ChurchTools: XSS und HTML-Injection in Eingabefeldern 2

In diesem Dokument wird die in ChurchTools immer noch vorhandene Cross-Site-Scripting-Sicherheitslücke beleuchtet. Seit dem Melden der Sicherheitslücke an den Hersteller Anfang Juni 2019 wurde bisher nur die aufgezeigte Möglichkeit eines Angriffs von außen unterbunden. Für einen Innentäter sind die aufgezeigten XSS-Angriffe weiterhin möglich.

Darüber hinaus werden Details zu bekannten und neuen verwundbaren Eingabefelder aufgelistet sowie weitere Angriffsmöglichkeiten für Innentäter aufgezeigt.

Betroffene Versionen

- ChurchTools 3.x
  - ChurchTools 3.54.0 (getestet)

Ältere Versionen werden nicht betrachtet, da der Hersteller diese nicht mehr wartet. Sie bleiben potenziell verwundbar.

Art

Persistent

Mögliche Auswirkungen

1. Identitätsdiebstahl (Identity theft)
2. Rechteausweitung (Privilege escalation)
3. Datendiebstahl (Data theft)
4. Verunstaltung der Anwendung (Defacement)

Es sind weitere Auswirkungen denkbar, da der Angreifer komplette Kontrolle per Javascript über die im Browser des Anwenders laufende ChurchTools-Anwendung hat. Proof-of-Concept-Code findet sich im ersten Bericht.

Die "Behebung" der Sicherheitslücke

Mit der Version 3.46.3 vom 4. Juni 2019 wurde die gemeldete kritische Sicherheitslücke in dem öffentlichen Formular zur Gruppenanmeldung geschlossen. Es werden nun offensichlich alle HTML-Tags aus den Textfeldern entfernt.

Die im vorherigen Bericht genannten Eingabefelder wurden mit dem Proof-of-Concept-Code überprüft. Die genannten Codefragmente werden nicht mehr ausgeführt. Weitere Tests zeigten auf, dass die durch den User eingegebenen Zeichenketten für die Ausgabe folgendermaßen umgewandelt ("escaped") werden:

- `<script>` wird zu `&lt;script&gt;`
- `</script>` wird zu `&lt;/script&gt;`
- `"` wird zu `&quot;`
- `` ` `` wird zu `&#x60;`
- `'` wird zu `&#39;`
- `=` wird zu `&#x3D;`

Weitere Umwandelungen finden nicht statt. Doch damit ist HTML-Injection und XSS weiterhin möglich.

Da die Anführungsstriche und Gleichheitszeichen umgewandelt werden, wird HTML-Injection etwas eingeschränkt, sie besteht jedoch weiterhin. XSS ist möglich, da ausschließlich die Kleinschreibung des Skript-

Tags umgewandelt wird. XSS funktioniert weiterhin mit Anpassung der Groß- Kleinschreibung. Verwendet man `<Script>` und `</Script>`, so können weiterhin Skripte durch den Anwender injected werden.

Die Sicherheitslücke wurde nur unzureichend behoben. Es sind weiterhin alle im ersten Bericht erwähnten Felder betroffen. Jedoch sind die Möglichkeiten durch das vorgenommene Escaping etwas eingeschränkter. Das Defacement der Anwendung ist weiterhin möglich.

Betroffene Felder

Dies ist *keine* komplette Auflistung aller betroffenen Felder in der Version 3.54.0. Für eine komplette Liste müssten *alle* Eingabefelder der Anwendung überprüft werden, ob sie überall sicher ausgegeben werden.

HTML-Injection bei Eingabefeldern impliziert meistens auch XSS, sofern das Eingabefeld nicht sicher ausgegeben wird. Manchmal macht nur die maximal mögliche Zeichenkettenlänge in einem Eingabefeld XSS unmöglich, weil einfach kein Platz für das Script-Tag und Javascriptcode vorhanden ist.

Darüber hinaus kann in der folgenden Liste ein Eingabefeld von XSS betroffen sein, obwohl es nicht so markiert ist. Es könnte nämlich sein, dass ein Eingabefeld irgendwo in der Anwendung unsicher ausgegeben wird, was jedoch während der Tests nicht entdeckt wurde.

