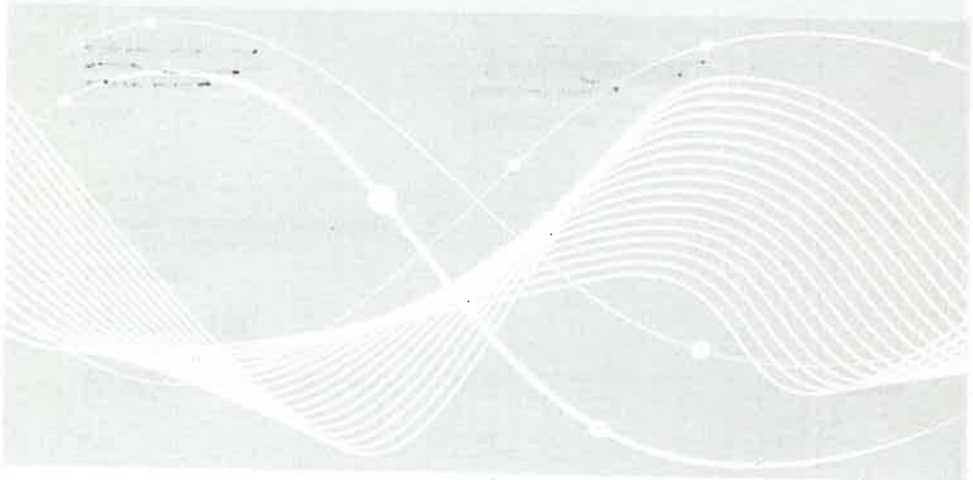
Bundesamt
für Sicherheit in der
Informationstechnik

Testbericht

Sicherheitstest des Content-Management-Systems Plone



Änderungshistorie

Version	Datum	Name	Beschreibung
0.1	12.02.2016	[redacted]	Initiale Erstellung
0.2	18.02.2016	[redacted]	Fehlerbeschreibungen
0.3	01.03.2016	[redacted]	Review
1.0	02.03.2016	[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

Vorlage:

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Internet: https://www.bsi.bund.de
© Bundesamt für Sicherheit in der Informationstechnik 2016

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

Dieses Dokument steht unter der Creative-Commons-Lizenz Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie http://creativecommons.org/licenses/by-nc-nd/4.0/.

Danksagung

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

Inhaltsverzeichnis

Änderungshistorie ... 2
Impressum ... 3
1 Zusammenfassung ... 9
1.1 Allgemeines ... 9
1.2 Zusammenfassung der Testergebnisse ... 9
1.2.1 Übersicht der Testergebnisse ... 9
1.2.2 Bewertung des Sicherheitsniveaus ... 11
1.3 Prüfpunkte ... 11
1.4 Vorgehen im Test ... 11
2 Phase 1: Vorbereitung ... 13
2.1 Informationsbasis ... 13
2.2 Aggressivität ... 13
2.3 Umfang und Testtiefe ... 13
2.4 Vorgehensweise und Sichtbarkeit ... 13
2.5 Technik ... 14
2.6 Ausgangspunkt ... 14
2.7 Testsystem ... 14
2.8 Testdaten ... 14
2.9 Testendekriterien ... 14
2.10 Betrachtung rechtlicher Fragen ... 14
2.11 Planung der Testdurchführung ... 15
2.12 Werkzeugeinsatz ... 15
3 Phase 2: Informationsbeschaffung ... 16
3.1 Auswahl der Testmodule ... 16
3.2 Prüfung der Testmodule ... 16
4 Phase 3: Bewertung der Informationen ... 17
4.1 Kritische Bereiche und erreichbare Ziele ... 17
4.2 Zugriffspfade ... 17
4.3 Beschreibung der Prüfpunkte ... 17
5 Phase 4: Aktive Eindringversuche ... 18
6 Phase 5: Abschlussanalyse ... 19
6.1 Content Management System ... 19
6.1.1 Zusammenfassung ... 19
6.1.2 Cross-Site-Scripting ... 19
6.1.3 Denial of Service ... 31
6.1.4 Brute Force von Benutzerpasswörtern ... 35
6.1.5 Missbrauch der Anwendung als SPAM-Verteiler ... 36
6.1.6 Cross-Site Request Forgery ... 36
6.1.7 Fehlendes Secure Cookie-Flag ... 37
6.1.8 Enumeration von Benutzernamen durch Passwort-Vergessen-Funktion ... 39
6.1.9 Formulare mit Autocomplete ... 39
6.1.10 Schwache Passwort-Richtlinien ... 40

6.1.11 Fehler in Authentifizierung und Session-Management ... 40
6.1.12 Verschleierung des Erstellers einer Seite ... 44
6.1.13 Berechtigungsüberschreitung ... 47
6.2 Betriebssystem ... 50
6.2.1 Zusammenfassung ... 50
6.3 Webserver ... 51
6.3.1 Zusammenfassung ... 51
Anhang: Bewertungskriterien der Schwere der Testergebnisse ... 52
Literaturverzeichnis ... 54
Stichwort- und Abkürzungsverzeichnis ... 55

Abbildungsverzeichnis

Abbildung 1: Ausführung von JavaScript-Code durch das Feld „Kommentar“ innerhalb "Kommentare moderieren" ... 21
Abbildung 2: Besuch der Seite xss-inject.html mit anschließender Ausführung von darin enthaltenem JavaScript-Code ... 22
Abbildung 3: Ausführung von JavaScript-Code durch das Feld "location" innerhalb eines Termins ... 24
Abbildung 4: Einfügen von JavaScript innerhalb der „Benutzereinstellungen“ im Feld „Vor- und Nachname“. ... 25
Abbildung 5: XSS im Feld "Ersteller" auf der Seite "Seite bearbeiten" ... 26
Abbildung 6: Ausführung von JavaScript im Feld "Ersteller" auf der Seite "Seite bearbeiten" ... 26
Abbildung 7: Hinzufügen eines Inhalts (hier: "Seite") im Bereich "Intern" ... 27
Abbildung 8: Einfügen von JavaScript als Seiten-Titel ... 27
Abbildung 9: Ausführung von JavaScript durch die Auswahl der vormals erstellten Seite im Bereich "Inhalte" ... 28
Abbildung 10: Ausführung von JavaScript durch Seiten-Titel über den Link-Einfügen-Dialog ... 28
Abbildung 11: Ausführung von JavaScript durch Seiten-Titel über den Link-Einfügen-Dialog ... 29
Abbildung 12: Ausführung von JavaScript-Code durch Feld-Erläuterung innerhalb Dateifeld ... 30
Abbildung 13: Übergabe des CSRF-Tokens „_authenticator“ innerhalb der URL ... 37
Abbildung 14: Fehlendes Secure Cookie-Flag ... 38
Abbildung 15: Übertragung des Cookies im Klartext über HTTP ... 38
Abbildung 16: Fehlender HTTP-Header "Cache-control: no store“ bzw. „Pragma: no-cache" innerhalb zugriffsbeschränkter Bereiche ... 43
Abbildung 17: Das Feld "Ersteller" für "Seite A" ... 46
Abbildung 18: Bearbeiten von "Seite A": Nutzer "Pentest Admin" an erste Stelle verschieben ... 46
Abbildung 19: Das Feld "Ersteller" für "Seite A" nach der Bearbeitung ... 47
Abbildung 20: Deaktivieren von "Berechtigungen von übergeordneten Ordnern übernehmen" ... 48
Abbildung 21: "Seite A" auf Privat schalten ... 48
Abbildung 22: Kopiervorgang von "Ordner A" durch Nutzer "B" macht "Seite A" sichtbar ... 48

Tabellenverzeichnis

Tabelle 1: Dokumente ... 13
Tabelle 2: Testplan ... 15
Tabelle 3: Hinzufügen eines Kommentars mit eingebettetem JavaScript-Code ... 20
Tabelle 4: Hochladen von HTML-Datei innerhalb des internen Bereichs ... 22
Tabelle 5: Einfügen von JavaScript-Code in das Feld „location“ innerhalb eines Termins ... 23
Tabelle 6: Hinzufügen eines Dateifelds in einen Formular-Ordner. Feld-Erläuterung enthält JavaScript-Code ... 30

Tabelle 7: Anlegen einer Nachricht mit eingebettetem JavaScript-Code innerhalb des Feldes "Lead Image Caption" ... 31
Tabelle 8: Hochladen eines beliebig großen Kommentars ... 33
Tabelle 9: Hochladen eines beliebig großen Kommentars durch das Kontaktformular ... 34

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 Plone in der Version 5.0 (5011).

