- Hackathon Gruppe 1.1
  - Angaben zum Projekt
- Einleitung
  - Motivation
  - Aufgabenstellung
    - Zeitplan
- Entwurf Meilenstein 1
  - Anforderungen/Arbeitspakete
    - Anforderungsanalyse und Konzeptentwicklung
    - Design und Entwicklung des Web-Formulars zur Stellenanzeigen-Einreichung
    - Backend-Entwicklung und Datenpool-Management
    - Sicherheitsprüfung und Schadcode-Vermeidung
    - Qualitätssicherung und Testing
    - Deployment und Rollout des Systems
    - Dokumentation und Support
  - Systementwurf
    - Systemarchitektur
    - Erscheinungsbild der Webseite/Mockup
    - Infrastruktur
    - Webserver
    - Webanwendung
  - Komponenten des Systems
    - Web-Formular für die Einreichung
    - PDF-Upload und Sicherheitsüberprüfung
    - Datenpool und Verwaltung
    - Öffentliche Stellenbörse
    - Nützliche Features
      - Benutzerregistrierung und -verwaltung
      - Statistiken und Analysen
      - Feedback-Funktion
      - Responsive Design
    - Erinnerungsfunktion
  - Sicherheitsbetrachtungen
    - CAPTCHA [9]
      - Cloudflare Turnstile
    - Sicherheitsüberprüfung der PDF-Dateien
  - Testplan
    - Bedeutung des Testplans
    - Unit Tests
    - Integrationstests
    - Sicherheitstests
    - Testplan Checkliste
  - Sicherheitsanforderungen
    - Bedrohungsmodellierung
      - STRIDE-Bedrohungsmodell
        - Identitätsvortäuschung

            - Manipulation
            - Nicht-Abstreitbarkeit
            - Informationspreisgabe
            - Denial of Service
            - Rechteerweiterung
- Realisierung Meilenstein 2
    - Arbeitsumgebung
        - Implementierung "Cloudflare Turnstile" [8]
            - Widget in Webseite einbinden
            - Validierung der serverseitigen Response
            - Zurücksetzen des Widgets bei Inaktivität
        - Automatisierte Löschung
            - Skript für Datenlöschung
            - Testen des Skripts auf Funktion
        - Textfelder
        - Dropdown-Menüs
        - Button
        - Datei-Upload
        - Formularverarbeitung
            - Zusammenspiel der Funktionalitäten
    - PDF-Sicherheitsüberprüfung
        - Problematik
        - Vorgehen
            - peepdf-3 [5]
            - PDFiD [4]
            - PDFExaminer
            - QuickSand
            - Simple-PDF-Analyzer [12]
            - APIs
            - Hochschul E-Mail-Server
        - Fazit
    - Test
        - Systemvoraussetzungen & Vorbereitung
        - Detaillierte Testszenarien Funktionalität
            - Erreichbarkeit und Grundfunktionen
            - PDF-Upload-Prozess
            - Metadaten-Generierung
            - Zugriffsrecht
            - Zeitgesteuerte Systemfunktionen
            - Passwortgeschützte PDFs
            - Sicherheitsaspekte
                - Dateigrößenmanagement
                - Dateityp-Validierung
                - Sicherheitsscans
                - Dateinamenskonventionen
                - Websicherheit

        - Dokumentation und Abschlussbewertung
    - Wirtschaftlichkeit
- Anleitungen Meilenstein 3
    - Installationsanleitung
    - Betreiberdokumentation
- Cross-Grading Meilenstein 4
    - System under Test
    - Testplan
    - Installation
    - Testdurchführung
    - .
    - Fazit
- Zusammenfassung und Fazit
    - Arbeitspakete und Aufgabenverteilung:
- Quellenverzeichnis

Hackathon Gruppe 1.1

Angaben zum Projekt

- **Datum**: 30.10.2024
- **Projektname**: Stellenbörse der HSNR
- **Gruppennummer**: Thema 1 - Gruppe 1
- **Namen der Gruppenmitglieder und deren Verantwortlichkeiten**:
    - Julius Birkhofen, 1490505, Projektleitung, Systementwurf
    - Ben Bolten, 1480250, Recherche CAPTCHA
    - David Borns, 1276321, Recherche Schadcodeerkennung
    - Philipp Pernot, 1475434, Testplan
    - Lukas Brünken, 1469512, Systementwurf
    - Ümit Satir, 1466470, Sicherheitsanforderungen, Testplan

Einleitung

Die Hochschule Niederrhein betreibt eine Stellenbörse, die Studierenden interne und externe Angebote wie Praxisphasen, Werkstudentenstellen oder Abschlussarbeiten bietet. Bisher erfolgte die Einreichung der Stellenanzeigen dezentral. Die Unternehmen übermittelten ihre Ausschreibungen per E-Mail an verschiedene Fachbereiche, die diese manuell einstellten. Dies stellte einen hohen Verwaltungsaufwand dar und führte zu redundanten Stellenanzeigen, welche von verschiedenen Fachbereichen eingestellt wurden.