Beispiele für Eingabefelder, die durch HTML-Injection und/oder XSS betroffen sind:

- Alle Module
    - Stammdaten bearbeiten
        - "Bezeichnung" HTML, DF (Stammdaten bearbeiten funktioniert nicht mehr)
        - Alle Textfelder HTML, DF (Stammdaten bearbeiten funktioniert nicht mehr)
- Personen & Gruppen
    - Person - Adresse
        - "Titel" HTML
        - "Vorname" HTML, SCRIPT (Startseite: Wer ist online, Aufgaben, Wichtige Einträge vom Wiki)
        - "Spitzname" HTML
        - "Nachname" HTML, SCRIPT (Startseite: Wer ist online, Aufgaben, Wichtige Einträge vom Wiki)
        - "Strasse" HTML, DF (Popup beim Bewegen der Maus über Personenname)
        - "Zusatz" HTML, DF
        - "PLZ" HTML
        - "Ort" HTML, DF
        - "Land" HTML
        - "Telefon privat" HTML
        - "Telefon geschäftl." HTML
        - "Handy" HTML
        - "Fax" HTML
        - "E-Mail" HTML, DF
        - "Benutzername" HTML, DF
        - "Optigem-Nr" HTML
        - In den Stammdaten definierte DB-Felder vom Typ Textfeld und Kommentarfeld. HTML, DF
    - Person - Information
        - "Geburtsname" HTML
        - "Geburtsort" HTML
        - "Beruf" HTML, DF
        - "Überwiesen von" HTML

        - "Überwiesen nach" HTML
        - "Taufort" HTML, DF
        - "Getauft durch" HTML, DF
        - In den Stammdaten definierte DB-Felder vom Typ Textfeld und Kommentarfeld. HTML, DF
    - Person - Kommentare
        - "Kommentar" HTML, DF
    - Gruppe bearbeiten
        - "Titel" HTML, DF (Gruppenliste funktioniert nicht mehr), SCRIPT (Startseite: Aufgaben, Checkins der letzten Tage)
        - "Maximale Teilnehmeranzahl" HTML
        - "Startzeit" HTML
        - "Bemerkung" HTML, SCRIPT (Startseite: Checkins der letzten Tage)
    - Gruppe
        - "Kommentar schreiben" HTML, DF (Alle vorherigen Kommentare verschwinden aus der Anzeige)
    - Gruppe - Gruppenteilnahme bearbeiten
        - "Bemerkung" HTML, DF (Bearbeiten/Löschen der Gruppenteilnahme nicht mehr möglich)
    - Gruppe - Weitere Felder hinzufügen
        - "Titel" HTML, DF (Ändern des Feldes nicht mehr möglich)
        - "Standardwert" HTML, DF (Ändern des Feldes nicht mehr möglich)
- Events
    - Dienstplan - Dienst zum Event hinzufügen oder entfernen - Dienst erstellen/bearbeiten
        - "Bezeichnung" HTML, DF
        - "Bemerkung" HTML, DF
    - Dienstplan - Eventitem - Weitere Infos bearbeiten
        - "Weitere Infos zum Event" HTML, DF
    - Abwesenheiten - Abwesenheit bearbeiten
        - "Kommentar" HTML, DF
    - Ablaufplan - Ablaufplan editieren
        - "Bezeichnung" HTML, SCRIPT (direkt nach dem Speichern)
        - "Predigtserie" HTML
    - Ablaufplan - Position bearbeiten
        - "Titel" HTML, DF
        - "Zuständig" HTML, DF
        - "Bemerkung" HTML, DF
    - Ablaufplan - Überschrift hinzufügen/bearbeiten