Softwaresystem (CMS):	Plone
Versionsnummer:	5.0 (5011)
Plugins:	EasyNewsletter 2.6.15 PloneFormGen 1.8.0
Hersteller:	Plone Foundation
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:	02.01.16 bis 12.02.16

1.2 Zusammenfassung der Testergebnisse

1.2.1 Übersicht der Testergebnisse

Kapitel	Beschreibung	Modul	Risikofaktor
6.1.2.1	Cross-Site-Scripting in „Kommentare moderieren" im Feld „Kommentar"	Plone Core	Schwer
6.1.2.2	Cross-Site-Scripting durch hochgeladene HTML-Dateien	Plone Core	Mittel
6.1.2.3	Cross-Site-Scripting in Termin im Feld „location"	Plone Core	Mittel
6.1.2.4	Cross-Site-Scripting in „Seite bearbeiten" in Feld „Ersteller"	Plone Core	Mittel
6.1.2.5	Cross-Site-Scripting in Seiten-Auswahl im Menüpunkt „Inhalte" in Seiten-Titel	Plone Core	Mittel
6.1.2.6	Cross-Site-Scripting in „Link-Einfügen"-Dialog im Seiten-Titel	Plone Core	Mittel
6.1.2.7	Cross-Site-Scripting in „Formularfeld anlegen"	PloneFormGen	Mittel
6.1.2.8	Cross-Site-Scripting in „Nachricht anlegen" im Feld „Lead Image Caption"	Plone Core	Mittel
6.1.3.2	Absenden beliebig großer Kommentare	Plone Core	Mittel
6.1.3.3	Absenden beliebig vieler Kommentare	Plone Core: Härtung	Mittel
6.1.3.4	Absenden beliebig großer Kommentare im Kontakt-Formular	Plone Core	Mittel

Kapitel	Beschreibung	Modul	Risikofaktor
6.1.3.5	Absenden beliebig vieler Kommentare im Kontakt-Formular	Plone Core	Mittel
6.1.4	Brute Force von Benutzerpasswörtern	Plone Core	Mittel
6.1.11.1	Passwort-Änderung führt nicht zur Invalidierung aller Sessions und der Aushandlung eines neuen Session-Cookies	Plone Core	Mittel
6.1.13.1	Unberechtigter Zugriff auf Daten oder Verzeichnisse durch Kopieren-Funktion	Plone Core	Mittel
6.1.13.2	Unberechtigter Zugriff auf Daten oder Verzeichnisse durch fehlerhaftes Berechtigungssystem	Plone Core	Mittel
6.1.3.1	Hochladen beliebig großer Profilbilder	Plone Core	Leicht
6.1.5	Missbrauch der Anwendung als SPAM-Verteiler	Plone Core	Leicht
6.1.6.1	CSRF-Token wird als GET-Parameter übergeben	Plone Core	Leicht
6.1.6.2	CSRF-Token besitzt unzureichende Zufälligkeit	Plone Core	Leicht
6.1.7	Fehlendes Secure Cookie-Flag	Plone Core	Leicht
6.1.8	Enumeration von Benutzernamen durch Passwort-Vergessen-Funktion	Plone Core	Leicht
6.1.9	Formulare mit Autocomplete	Plone Core	Leicht
6.1.10	Schwache Passwort-Richtlinien	Plone Core	Leicht
6.1.11.2	Mehrfachanmeldung möglich	Plone Core	Leicht
6.1.11.3	Abmeldung führt nicht zur Invalidierung der Session	Plone Core	Leicht
6.1.11.4	Cacheable HTTP(S) responses	Plone Core	Leicht
6.1.11.5	Session-Timeouts zu groß gewählt	Plone Core: Härtung	Leicht
6.1.12	Verschleierung des Erstellers einer Seite	Plone Core	Leicht
6.1.13.3	Kopieren und Hinzufügen von sichtbaren Inhalten trotz fehlender Berechtigung	Plone Core	Leicht
6.1.13.4	Unberechtigter Zugriff auf Daten oder Verzeichnisse durch nicht-intuitives Berechtigungssystem	Plone Core	Leicht
6.1.11.6	Mehrmalige Vergabe des selben Session-Cookies	Plone Core	Hinweis
6.1.13.5	Unberechtigter Zugriff auf Daten oder Verzeichnisse innerhalb privat geschalteter Unterordner	Plone Core	Hinweis
6.1.13.6	Fehlende Freigabe- und Status-Einstellmöglichkeiten im Hinzufügen-Dialog	Plone Core	Hinweis

1.2.2 Bewertung des Sicherheitsniveaus

Ausgehend von den identifizierten Sicherheitsrisiken kann dem CMS Plone inklusive der getesteten Plugins ein mittleres Sicherheitsniveau attestiert werden.

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:

- Cross-Site-Scripting innerhalb des Plone Cores
- Nicht-intuitives Berechtigungssystem

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	Version / Stand
Plone 5 Documentation	http://docs.plone.org/	-

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	02.01.16 bis 12.02.16
Fertigstellung des Testberichts	02.03.2016

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
IntelliJ (PyCharm)	JetBrains	Untersuchung des Codes	Testteam
Nikto	CIRT	Schwachstellen-Scanner auf Applikationsebene	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

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

Die folgenden Beschreibungen zeigen, dass die schweren Schwachstellen innerhalb des Plone Cores (Basis-Installation ohne Plugins und Themes) zu finden sind. Die (wenigen installierten) Plugins zeigen hingegen keine kritischen Schwachstellen auf. Dieses Ergebnis resultiert in einem weiten Teil aus der Tatsache, dass bei Plone schon viele Erweiterungen im Core enthalten sind.

Erweiterungen für Plone werden üblicherweise als Python Package auf Basis eines vorgegebenen Code Template entwickelt. Durch diese vorgegebene Struktur und die Verwendung von Zope-Komponenten können einige Ursachen für Schwachstellen in den Plugins vermieden werden.

Auffällig ist, dass altbekannte Schwachstellentypen (Cross-Site-Scripting) auftreten und sehr trivial ausgenutzt werden können. Zudem ist das Berechtigungssystem wenig intuitiv. Hieraus entsteht hinsichtlich der ungeschulten Zielgruppe potentiell ein Sicherheitsproblem, falls durch mangelndes Verständnis oder fehlerhafte Berechtigungsvergabe vertrauliche Informationen veröffentlicht werden. Es lässt sich vermuten, dass diese Probleme aufgrund der relativ kurzen Verfügbarkeit von Plone 5 auftreten und bei der Implementierung der neuen CMS-Version nicht ausreichend betrachtet wurden.

6.1.2 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:

HTML-Codierung ist der grundsätzliche Schutzmechanismus gegen diese Angriffe. Viele Frameworks bzw. Sprachen stellen hierfür entsprechende Funktionen zur Verfügung.

Folgende Zeichen müssen mindestens in die entsprechende HTML-Entities umgewandelt werden (welche nicht als HTML/JavaScript interpretiert 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 \\).

6.1.2.1 Cross-Site-Scripting in „Kommentare moderieren" im Feld „Kommentar"

Betroffenes Modul: Plone Core

Ergebnis:

Ein unangemeldeter Benutzer ist über die Kommentar-Funktion im Frontend in der Lage, JavaScript-Code in das Feld „Kommentar" einzufügen (siehe Tabelle 3), welches im Backend auf der Seite *„Kommentare moderieren"* unmaskiert ausgegeben und im Kontext des Backendbenutzers ausgeführt wird (siehe Abbildung 1).

```
POST /audio-und-video/document_view HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://deb-plone-scenario4.cms2.local/audio-und-video
Cookie: I18N_LANGUAGE="de"
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------
92402929314851
Content-Length: 809

[TRUNCATED]

-----------------------------92402929314851
Content-Disposition: form-data; name="form.widgets.text"

<script>alert("XSS")</script>

[TRUNCATED]
```

Tabelle 3: Hinzufügen eines Kommentars mit eingebettetem JavaScript-Code.

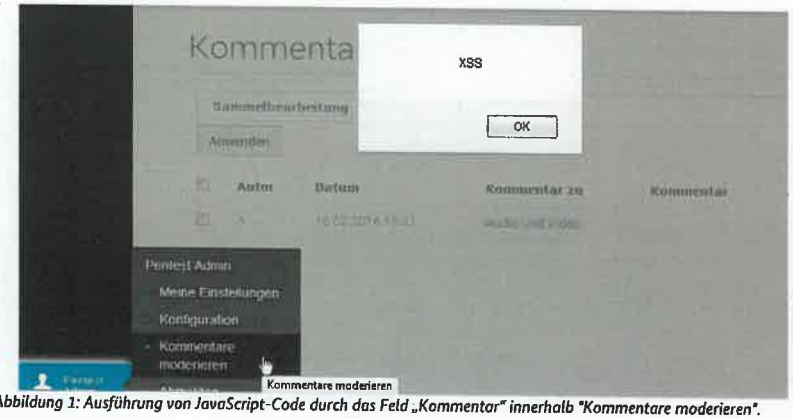


Abbildung 1: Ausführung von JavaScript-Code durch das Feld „Kommentar" innerhalb "Kommentare moderieren".

