwache Passwörter besitzen ist dies dennoch eine kritische Schwachstelle, dies spiegelt sich CVSS 3.0 Wertung von 7.4 wieder.
Diese Lücke fällt in den OWASP Top Ten in die Kategorie ,,Cryptographic Failures", spezieller unter den Punkt ,,CWE-259: Use of Hard-coded Password".

4.4 Lösung

Die meisten Webserver können gewisse Pfade mit `HTTP Basic Authentication` schützen, diese können dann im Hintergrund eine Vielzahl von verschiedene Wege verwenden, um die Identität des Nutzers sicherzustellen. Besonders relevant ist dies, da somit auf bereits bestehende Schnittstellen zurückgegriffen werden können, in denen die Nutzer verwaltet werden.
Mit Apache2 beispielsweise kann ein LDAP Server verwendet werden[2]:

[2] `https://httpd.apache.org/docs/2.4/howto/auth.html`

```
<Directory "/var/www/html/backend">
    AuthName "backend"
    AuthType Basic
    AuthBasicProvider file
    AuthUserFile "/usr/local/apache/passwd/passwords"
    AuthLDAPURL ldap://ldaphost/o=yourorg
    AuthGroupFile "/usr/local/apache/passwd/groups"
    Require group GroupName
    Require ldap-group cn=mygroup,o=yourorg
</Directory>
```

In PHP kann dann der Nutzername über `$_SERVER['PHP_AUTH_USER']` verwendet werden.

5 Fehlendes Logging

5.1 Zusammenfassung

`Schwere: N/A`

Wenn ein Nutzer eine Anfrage an den Server stellt und dabei die Daten über POST versendet, werden diese nicht geloggt.

5.2 Beschreibung

Da in dem Code durchweg `$_REQUEST` verwendet wird, werden Anfragedaten auch über POST akzeptiert, d.h. jede Aktion kann getätigt werden, ohne dass Details zu dieser von dem Webserver erfasst werden. Dies macht es im Falle eines Angriffes unmöglich zu verfolgen was dieser getätigt hat und wie dieser vorging.
Auch für die Nachvollziehbarkeit der gewünschten Interaktionen sind gute Logs ein wichtiges Werkzeug, um z.B. Klarheit in Streitige Situationen zu bringen.

5.3 Wertung

Da jede Software Schwachstellen haben kann ist es wichtig einen guten Weg zu haben die Ausnutzung dieser zu verfolgen. Sonst ist man im zweifel nach einem Angriff nicht in der Lage die Daten und den Service wieder zu verwenden und ist gezwungen hohe Kosten zu tragen um zum einen die nun potentiell unsicheren Daten wieder zu verwenden zu können und um den gesamten Service neu entwickeln zu lassen.
Diese Lücke fällt in den OWASP Top Ten in die Kategorie „Security Logging and Monitoring Failures".

5.4 Lösung

An allen Stellen, an denen sich der Zustand des Systems ändert sollte dies in einen Log eingetragen werden. Dazu kann in PHP beispielsweise `error_log` verwendet werden. Beispielsweise sollte vermerkt werden, wenn ein Nutzer einen Antrag abgibt:

```
<?php
...
fwrite($file,$_REQUEST["fname"]."\r\n");
fwrite($file,$_REQUEST["name"]."\r\n");
fwrite($file,$_REQUEST["str"]."\r\n");
fwrite($file,$_REQUEST["plz"]."\r\n");
fwrite($file,$_REQUEST["ort"]."\r\n");
fwrite($file,urlencode($_REQUEST["begr"])."\r\n");
fwrite($file,"New\r\n");

error_log("Nutzer hat einen Antrag abgegeben mit den Daten: "
```

```
    . var_export($_REQUEST, true) . ', ' . var_export($_SESSION, true)
    . ', ' . var_export($_SERVER, true));
?>
```