Das Ziel dieses Projekts ist es, ein zentrales Einreichsystem zu entwickeln, das die Erfassung und Verwaltung der Anzeigen vereinfacht. Über ein Web-Formular sollen Unternehmen in der Lage sein, einseitige PDF-Dokumente hochzuladen und zusätzliche Metadaten einzugeben. Die Dateien werden zentral in einem Verzeichnis gespeichert und regelmäßig von Mitarbeitenden der Hochschule auf dem Stellenportal veröffentlicht. Außerdem wird besonderer Wert auf die Sicherheitsaspekte gelegt, um zu verhindern, dass mit

der hochgeladenen PDF-Datei Schadsoftware eingeschleust wird. Ebenfalls wird die Speichergröße und der Dateityp des Uploads kontrolliert. Durch das Einbinden von CAPTCHAs wird eine Sicherheitsvorkehrung gegen Spam und ein Überladen des Speichers implementiert. Die im Datenpool hinterlegten Dokumente werden automatisch nach einer Frist von 4 Wochen gelöscht.

Optional: Erweiterungsmöglichkeiten, wie z. B. Erinnerungen an Unternehmen zur Verlängerung der Anzeigen, werden ebenfalls berücksichtigt.

Die Dokumentation beschreibt den vollständigen Entwicklungsprozess, von der Anforderungsanalyse über Architektur und Implementierung bis hin zur Testphase. Sie zeigt auf, wie das System den Verwaltungsaufwand der Hochschule reduziert und ein effizientes Stellenausschreibungssystem entwirft.

Motivation

Unsere Motivation für die Entwicklung dieses Projekts liegt darin, Effizienz, Transparenz und Sicherheit in die Prozesse der Stellenbörse zu bringen. Mit einem zentralen Einreichsystem werden Fehler vermieden und Unternehmen erhalten eine benutzerfreundliche Möglichkeit, ihre Ausschreibungen direkt einzureichen. Gleichzeitig ermöglichen automatisierte Metadatenvorschläge, dass relevante Stellenangebote schneller und präziser im System erfasst werden.

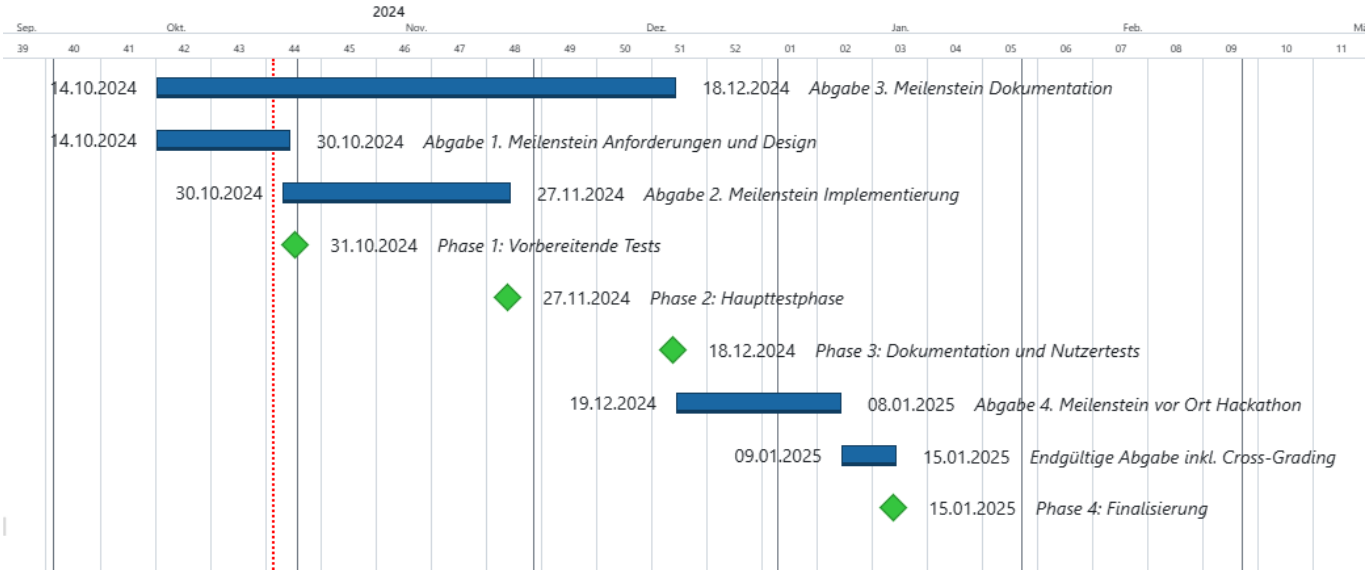
Durch unser Studium besteht ein Fokus auf die Sicherheit der Architektur. Durch gezielte Sicherheitsmaßnahmen schützen wir das System vor potenziellen Angriffen und verhindern das Einschleusen von Schadsoftware. Dies gewährleistet nicht nur den störungsfreien Betrieb, sondern stärkt auch das Vertrauen der beteiligten Unternehmen und Nutzenden.