Diese Schwachstelle resultiert aus der Kommentar-Funktionalität, welche standardmäßig deaktiviert ist. Dennoch weist sie ein hohes Risiko auf, da sie zur Kompromittierung von Nutzern mit höheren Rechten (Redakteur oder Administrator) geeignet ist.

Risikofaktor: Schwer

6.1.2.2 Cross-Site-Scripting durch hochgeladene HTML-Dateien

Betroffenes Modul: Plone Core

Ergebnis:

Als angemeldeter Benutzer ist es im internen Bereich möglich, eigene HTML-Dateien mit JavaScript-Code hochzuladen (siehe Tabelle 4). Während die Seite */intern/<Dateiname>.html/view* ohne Seiteneffekte dargestellt wird, wird beim Besuch der Seite */intern/<Dateiname>.html* JavaScript-Code ausgeführt (siehe Abbildung 2).

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Maßnahme:

Die Seite */intern/<Dateiname>.html* sollte eine Weiterleitung zu */intern/<Dateiname>.html/view* realisieren.

Alternativ kann eine Weiterleitung / View auf *<Dateiname>/raw* erfolgen und für gefährliche MIME-Typen (text/html, javascript, ggf. xml) der HTTP Content Type auf text/plain gesetzt werden.


```
POST /intern/++add++File/@@z3cform_validate_field HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
X-Requested-With: XMLHttpRequest
Referer: https://deb-plone-scenario4.cms2.local/intern/++add++File
Content-Length: 2766
Content-Type: multipart/form-data; boundary=---------------------------
22504913129425
Cookie: I18N_LANGUAGE="de";
__ac="ykaXVlzsVJ2fNBLq175H7wAjQzG1rGhXnIhDfN18Lq81NmM2ZDA2YTM0ZGFhMGZkZjdhNzRiM
Tk4NjY0NTNjODM2MjczNTc0IQ%3D%3D"
Connection: close

-----------------------------22504913129425
Content-Disposition: form-data; name="form.widgets.title"

XSS-Inject
-----------------------------22504913129425
Content-Disposition: form-data; name="form.widgets.description"


-----------------------------22504913129425
Content-Disposition: form-data; name="form.widgets.file"; filename="xss-
inject.html"
Content-Type: text/html

<script>alert("XSS");</script>

[TRUNCATED]
```

Tabelle 4: Hochladen von HTML-Datei innerhalb des internen Bereichs.

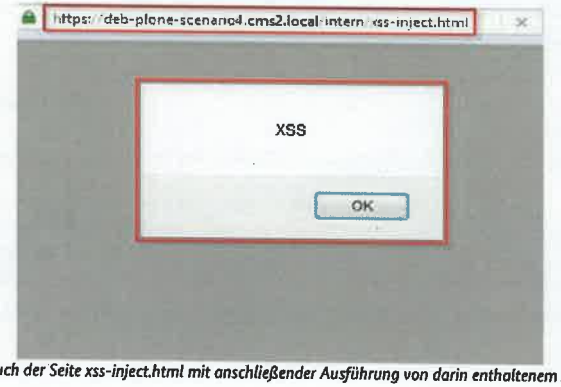


Abbildung 2: Besuch der Seite xss-inject.html mit anschließender Ausführung von darin enthaltenem JavaScript-Code.

Risikofaktor: Mittel

6.1.2.3 Cross-Site-Scripting in Termin im Feld „location"

Betroffenes Modul: Plone Core

Ergebnis:

Ein angemeldeter Nutzer kann einen Termin anlegen, in dem er bösartigen JavaScript-Code in das Feld „Ort" (Parameter „location") eintragen kann (siehe Tabelle 5). Der Parameter wird nicht ausreichend maskiert und im Frontend ausgegeben, wodurch ein XSS-Angriff möglich wird (siehe Abbildung 3).

```
POST /intern/++add++Event/@@z3cform_validate_field HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
X-Requested-With: XMLHttpRequest
Referer: https://deb-plone-scenario4.cms2.local/intern/++add++Event
Content-Length: 4874
Content-Type: multipart/form-data; boundary=---------------------------
652074611966
Cookie: I18N_LANGUAGE="de";
__ac="MLLRO93xKDlsiFtzyFPPBv1yeIkOWIlv3EdjfEcCOHA1NmM2ZGNmMTM0ZGFhMGZkZjdhNzRiM
Tk4NjY0NTNjODM2MjczNTc0IQ%3D%3D"
Connection: close

[TRUNCATED]

-----------------------------652074611966
Content-Disposition: form-data; name="form.widgets.IEventLocation.location"

<script>alert("XSS")</script>

[TRUNCATED]
```

Tabelle 5: Einfügen von JavaScript-Code in das Feld „location" innerhalb eines Termins.

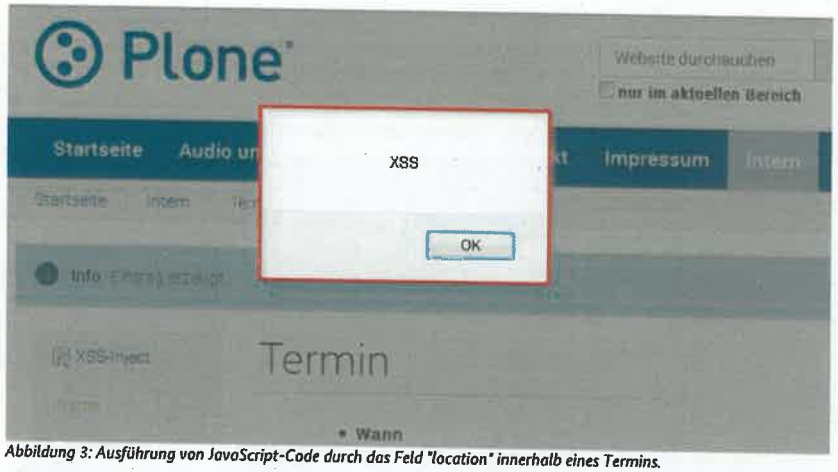


Abbildung 3: Ausführung von JavaScript-Code durch das Feld "location" innerhalb eines Termins.

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Risikofaktor: Mittel

6.1.2.4 Cross-Site-Scripting in „Seite bearbeiten" in Feld „Ersteller"

Betroffenes Modul: Plone Core

Ergebnis:

Ein angemeldeter Nutzer kann innerhalb seines Benutzerprofils im Tab „Persönliche Informationen" in das Feld „Vor- und Nachname" JavaScript-Code eingeben (siehe Abbildung 4), welcher auf der Seite „Seite bearbeiten" (siehe Abbildung 5) durch das Feld „Ersteller" zur Ausführung gebracht werden kann (siehe Abbildung 6).

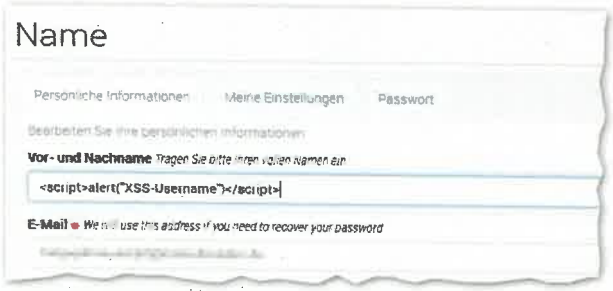


Abbildung 4: Einfügen von JavaScript innerhalb der „Benutzereinstellungen" im Feld „Vor- und Nachname".

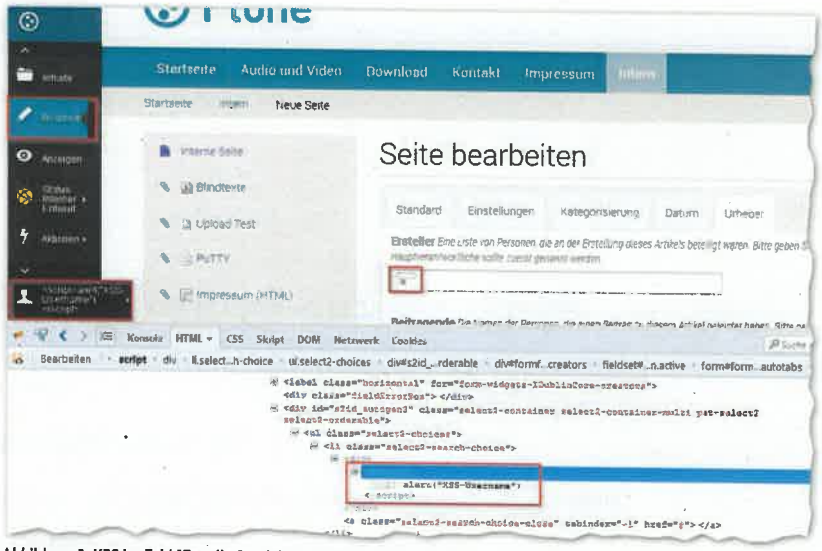


Abbildung 5: XSS im Feld "Ersteller" auf der Seite "Seite bearbeiten".