        - "Titel" HTML, DF
    - Ablaufplan - Im Ablaufplan den Text für Servicegruppen bearbeiten
        - Text HTML, DF
    - Songs - Song erstellen/bearbeiten
        - "Bezeichnung" HTML, DF
        - "Autor" HTML, DF
        - "Copyright" HTML, DF
        - "CCLI-Nummer" HTML, DF
    - Songs - Song anklicken - Arrangement-Tab - Bearbeiten
        - "Bezeichnung" HTML, DF
        - "BMP" HTML
        - "Takt" HTML
        - "Bemerkungen" HTML, DF
    - Songs - Song anklicken - Arrangement-Tab
        - "Kommentar schreiben" HTML, DF
    - Songs - Song anklicken - Arrangement-Tab - "Link hinzufügen"
        - "Bezeichnung" HTML, DF
- Ressourcen
    - Buchungsplan - Buchungsanfrage erstellen/bearbeiten/kopieren
        - "Titel" HTML
        - "Bemerkung" HTML
        - "Weitere Infos" HTML, DF
- Kalender
    - Kalender - Kalender verwalten - Neuen Kalender erstellen / Einstellungen für Kalender
        - Titel HTML, SCRIPT (Kalender - Liste der Kalender)
    - Kalender - Termin erstellen
        - Titel HTML
        - Bemerkung HTML, SCRIPT (Anzeige des Kalenders mit dem Termin)
        - Weitere Infos HTML, SCRIPT (Popup bei Hover über Termin in diesem Kalender)
- Wiki
    - Link hinzufügen
        - "Bezeichnung" 255 Zeichen. HTML

Legende - Testfälle (Proof of Concept)

- HTML = HTML-Injection möglich.

```
<b>bold</b>
```

wird als fettgedruckter Text gerendert.

- SCRIPT = XSS möglich. Direkte Skriptausführung, da kein Escaping der Benutzereingabe erfolgt.

  ```
  <script>alert('!')</script>
  ```

  wird ausgeführt: Modales Dialogfenster wird angezeigt.

- DF = Defacement möglich. Anwendung ist (teilweise) nicht mehr verwendbar.

  ```
  <Script>alert('!')</Script>
  ```

  Ausführung einer ungültigen Anweisung durch Escaping der Anführungsstriche. Abbruch der Scriptausführung => Funktionalität der Anwendung wird beeinträchtigt.

  ```
  <Script>document.write()</Script>
  ```

  Überschreiben des aktuellen Documents. Anwendung wird unbedienbar.

Weitere Angriffsmöglichkeiten von innen

Neben den bereits aufgezeigten XSS-Möglichkeiten in einfachen Texteingabefeldern gibt es für einen Angreifer von innen - also jemand, der bereits einen ChurchTools-Zugang besitzt - weitere Möglichkeiten für XSS. Potenzielle Kandidaten sind Eingabefelder, bei denen ein Rich-Text-Editor (RTE) zum Einsatz kommt. Diese eignen sich besonders, da Text unbeschränkter Länge eingegeben werden kann.

A. XSS im Wiki

Sobald der Angreifer die Berechtigung "Einzelne Wiki-Kategorien editieren (edit category)" besitzt, kann er beliebig lange Skripte anderen Anwendern unterschieben.

Dazu im Wiki die Seite öffnen, in die das Skript eingefügt werden soll. Anschließend folgende Anweisung in der Javascript-Konsole des Browsers ausführen:

```
churchInterface.jsendWrite({func:"save",
                            doc_id:currentPage.doc_id,
                            wikicategory_id:currentPage.wikicategory_id,
                            val:currentPage.text +"<script>alert('DANGER!')</script>",
                            auf_startseite_yn:currentPage.auf_startseite_yn,
                            identifier: currentPage.identifier}, null, null, !1)
```

Sobald nun ein anderer Benutzer diese so bearbeitete Wikiseite besucht, wird das eingeschleuste Skript ausgeführt. Anstelle des "alert('DANGER!')" im Beispiel kann beliebig langer Javascript-Code eingebettet werden.

"Optimieren" des XSS-Angriffs im Wiki

Ein Angreifer würde die Lebensdauer und die Chance auf Ausführung seines Skripts durch Anpassung folgender Parameter bewusst erhöhen:

1. Seite für Skripteinbettung
   - Innerhalb der Standardkategorie des Wikis, weil dort die Menge der potenziell berechtigten Benutzer am höchsten ist.
   - Ideal sind die immer vorhandenen Wiki-Seiten "main" oder "Sicherheitsbestimmungen". Die Anzahl der Besucher ist dort am höchsten.