Langfristig wird dieses System nicht nur zur Entlastung der Fachbereiche beitragen, sondern auch die Zusammenarbeit zwischen Hochschule und Unternehmen fördern. Studierende profitieren von einem strukturierten und stets aktuellen Stellenangebot, während Untenehmen flexibler und direkter mit der Hochschule kooperieren können.

Mit diesem Projekt möchten wir einen zukunftsorientierten Beitrag zur Optimierung der Hochschulprozesse leisten und zeigen, wie digitale Lösungen den Alltag von Verwaltungen, Unternehmen und Studierenden gleichermaßen verbessern können.

Aufgabenstellung

Zeitplan



Entwurf (Meilenstein 1)

Anforderungen/Arbeitspakete

Anforderungsanalyse und Konzeptentwicklung

Teilaufgaben:

- Analyse der aktuellen Prozesse und der technischen Anforderungen.
- Festlegung der benötigten Metadaten (Stellentyp, Studiengang, etc.).
- Definition der Sicherheitsanforderungen (Schadcode-Vermeidung beim PDF-Upload).
- Evaluierung von möglichen Erweiterungen (Laufzeitverlängerung für Unternehmen, automatische Metadaten-Vorschläge).

Design und Entwicklung des Web-Formulars zur Stellenanzeigen-Einreichung

Teilaufgaben:

- Gestaltung des Frontends des Formulars (benutzerfreundliches Interface).
- Implementierung der Funktion zum Hochladen von PDF-Dateien.
- Entwicklung eines automatisierten Systems zur Extraktion und Vorschlagsgenerierung von Metadaten auf Basis der hochgeladenen PDF-Datei.

Backend-Entwicklung und Datenpool-Management

Teilaufgaben:

- Erstellung einer Datenbankstruktur zur Speicherung der PDF-Dokumente und Meta-Daten.
- Implementierung von Funktionen zur Verwaltung der Stellenanzeigen (Hinzufügen, Bearbeiten, Löschen).
- Automatisierung des Prozesses zur Entfernung von Stellenanzeigen nach Ablauf der Frist (4 Wochen).

Sicherheitsprüfung und Schadcode-Vermeidung

Teilaufgaben:

- Implementierung eines Virenscanners oder einer Schadcode-Prüfung für hochgeladene Dokumente.
- Integration eines PDF-Analysetools zur zusätzlichen Validierung.
- Durchführung von Penetrationstests und Sicherheitsüberprüfungen.

Qualitätssicherung und Testing

Teilaufgaben:

- Durchführung von Unit-Tests, Integrationstests und Systemtests für alle entwickelten Komponenten.
- Sicherstellung der Funktionsfähigkeit auf unterschiedlichen Endgeräten und Browsern.
- Testen des gesamten Einreichungsprozesses sowie des PDF-Uploads.

Deployment und Rollout des Systems

Teilaufgaben:

- Einrichtung einer Testumgebung.
- Einrichtung finaler Umgebung.