Abbildung 6: Ausführung von JavaScript im Feld "Ersteller" auf der Seite "Seite bearbeiten".

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Risikofaktor: Mittel

6.1.2.5 Cross-Site-Scripting in Seiten-Auswahl im Menüpunkt „Inhalte" in Seiten-Titel

Betroffenes Modul: Plone Core

Ergebnis:

Ein angemeldeter Nutzer kann beim Hinzufügen eines Inhalts (siehe Abbildung 7) im Feld „Titel" JavaScript-Code eingeben (siehe Abbildung 8), welcher im Menüpunkt „Inhalte" bei der Auswahl der Seite ausgeführt wird (siehe Abbildung 9).

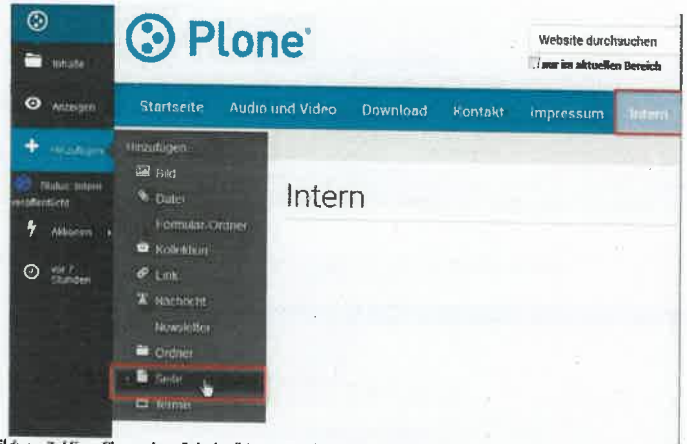


Abbildung 7: Hinzufügen eines Inhalts (hier: "Seite") im Bereich "Intern".

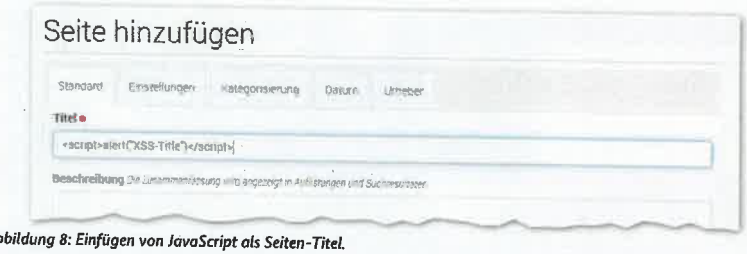


Abbildung 8: Einfügen von JavaScript als Seiten-Titel.

Abbildung 9: Ausführung von JavaScript durch die Auswahl der vormals erstellten Seite im Bereich "Inhalte".

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Risikofaktor: Mittel

6.1.2.6 Cross-Site-Scripting in „Link-Einfügen"-Dialog im Seiten-Titel

Betroffenes Modul: Plone Core

Ergebnis:

Das Feld „Titel", welches beim Hinzufügen einer Seite frei gewählt werden kann, kann JavaScript-Code enthalten (siehe Abbildung 8), welcher während der Bearbeitung von Seiten beim Hinzufügen von Links ausgeführt wird (siehe Abbildung 10 und Abbildung 11).

Abbildung 10: Ausführung von JavaScript durch Seiten-Titel über den Link-Einfügen-Dialog.

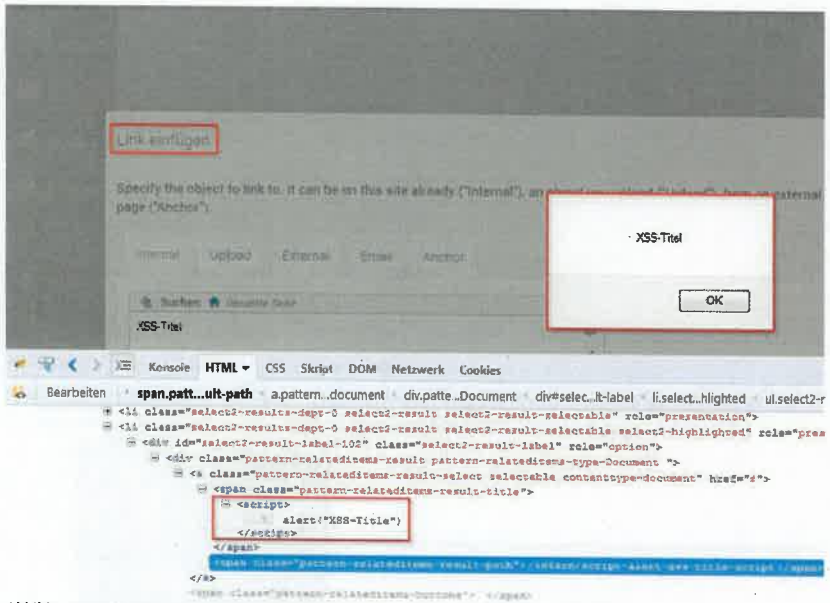


Abbildung 11: Ausführung von JavaScript durch Seiten-Titel über den Link-Einfügen-Dialog.

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Risikofaktor: Mittel

6.1.2.7 Cross-Site-Scripting in „Formularfeld anlegen"

Betroffenes Modul: Plone FormGen

Ergebnis:

Ein angemeldeter Nutzer kann das Feld eines "Formularordners" um eine "Felderläuterung" ergänzen, in die er bösartigen JavaScript-Code eintragen kann (siehe Tabelle 6). Der Parameter wird nicht ausreichend maskiert und im Frontend ausgegeben, wodurch ein XSS-Angriff möglich wird (siehe Abbildung 12).


```
POST /intern/neue-form/portal_factory/FormFileField/formfilefield.2016-02-
19.0693522237/base_edit HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://deb-plone-scenario4.cms2.local/intern/neue-
form/portal_factory/FormFileField/formfilefield.2016-02-19.0693522237/edit?
_authenticator=8b3bff342af0fbc9e6e8e3e398bf1fee5ed479c2
Cookie: I18N_LANGUAGE="de";
__ac="MLLRO93xKDlsiFtzyFPPBv1yeIkOWI1v3EdjfEcCOHA1NmM2ZGNmMTM0ZGFhMGZkZjdhNzRiM
Tk4NjY0NTNjODM2MjczNTc0IQ%3D%3D"
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------
98102883424569
Content-Length: 1337

[TRUNCATED]

-----------------------------98102883424569
Content-Disposition: form-data; name="description"

<img src="invalid.img" onerror="alert('XSS')" >

[TRUNCATED]
```

Tabelle 6: Hinzufügen eines Dateifelds in einen Formular-Ordner. Feld-Erläuterung enthält JavaScript-Code.

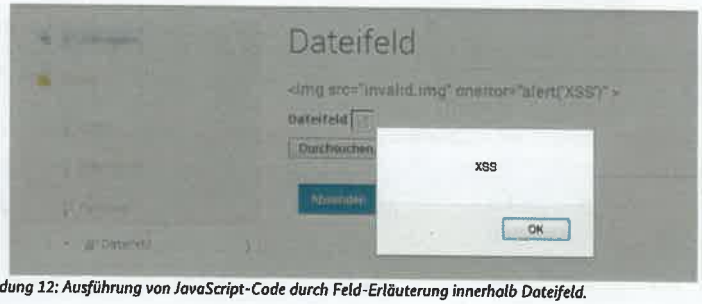


Abbildung 12: Ausführung von JavaScript-Code durch Feld-Erläuterung innerhalb Dateifeld.