B. Fileupload - Skript aus hochgeladener Datei

Mit Hilfe des Dateiupload-Features können beliebig komplexe Skripte dem Anwender untergeschoben werden.

Hierfür benötigt der Angreifer eine Upload-Berechtigung, d.h. mindestens das Recht, eines der Uploadmöglichkeiten zu nutzen, z. B.

- Mein Profil
  - Änderung des Personenbildes
- Personen & Gruppen
  - Person
    - Änderung des Personenbildes (nur JPG- oder PNG-Bild)
    - Anhänge
  - Gruppe
    - "Gruppenbild hinzufügen"
- Events
  - Dienstplan
    - Datei zum Event hochladen
  - Songs - Song anklicken - Arrangement-Tab
    - Datei hochladen

Tragischerweise hat ein Anwender meistens das Recht seine eigenen Daten zu ändern, also auch ein eigenes Bild hochzuladen.

Anschließend bindet der Angreifer das hochgeladene Skript unter Verwendung eines für XSS anfälligen Eingabefeldes ein.

Beispiel: Einbinden in eine Wiki-Seite

1. Skript zu ChurchTools hochladen (z.B. statt dem eigenen Bild die Skriptdatei hochladen)
2. URL der hochgeladen Skript-Datei zur Verwendung im folgenden Schritt kopieren. URL sieht ungefähr so aus: https://victim.church.tools/?q=churchwiki/filedownload&id=12345&filename=d7f878f0003fa7cd8eb34736e49dd4b1abee868bf94b34ef1b1eaf647544547b (https://victim.church.tools/?q=churchwiki/filedownload&id=12345&filename=d7f878f0003fa7cd8eb34736e49dd4b1abee868bf94b34ef1b1eaf647544547b)
3. Folgendes in der Javascript-Konsole des Browsers ausführen, die zuvor kopierte URL an die passende Stelle einfügen:

```
churchInterface.jsendWrite({func:"save",
                            doc_id:currentPage.doc_id,
                            wikicategory_id:currentPage.wikicategory_id,
                            val:currentPage.text+"<script src='URL'></script>",
                            auf_startseite_yn:currentPage.auf_startseite_yn,
                            identifier: currentPage.identifier}, null, null, !1)
```

Sobald nun ein anderer Benutzer diese so bearbeitete Wikiseite besucht, wird das Skript direkt von ChurchTools abgerufen und im Kontext des anderen Benutzers ausgeführt.

"Verstecken" des Uploads

Ein Angreifer würde die Lebensdauer seines Skripts durch Anpassung folgender Parameter bewusst erhöhen:

1. Dateiname
    - Skriptdatei mit unverfänglichem Dateinamen und Dateiendung, wie z.B. diagramm.jpg.

2. Uploadort der Skriptdatei
    - Eigenes Bild
    - Wiki-Seite, auf der bereits viele Bild-Dateianhänge vorhanden sind.
    - Selten besuchte Wiki-Seite
    - Neue Wiki-Seite mit einem kryptischen Namen.

Maßnahmen zur Behebung

HTML-Injection und XSS

Gemäß der allgemeinen Regel "Never trust user inputs!" kann nur das wiederholt werden, was bereits im ersten Bericht stand.

Zur Behebung der Schwachstelle sollte in folgenden Bereichen nachgebessert werden:

1. Escaping **aller** Variableninhalte bei HTML- und JavaScript-Ausgabe
2. Sanitizing **aller** HTML-Benutzereingaben auf dem Server
3. Validierung aller Benutzereingaben auf dem Server

Der Einsatz von bewährten Frameworks für das Escaping und der Validierung ist einer Eigenimplementierung vorzuziehen. Für HTML-Sanitizing kann z. B. HTML Purifier (http://htmlpurifier.org/) eingesetzt werden. Vue.js (https://vuejs.org/) sollte auch bei älteren Teilen der Anwendung zum Einsatz kommen.

File-Upload

Der Missbrauch eines Dateiuploads zur Ablage von Javascript-Code könnte durch die Validierung der hochgeladenen Daten minimiert werden.