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Risikofaktor: Mittel

6.1.2.8 Cross-Site-Scripting in „Nachricht anlegen" im Feld „Lead Image Caption"

Betroffenes Modul: Plone Core

Ergebnis:

Ein angemeldeter Nutzer kann Inhalte vom Typ "Nachricht" anlegen, in denen er bösartigen JavaScript-Code in das Feld "Lead Image Caption" eintragen kann (siehe Tabelle 7). Der Parameter wird nicht ausreichend maskiert und im Frontend ausgegeben, wodurch ein XSS-Angriff möglich wird.

```
POST /intern/nachricht/@@edit HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://deb-plone-scenario4.cms2.local/intern/nachricht/edit?
_authenticator=2e2f59291c18a24be6900b972fa61189dcbd5834
Cookie: I18N_LANGUAGE="de"; __ac="Cv
%2BDsclBO1lstRgNS7tBy8QBHptb6WUnEngr5AvCNqo1NmNhYTg2ZTM0ZGFhMGZkZjdhNzRiMTk4NjY
0NTNjODM2MjczNTc0IQ%3D%3D"
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------
4601219506801
Content-Length: 3202

[TRUNCATED]

-----------------------------4601219506801
Content-Disposition: form-data; name="form.widgets.ILeadImage.image_caption"

<script>alert('XSS');</script>

[TRUNCATED]
```

Tabelle 7: Anlegen einer Nachricht mit eingebettetem JavaScript-Code innerhalb des Feldes "Lead Image Caption".

Hierdurch entsteht in Multi-Benutzer-/Rollen-Umgebungen das Risiko, dass Nutzer (z.B. Autor, Redakteur, etc.) auf diesem Weg Aktionen im Namen eines anderen Nutzers mit einer Rolle mit umfangreicheren Rechten (z.B. Administrator) durchführen.

Risikofaktor: Mittel

6.1.3 Denial of Service

Beschreibung:

Ein „Denial of Service" (DoS) Attacke ist ein Angriff auf die Verfügbarkeit des Systems durch Überlastung oder Sperrung des Systems. Dabei werden einzelne Ressourcen entweder überlastet (z. B. Bandbreite, CPU-Zeit, Speicher) oder gesperrt (Nutzerkonten). Der Angriff kann sowohl nicht-persistent (Zustand besteht nur für die Dauer des laufenden Angriffs) als auch persistent (Zustand besteht nach Abbruch des Angriffs weiterhin) sein.

Ein DoS kann genauso unbeabsichtigt im normalen Betrieb verursacht werden. Ebenso kann ein DoS zur Ablenkung von parallel laufenden Angriffen dienen oder sogar das eigentliche Ziel eines Angriffes sein.

6.1.3.1 Hochladen beliebig großer Profilbilder

Betroffenes Modul: Plone Core

Ergebnis:

Ein Benutzer ist in der Lage, beliebig große Profilbilder hochzuladen. Diese werden zwar serverseitig skaliert, durch das fortlaufende Hochladen beliebig großer Profilbilder ist jedoch ein DoS-Angriff möglich.

Maßnahme:

Das Problem kann durch Einführung einer Richtlinie in Bezug auf die maximale Größe von Profilbildern behoben werden. Bilder, welche die Größenbeschränkung überschreiten, sollten vom Server abgelehnt bzw. von Plone nicht verarbeitet werden.

Risikofaktor: Leicht

6.1.3.2 Absenden beliebig großer Kommentare

Betroffenes Modul: Plone Core

Ergebnis:

Ein nicht angemeldeter Benutzer ist in der Lage, beliebig große Kommentare zu verfassen. Hierbei gibt Plone keine Restriktionen hinsichtlich der Größe des Kommentars vor, sodass im Test Kommentare in der Größenordnung von 10 MB hochgeladen werden konnten. Durch das fortlaufende Absenden großer Kommentare ist hierdurch ein DoS-Angriff möglich (siehe Tabelle 8).

```
POST /audio-und-video/document_view HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://deb-plone-scenario4.cms2.local/audio-und-video
Cookie: I18N_LANGUAGE="de"
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------
92402929314851
Content-Length: 809

[TRUNCATED]

-----------------------------92402929314851
Content-Disposition: form-data; name="form.widgets.text"

AAAAAAAAAAAAA[...]

[TRUNCATED]
```

Tabelle 8: Hochladen eines beliebig großen Kommentars.

Maßnahme:

Es sollte eine Richtlinie in Bezug auf die maximale Größe von Kommentaren eingeführt und geprüft werden.

Risikofaktor: Mittel

6.1.3.3 Absenden beliebig vieler Kommentare

Betroffenes Modul: Plone Core: Härtung

Ergebnis:

Ein nicht angemeldeter Benutzer ist in der Lage, beliebig viele Kommentare zu verfassen. Durch das fortlaufende Absenden großer Kommentare ist hierdurch ein DoS-Angriff möglich (vgl. Tabelle 9).

Maßnahme:

Durch Verwendung eines CAPTCHAs kann ein ein automatisiertes Belegen zahlreicher Ressourcen unterbunden werden. Eine CAPTCHA-Funktionalität ist im Core enthalten und kann dort aktiviert werden.

Risikofaktor: Mittel

6.1.3.4 Absenden beliebig großer Kommentare im Kontakt-Formular

Betroffenes Modul: Plone Core

Ergebnis:

Ein nicht angemeldeter Benutzer ist in der Lage, beliebig große Kommentare im Kontakt-Formular zu verfassen (siehe Tabelle 9). Hierbei gibt Plone keine Restriktionen hinsichtlich der Größe des Kommentars vor, sodass im Test Kommentare in der Größenordnung von 10 MB hochgeladen werden konnten. Durch das fortlaufende Absenden großer Kommentare ist hierdurch ein DoS-Angriff möglich.

```
POST /kontakt HTTP/1.1
Host: deb-plone-scenario4.cms2.local
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101
Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://deb-plone-scenario4.cms2.local/kontakt
Cookie: I18N_LANGUAGE="de"
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------
29041088917062
Content-Length: 1337

[TRUNCATED]

-----------------------------29041088917062
Content-Disposition: form-data; name="comments"

AAAAAAAAAAAAAAAAAAA [...]

[TRUNCATED]
```

Tabelle 9: Hochladen eines beliebig großen Kommentars durch das Kontaktformular.

Maßnahme:

Es sollte eine Richtlinie in Bezug auf die maximale Größe von Kommentaren im Kontakt-Formular eingeführt werden.

Risikofaktor: Mittel

6.1.3.5 Absenden beliebig vieler Kommentare im Kontakt-Formular

Betroffenes Modul: Plone Core

Ergebnis:

Ein nicht angemeldeter Benutzer ist in der Lage, beliebig viele Kommentare im Kontakt-Formular zu verfassen (vgl. Tabelle 9). Durch das fortlaufende Absenden großer Kommentare ist hierdurch ein DoS-Angriff möglich.

Maßnahme:

Durch Verwendung eines CAPTCHAs kann ein ein automatisiertes Belegen zahlreicher Ressourcen unterbunden werden.

Risikofaktor: Mittel

6.1.4 Brute Force von Benutzerpasswörtern

Betroffenes Modul: Plone Core

Ergebnis:

Das Passwort lässt sich über Brute Force angreifen, da die Anmeldemaske es beliebig oft zulässt, eine Kombination aus Benutzername und Passwort einzugeben.

Maßnahme:

Mehrere Maßnahmen sind möglich:

- Ansteigende zeitliche Verzögerung (z. B. Verdopplung der Wartezeit bei jedem Versuch) für eine erneute Eingabe eines Passwortes nach einem Fehlversuch.
- Sperren des Kontos nach einer vorgegebenen Anzahl an Fehlversuchen (typischerweise 5).
- Verwendung einer fixen Verzögerung nach erfolgreichen oder nicht erfolgreichen Anmeldeversuchen. Auf diese Weise können keine Rückschlüsse auf Basis der zeitlichen Verzögerung hinsichtlich der Existenz eines Kontos geschlossen werden.
- Verwendung von CAPTCHA, um ein automatisiertes Ausprobieren zu unterbinden.

Um eine höhere Sicherheit zu erreichen, ist es oftmals sinnvoll, zwei oder mehrere der genannten Maßnahmen miteinander zu kombinieren.

Es sollte jedoch bei der Konzipierung der Gegenmaßnahme darauf geachtet werden, dass sie keine Möglichkeit bietet, einen oder gar alle Benutzer dauerhaft auszusperren.

Hierfür kann bspw. das Plugin „Products.LoginLockout" genutzt werden.

Risikofaktor: Mittel

6.1.5 Missbrauch der Anwendung als SPAM-Verteiler

Betroffenes Modul: Plone Core

Ergebnis:

Die Passwort-vergessen-Funktion (https://deb-plone-scenario4.cms2.local/mail_password_form) kann beliebig oft mit einem validen Benutzernamen aufgerufen werden. Da bei jedem Aufruf eine neue E-Mail an den Benutzer geschickt wird, kann die Anwendung als Spam-Verteiler genutzt werden, der das E-Mail-Postfach des Benutzers füllt.

Maßnahme:

Durch Implementierung eines CAPTCHA-Schutzes kann der automatisierte Aufruf der Passwort-vergessen-Funktion erschwert werden.

Risikofaktor: Leicht

6.1.6 Cross-Site Request Forgery

Beschreibung:

Bei Cross-Site Request Forgery (CSRF, auch XSRF oder „Session-Riding") wird ein Opfer dazu gebracht, unwissentlich einen präparierten HTTP-Request abzuschicken. Dieser Request löst dann im Rahmen einer laufenden Session eine Aktion im Namen des Opfers aus. Dies geschieht beispielsweise beim Besuch einer bösartigen Webseite, in der ein entsprechender Link z. B. als img-, script- oder IFrame-Tag auf eine andere Anwendung derart enthalten ist, dass dieser Link für das Opfer nicht zu erkennen ist. Der Browser folgt aber quasi im Hintergrund diesem Link und löst erfolgreich eine Aktion aus, sofern das Opfer aktuell eine gültige Session für diese andere Anwendung hat.

6.1.6.1 CSRF-Token wird als GET-Parameter übergeben

Betroffenes Modul: Plone Core

Ergebnis:

Plone benutzt zum Schutz vor CSRF in der URL den Parameter „_authenticator" (siehe Abbildung 13). Hierdurch ist es theoretisch möglich, dass das Token beim Besuch einer externen Seite durch einen platzierten Link im Referer überliefert wird.

Maßnahme:

Das CSRF-Token sollte innerhalb eines (versteckten) Formularfelds mittels HTTP-POST übertragen werden.

Abbildung 13: Übergabe des CSRF-Tokens „_authenticator" innerhalb der URL.

Risikofaktor: Leicht

6.1.6.2 CSRF-Token besitzt unzureichende Zufälligkeit

Betroffenes Modul: Plone Core

Ergebnis:

Plone benutzt zum Schutz vor CSRF in der URL den Parameter „_authenticator" (siehe Abbildung 13). Für einen Nutzer existieren vier Authentifikatoren, die sich über die Zeit nicht ändern. Sofern ein Angreifer in den Besitz von einem der vier Authentifikatoren eines anderen Nutzers kommt, kann er einen zeitunabhängigen CSRF-Angriff starten.

Maßnahme:

Die verwendeten CSRF-Tokens sollten für jede Session neu generiert werden.

Risikofaktor: Leicht

6.1.7 Fehlendes Secure Cookie-Flag

Betroffenes Modul: Plone Core

Beschreibung:

Das Attribut „secure" verhindert, dass Cookies unverschlüsselt per HTTP versendet werden. Dies könnte geschehen zum einen durch (ggf. versehentlich) unverschlüsselte Inhalte einer Webanwendung, zum

anderen aber auch durch aktive Angriffe, z. B. durch Injizierung oder Präsentation unverschlüsselter Links bzw. Referenzen, aufgrund derer der Browser die Session-ID unverschlüsselt versenden würde.

Ergebnis:

Das „secure"-flag für den Cookie-Wert „__ac" ist nicht gesetzt (siehe Abbildung 14). Sofern ein angemeldeter Benutzer die Seite über HTTP aufruft, wird der Cookie im Klartext übertragen, sodass ein Man-In-The-Middle-Angriff möglich ist. Die Übertragung des Cookies im Klartext erfolgt auch, wenn der Server eine HTTPS-Weiterleitung aktiv geschaltet hat (siehe Abbildung 15).

Abbildung 14: Fehlendes Secure Cookie-Flag.

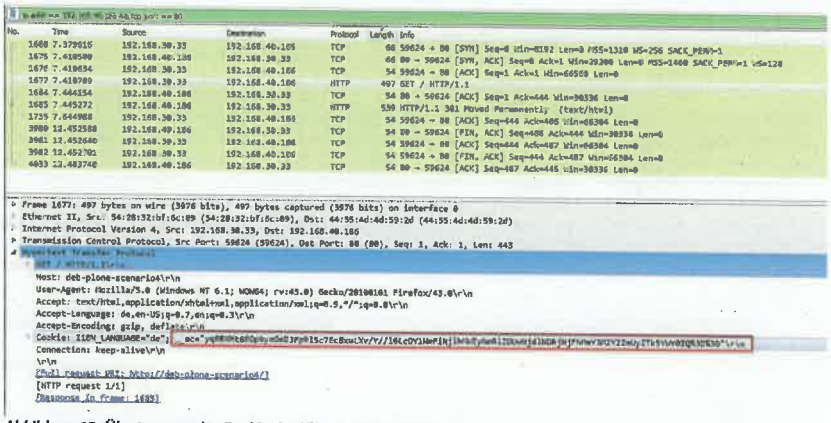


Abbildung 15: Übertragung des Cookies im Klartext über HTTP.

Maßnahme:

Für den Cookie-Parameter „__ac" sollte das secure-Flag gesetzt werden.

Risikofaktor: Leicht

6.1.8 Enumeration von Benutzernamen durch Passwort-Vergessen-Funktion

Betroffenes Modul: Plone Core

Beschreibung:

Ziel der User Enumeration ist das Sammeln von validen Benutzerdaten, welche für die Anmeldung benötigt werden und einen ersten Schritt zur Übernahme von Benutzerkonten darstellen. Dies kann beispielsweise automatisiert (Brute Force) durch eindeutige Fehlermeldungen bei Registrierung und Anmeldung realisiert werden.

Ergebnis:

Ein Angreifer kann Benutzernamen durch die Passwort-Vergessen-Funktion (https://deb-plone-scenario4.cms2.local/mail_password_form) erraten, indem er systematisch Nutzernamen ausprobiert. Existiert ein Nutzername, antwortet die Anwendung mit „Bestätigung des Passwort-Zurücksetzens gesendet." Existiert der Nutzernamen nicht, wird eine Fehlermeldung angezeigt.

Maßnahme:

Fehlermeldungen bei Registrierung und „Passwort-vergessen"-Funktion dürfen nicht auf vorhandene Benutzerkonten schließen lassen. Bei der Registrierungsfunktion sollte bspw. keine Fehlermeldung erzeugt werden, sondern der Benutzer via E-Mail darüber informiert werden, dass über seine E-Mail-Adresse versucht wurde, sich an der Anwendung zu registrieren. Im Falle der „Passwort-vergessen"-Funktion sollte die Meldung „Passwort versendet" trotz der nicht vorhandenen E-Mail Adresse erscheinen.

Darüber hinaus sollte wenigstens ein persönliches Merkmal verifizieren, dass die „Passwort vergessen"-Funktion nicht durch Brute Force missbraucht wird. Weitere Informationen sind unter der folgenden URL verfügbar: https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet.

Risikofaktor: Leicht

6.1.9 Formulare mit Autocomplete

Betroffenes Modul: Plone Core

Beschreibung:

Im Anmeldeformular können Benutzereingaben automatisch vervollständigt werden. Diese Daten werden lokal auf dem Client (im Browser) abgelegt und können dort von einem Angreifer oder anderen Benutzer des Systems ermittelt werden. Daher sollten insbesondere Formulare, die zur Eingabe von Anmeldedaten (Benutzername und Passwort/PIN) verwendet werden, diese Funktion deaktivieren. Auch für andere sensible Formularfelder (z.B. Bankverbindungen, Kreditkartennummern) sollte die automatische Vervollständigung nicht verwendet werden.

Ergebnis:

Im Anmeldeformular ist das Autocomplete-Attribut nicht deaktiviert.

Maßnahme:

Eine Webanwendung sollte durch Setzen des Attributs 'autocomplete = "off" ' verhindern, dass sensible Formulardaten vom Browser gespeichert und bei der nächsten Verwendung automatisch vervollständigt werden.

Siehe auch:
https://www.owasp.org/index.php/Session_Management#Prevent_Sensitive_Data_from_Being_Cached_on_Client_Side

Risikofaktor: Leicht

6.1.10 Schwache Passwort-Richtlinien

Betroffenes Modul: Plone Core

Beschreibung:

Passwörter sollten eine gewisse Mindestkomplexität aufweisen, die vor dem Setzen eines neuen Passwortes überprüft wird. Diese Komplexität erschwert einem möglichen Angreifer das Erraten des Passwortes. Erfüllt das neue Passwort diese Bedingungen nicht, muss es abgewiesen werden.

Ergebnis:

Die bestehende Kennwort-Richtlinie gibt vor, dass Benutzer-Passwörter aus mindestens 5 beliebigen Zeichen bestehen müssen.

Maßnahme:

Um eine ausreichende Komplexität des Passwortes zu gewährleisten, sollte es entsprechend überprüft werden. Als sicher anzusehen sind beispielsweise folgende Bildungsregeln:

- Mindestlänge 8 Zeichen und
- bestehend aus mindestens drei der folgenden 4 Kategorien: Kleinbuchstaben, Großbuchstaben, Zahlen und Sonderzeichen.