1. Definition von zulässigen Dateitypen/-inhalten bei den unterschiedlichen Dateiuploadmöglichkeiten in der Anwendung.
2. Überprüfung und Validierung des Dateiinhaltes auf dem Server gegen die zulässigen Dateitypen/-inhalte.

Konkret bedeutet das z. B. für den Bilderupload im Profil eines Benutzers:

- Zulässige Dateitypen: png-/jpg-/gif-Bild
- Überprüfung auf dem Server, ob es sich bei den geposteten Bilddaten wirklich um ein Bild der Formate png, jpg, gif handelt. Nur wenn es sich um ein Bild der zulässigen Formate handelt, wird es gespeichert. Andernfals erhält der Benutzer eine Fehlermeldung.

Historie

- **Ende Mai 2019** Zufällige Entdeckung einer kritischen Sicherheitslücke bei der Dateneingabe in ChurchTools. Weitere Tests bei weiteren Eingabefeldern führen zur Entdeckung der HTML-Injection-Sicherheitslücke. Es sind alle Versionen betroffen, einschließlich der Version 3.46.2.
- **1. Juni 2019** Erstellung eines Berichts über die Sicherheitslücken über XSS und HTML-Injection in ChurchTools. Der Bericht enthält Vorschläge zu Behebung der Sicherheitslücken.
- **2. Juni 2019** Meldung der kritischen Sicherheitslücke durch Übersendung des Berichts an den Support (support@churchtools.de ()).
- **3. Juni 2019** Die kritische Sicherheitslücke wird bestätigt. Hersteller verspricht, die kritische Sicherheitslücke "im Laufe der Woche" in Version 3.46.3 zu fixen. Die Behebung der weiteren Sicherheitslücken wird zugesagt: "Die anderen von Ihnen beschrieben Sicherheitslücken (Angriffe von innen) werden wir prüfen

und so schnell wie möglich innerhalb der nächsten Sprints fixen".

- **4. Juni 2019** Version 3.46.3 erscheint. Die kritische Sicherheitslücke ist behoben.
- **12. Dezember 2019** Erneute Untersuchung der gemeldeten Sicherheitslücke durch Angreifer von innen mit der aktuellen Version 3.53.0. Trotz der Zusage per E-Mail, die HTML-Injection-Sicherheitslücke bei diversen Eingabefeldern zu prüfen und zu fixen, ist wenig passiert. Der Fix durch Hersteller ist unzureichend, die Vorschläge zur Behebung wurden nicht umgesetzt. Defacement durch HTML-Injection ist weiterhin in im Bericht von Juni 2019 erwähnten Eingabefeldern möglich. XSS ist in einigen Felder weiterhin möglich.
- **13. Dezember 2019** Weitere Untersuchung des Fixes (clientseitigen Escaping). Dabei werden weiterere XSS-Eingabefelder entdeckt. Bei der Ausweitung der Test von Eingabefeldern wird die XSS-Sicherheitslücke im Wiki entdeckt.
- **17. Dezember 2019** Erstellung eines zweiten Berichts.
- **20. Dezember 2019** Update des zweiten Berichts mit Version 3.54.0. Keine Änderungen zur Vorversion.

Weiterführende Links

- Dipl.-Inform. Carsten Eilers: Artikelserie über Cross-Site Scripting (XSS) (https://www.ceilers-news.de/serendipity/842-Die-IoT-Top-10,-1-Unsichere-Weboberflaechen,-Teil-5.html)
- Pathirennehelage Nadeeshani: Protect your website form cross site scripting (XSS) attacks (https://medium.com/@nshani/protect-your-website-form-cross-site-scripting-xss-605fe0f50d5f)
- Open Web Application Security Project: Cross Site Scripting Prevention Cheat Sheet (https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)
- HTML5 Security Cheatsheet - A collection of HTML5 related XSS attack vectors (https://html5sec.org/)
- Ashar Javed: On Breaking PHP-Based Cross-Site Scripting Protections In The Wild (http://slides.com/mscasharjaved/on-breaking-php-based-cross-site-scripting-protections-in-the-wild#/1)

Autor

- Michael Leimer <m.leimer@gmx.de ()> im Dezember 2019