Hierfür kann bspw. das Plugin „Products.PasswordStrength" genutzt werden.

Risikofaktor: Leicht

6.1.11 Fehler in Authentifizierung und Session-Management

Beschreibung:

Anwendungsfunktionen, die die Authentifizierung und das Session-Management umsetzen, werden oft nicht korrekt implementiert. Dies erlaubt es Angreifern, Passwörter oder Session-Tokens zu kompromittieren oder die Schwachstellen so auszunutzen, dass sie die Identität anderer Benutzer annehmen können.

6.1.11.1 Passwort-Änderung führt nicht zur Invalidierung aller Sessions und der Aushandlung eines neuen Session-Cookies

Betroffenes Modul: Plone Core

Beschreibung:

Die Funktion zur Passwortänderung wird oftmals dazu genutzt, um den unrechtmäßigen Gebrauch eines Benutzerkontos durch Dritte zu unterbinden.

Ergebnis:

Die Änderung des Passwortes für einen Nutzer führt nicht zur Invalidierung aller laufenden Sessions und der Aushandlung eines neuen Cookies. Hierdurch ist es für einem Benutzer nicht möglich, Benutzer die unrechtmäßig das Benutzerkonto nutzen, durch die Änderung des Passworts auszusperren.

Maßnahme:

Eine Passwort-Änderung sollte alle laufenden Sessions eines Nutzers invalidieren und in der Aushandlung eines neuen Cookies resultieren.

Risikofaktor: Mittel

6.1.11.2 Mehrfachanmeldung möglich

Betroffenes Modul: Plone Core

Beschreibung:

Durch eine Mehrfachanmeldung kann ein Benutzerkonto von mehreren Personen genutzt werden. Hierdurch wird die Erkennung einer Übernahme des Benutzerkontos durch einen Angreifer und eine Nachvollziehbarkeit der ausgeführten Aktionen erschwert.

Ergebnis:

Der Testnutzer konnte sich mehrmals mit den gleichen Zugangsdaten von unterschiedlichen IP-Adressen am System anmelden.

Maßnahme:

Das Problem kann durch Einführung einer Geräteverwaltung, die es einem Benutzer erlaubt, verschiedene Geräte für sein Benutzerkonto zu registrieren, minimiert werden. Für ein Benutzerkonto sollte pro freigeschaltetem Gerät nur eine Sitzung zur selben Zeit aktiv sein. Eine zweite Anmeldung unter dem gleichen Gerät mit dem gleichen Benutzerkonto sollte verhindert werden. Jede mit einem Gerät getätigte Benutzeraktion sollte aufgezeichnet werden.

Hierfür kann bspw. das Plugin „Products.NoDuplicateLogin" genutzt werden.

Risikofaktor: Leicht

6.1.11.3 Abmeldung führt nicht zur Invalidierung der Session

Betroffenes Modul: Plone Core

Beschreibung:

Wird beim Abmeldeprozess die Session nur Client-seitig terminiert, kann ein Angreifer die Session wieder aufnehmen, sofern er in den Besitz des Session-Cookies kommt.

Ergebnis:

Beim Abmeldevorgang wird der Cookie-Wert „__ac" Client-seitig auf null gesetzt. Wird der vormalige Wert wieder gesetzt, kann die Session wieder aufgenommen werden.

Maßnahme:

Die Session sollte beim Abmeldevorgang serverseitig terminiert werden.

Risikofaktor: Leicht

6.1.11.4 Cacheable HTTP(S) responses

Betroffenes Modul: Plone Core

Beschreibung:

Webbrowser können Internetseiten zur Verbesserung der Performance zwischenspeichern. Hierdurch müssen Inhalte nicht erneut heruntergeladen werden. Um zu verhindern, dass sensitive Daten vom Webbrowser zwischengespeichert werden, kann im HTTP-Response-Header die Direktive „Cache-control: no-store" bzw. „Pragma: no-cache" gesetzt werden.

Ergebnis:

In den HTTP(S)-Responses von Plone fehlt der HTTP-Header „Cache-control: no-store" bzw. „Pragma: no-cache" (siehe Abbildung 16). Sofern ein Angreifer Zugriff auf die Cache-Dateien erhält, können somit sensible Daten ausgelesen werden.

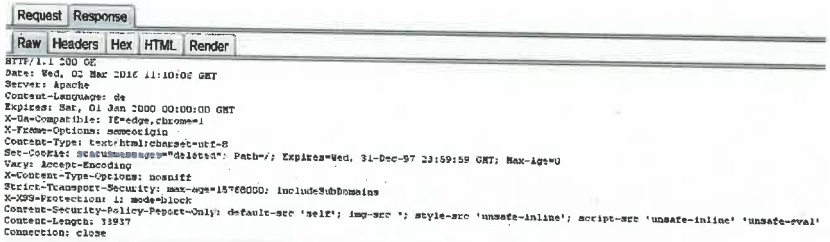


Abbildung 16: Fehlender HTTP-Header "Cache-control: no store" bzw. „Pragma: no-cache" innerhalb zugriffsbeschränkter Bereiche.

Maßnahme:

Das Caching sollte für zugangsbeschränkte Bereiche über die HTTP-Header-Direktive „Cache-control: no-store" bzw. „Pragma: no-cache" verhindert werden.

Risikofaktor: Leicht

6.1.11.5 Session-Timeouts zu groß gewählt

Betroffenes Modul: Plone Core: Härtung

Beschreibung:

Sessions speichern die Anmeldung von Benutzern an einem System bzw. an einer Applikation für die Dauer der Verwendung. Sicherheitskritische Sessions sollten nach einer festgelegten Zeitspanne der Nichtnutzung ablaufen. Die potentielle Angriffsfläche wird somit reduziert.

Ergebnis:

Im Modul plone.session ist ein Timeout von 12 Stunden gewählt.

Maßnahme:

Der Session-Timeout kann über das Management-Interface (https://deb-plone-scenario4.cms2.local/acl_users/session/manage_propertiesForm) angepasst werden. Gewöhnlich sollten Sessions nicht länger als zwei Stunden gültig sein.

Risikofaktor: Leicht

6.1.11.6 Mehrmalige Vergabe des selben Session-Cookies

Betroffenes Modul: Plone Core

Ergebnis:

Das mehrmalige Anmelden des selben Nutzers innerhalb einer kurzen Zeitspanne (ca. 5-10 Sekunden) resultiert in der Wiederverwendung des Cookies. Sollte der in diesem Zeitfenster für den Cookie generierte zufällige Wert auch für andere Nutzer genutzt werden, kann der für einen beliebigen Nutzer generierte Cookie von einem Angreifer erraten/berechnet werden.

Maßnahme:

Für jede Anmeldung sollte ein anderer Cookie-Wert generiert werden.

Risikofaktor: Hinweis

6.1.12 Verschleierung des Erstellers einer Seite

Betroffenes Modul: Plone Core

Beschreibung:

Im Bereich "Intern" werden alle Seiten und ihre Ersteller aufgelistet. Für den Urheber einer Seite ist es möglich, als Ersteller einen Benutzer seiner Wahl einzutragen. Hierdurch ist es einem Angreifer zum Beispiel möglich, eine falsche Identität vorzugeben.

Hierzu muss im Dialog „Seite bearbeiten", Tab „Urheber", Feld „Ersteller" ein beliebiger Nutzer hinzugefügt und mittels Drag-And-Drop an vorderste Stelle geschoben werden. Im Bereich „Intern" wird innerhalb der Tabelle in der Spalte „Ersteller" der gewählte Nutzer angezeigt.

Hierdurch wird zwar nicht der Wert des Erstellers in der Content Owner Information verändert, d.h. es bleibt weiterhin eindeutig zuordenbar, wer den Content angelegt oder verändert hat. Allerdings hat es für einen wenig versierten Nutzer den Anschein, dass der Inhalt von einer anderen Person stammt.

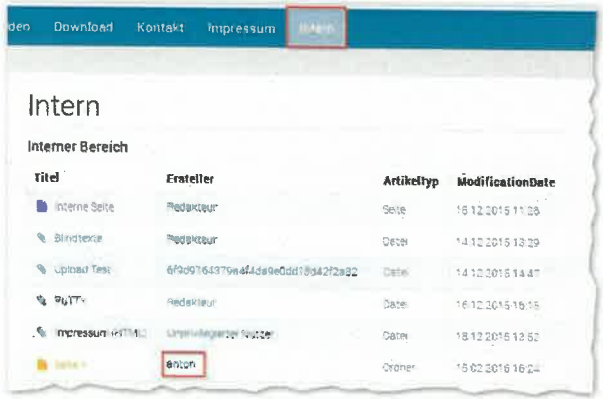


Abbildung 17: Das Feld "Ersteller" für "Seite A".

Abbildung 18: Bearbeiten von "Seite A": Nutzer "Pentest Admin" an erste Stelle verschieben.

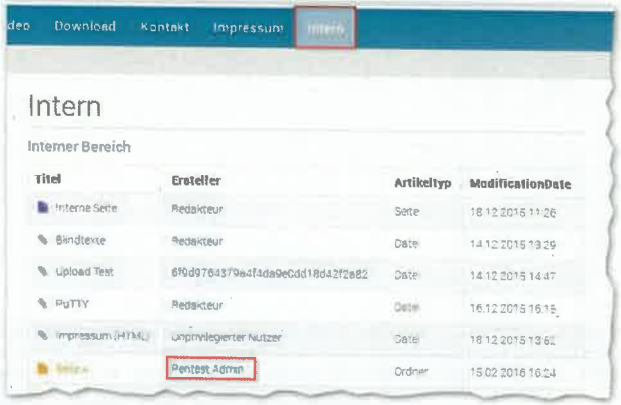


Abbildung 19: Das Feld "Ersteller" für "Seite A" nach der Bearbeitung.

Maßnahme:

Wird ein anderer Nutzer in das Feld "Ersteller" eingetragen, sollte dieser darüber benachrichtigt werden und entsprechend bestätigen bzw. ablehnen können.

Risikofaktor: Leicht

6.1.13 Berechtigungsüberschreitung

Beschreibung:

Einem Benutzer der Anwendung sollte es nicht möglich sein, außerhalb seiner Zugriffsberechtigungen auf Funktionen oder Informationen des Systems zugreifen zu können. Dies ist dann der Fall, wenn eine Funktion die erforderlichen Berechtigungen vor der Ausführung, nicht überprüft.

Beispielsweise ruft ein Angreifer eine Methode direkt auf, ohne dafür die HTML-Seite aufzurufen, die diese Funktion enthält. Zwar ist ihm die HTML-Seite nicht direkt zugänglich, jedoch die zugehörige Backend-Funktion, da diese keine eigene Prüfung der Berechtigung durchführt.

6.1.13.1 Unberechtigter Zugriff auf Daten oder Verzeichnisse durch Kopieren-Funktion

Betroffenes Modul: Plone Core

Ergebnis:

Einem angemeldeten Nutzer ist es durch das Kopieren eines Ordners möglich, alle darin enthaltenen Dateien anzuzeigen, auch wenn er vormals keine Freischaltung hierfür hatte.

Vorbereitung: Hierzu meldet sich Benutzer „A" am System an und erstellt innerhalb des internen Bereichs einen Ordner mit dem Namen „Ordner A" mit einer darin befindlichen Seite mit dem Namen „Seite A". Für „Seite A" wird die Freigabe entsprechend angepasst, sodass "Berechtigungen von übergeordneten Ordnern

übernehmen" deaktiviert wird (siehe Abbildung 20). Weiterhin wird „Seite A" auf Privat geschaltet (siehe Abbildung 21).

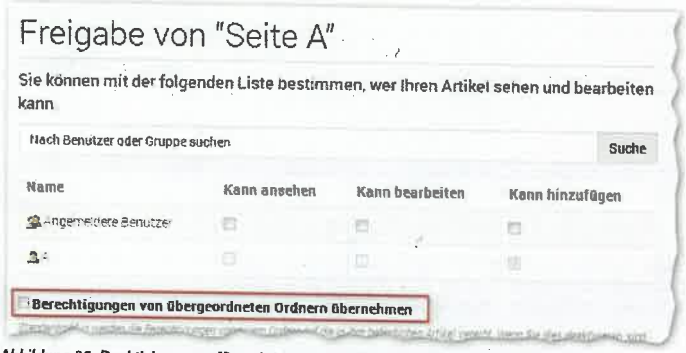


Abbildung 20: Deaktivieren von "Berechtigungen von übergeordneten Ordnern übernehmen".

Abbildung 21: "Seite A" auf Privat schalten.

Angriff: Benutzer „B" meldet sich am System an und wählt im internen Bereich „Ordner A" aus. Im aktuellen Zustand ist „Seite A" für Benutzer „B" nicht sichtbar. Durch die Betätigung von „Aktionen" > „Kopieren" und „Aktionen" > „Einfügen" wird ein weiterer Ordner mit dem Namen „Ordner A" (hier: „Kopie von Ordner A") erstellt. Die Auswahl von „Kopie von Ordner A" zeigt „Seite A" mit dem Status „Intern" an (siehe Abbildung 22).

Abbildung 22: Kopiervorgang von "Ordner A" durch Nutzer "B" macht "Seite A" sichtbar.

Maßnahme:

Bei Kopiervorgängen muss der Status der Inhalte mit übernommen werden.

Risikofaktor: Mittel

6.1.13.2 Unberechtigter Zugriff auf Daten oder Verzeichnisse durch fehlerhaftes Berechtigungssystem

Betroffenes Modul: Plone Core

Ergebnis:

Als registrierter Benutzer ist es über die Seite „Intern" möglich, alle darin befindlichen Inhalte trotz fehlender Berechtigung anzusehen. Die Berechtigung „Kann ansehen" wird erst aktiv, wenn der Status des Inhalts auf „Privat" gesetzt wird.

Maßnahme:

Die „Kann ansehen"-Berechtigung sollte nicht abhängig von dem Status („Privat") des Inhalts sein.

Risikofaktor: Mittel

6.1.13.3 Kopieren und Hinzufügen von sichtbaren Inhalten trotz fehlender Berechtigung

Betroffenes Modul: Plone Core

Ergebnis:

Als registrierter Benutzer ist es über die Seite „Intern" möglich, alle sichtbaren Inhalte mittels „Aktionen" > „Kopieren" beliebig oft zu duplizieren.

Da der Nutzer anschließend keinen Zugriff auf das Objekt hat, ist die Funktion nur bedingt sinnvoll.

Maßnahme:

Die Kopieren-/Einfügen-Operation sollte nur für Inhalte zur Verfügung stehen, für die der entsprechende Benutzer die „Kann bearbeiten"-Berechtigung aufweist.

Alternativ können dem Benutzer Zugriffs- bzw. Bearbeitungsrechte auf das kopierte Objekt eingeräumt werden.

Risikofaktor: Leicht

6.1.13.4 Unberechtigter Zugriff auf Daten oder Verzeichnisse durch nicht-intuitives Berechtigungssystem

Betroffenes Modul: Plone Core

Ergebnis:

Ist der Status eines Inhalts auf „Privat" gesetzt, die Freigabe „Berechtigungen von übergeordneten Ordnern übernehmen" jedoch aktiviert, können andere registrierte Benutzer den Inhalt weiterhin einsehen.

Dieses Verhalten könnte insbesondere bei ungeschulten Nutzern für Verwirrung sorgen und die unbeabsichtigte Veröffentlichung sensibler Daten bewirken.

Maßnahme:

Auf „Privat" gesetzte Inhalte sollten ungeachtet von der gesetzten Freigabe nur für den betreffenden Benutzer einsehbar sein.

Risikofaktor: Leicht

6.1.13.5 Unberechtigter Zugriff auf Daten oder Verzeichnisse innerhalb privat geschalteter Unterordner

Betroffenes Modul: Plone Core

Ergebnis:

Ist der Status eines Ordners auf „Privat" gesetzt und die Freigabe „Berechtigungen von übergeordneten Ordnern übernehmen" deaktiviert, können andere registrierte Benutzer den Ordner nicht sehen. Werden Inhalte in diesen Ordner hinzugefügt, erben diese den Status „Privat" jedoch nicht, sodass andere Nutzer über die URL auf diese Inhalte zugreifen können.

An dieser Stelle ist das Rechtesystem für unerfahrene Nutzer nicht intuitiv.

Maßnahme:

Inhalte innerhalb Privat-geschalteter Ordner sollten für andere Nutzer nicht einsehbar sein.

Risikofaktor: Hinweis

6.1.13.6 Fehlende Freigabe- und Status-Einstellmöglichkeiten im Hinzufügen-Dialog

Betroffenes Modul: Plone Core

Beschreibung:

Der Hinzufügen-Dialog weist keine Einstellungsmöglichkeiten zum Setzen der Freigabe und des Status auf. Hierdurch ist es einem Benutzer nicht direkt möglich, Inhalte für einen eingeschränkten Benutzerkreis zu erzeugen, sodass diese bis zum nachträglichen Setzen der Einstellungen allgemein einsehbar sind.

Maßnahme:

Freigabe- und Status-Einstellmöglichkeiten sollten in den Hinzufügen-Dialog aufgenommen werden.

Risikofaktor: Hinweis

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 Plone 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 Plone 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.

	Beschreibung
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.

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.