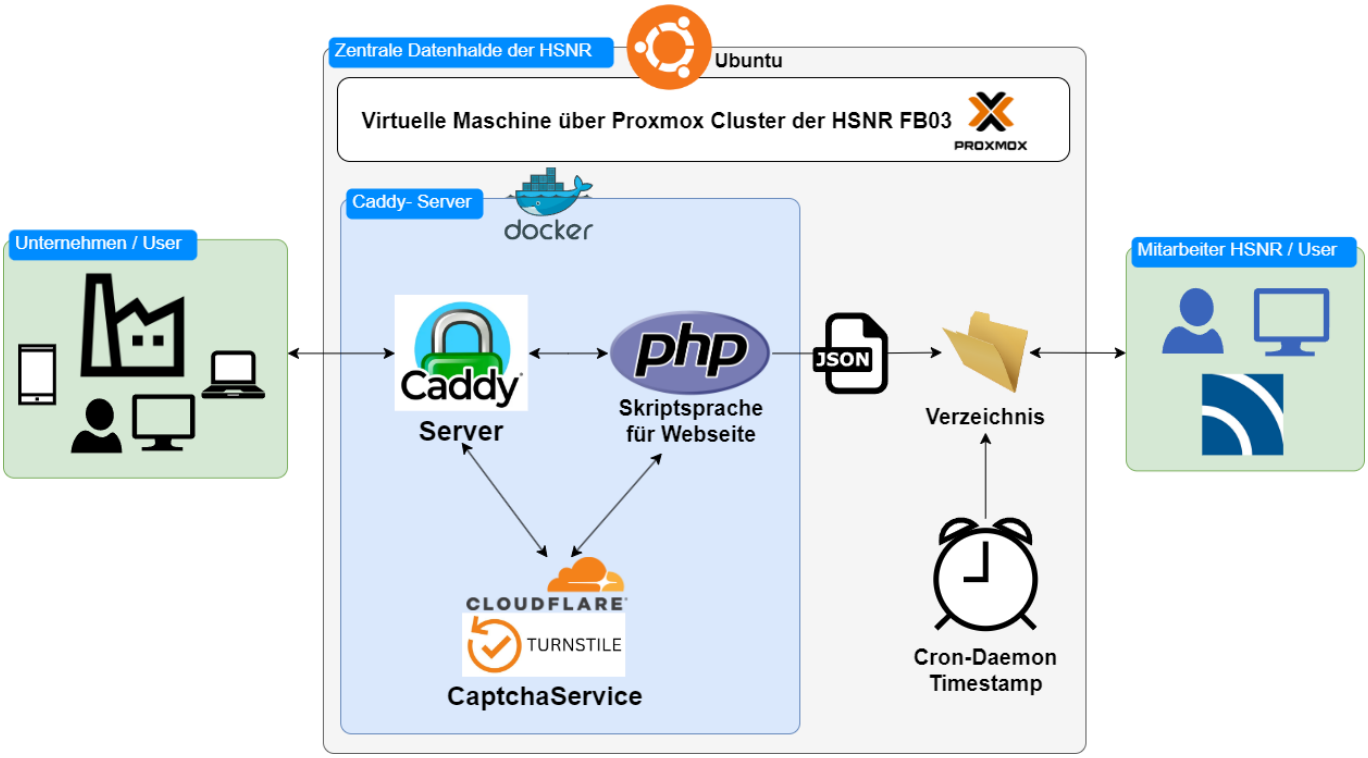
Dokumentation und Support

Teilaufgaben:

- Erstellung einer Benutzeranleitung für Unternehmen zur Nutzung des Einreichsystems.
- Erstellung einer technischen Dokumentation für Administratoren und Entwickler.
- Einrichten der Proxmox-Maschine. Erstellen des Servers und Erreichbarkeit aus dem Internet ermöglichen.
- Erstellen einer Landing-Page für das Hochladen einer eigenen Anzeige.
- Erstellen des Interface zum Dateiupload und Einstellen der Metadaten per Dropdown-Menü und Upload-Button für Datei-Upload.
- Nach Drücken des Upload-Buttons → Prüfen der PDF-Datei nach potenziellen Schadcode.
- Wenn die Prüfung erfolgreich ist, Speichern/Anlegen der PDF-Datei und Metadaten als JSON-Datei auf dem Server.
- Erstellen einer Übersichtsseite mit allen aktuellen Stellenanzeigen.
- Nach erfolgreichem Upload → Weiterleiten auf Übersichtsseite aller Stellenanzeigen.
- Prüfmechanismus integrieren, welcher Stellenanzeigen löscht, die älter als 30 Tage sind.
- Hinweis-Email an Unternehmen kurz vor Ablauf der Anzeige mit Möglichkeit zur Verlängerung der Anzeige.
- Implementieren von Sicherheitsvorrichtungen während des Prozesses.

Systementwurf

Systemarchitektur

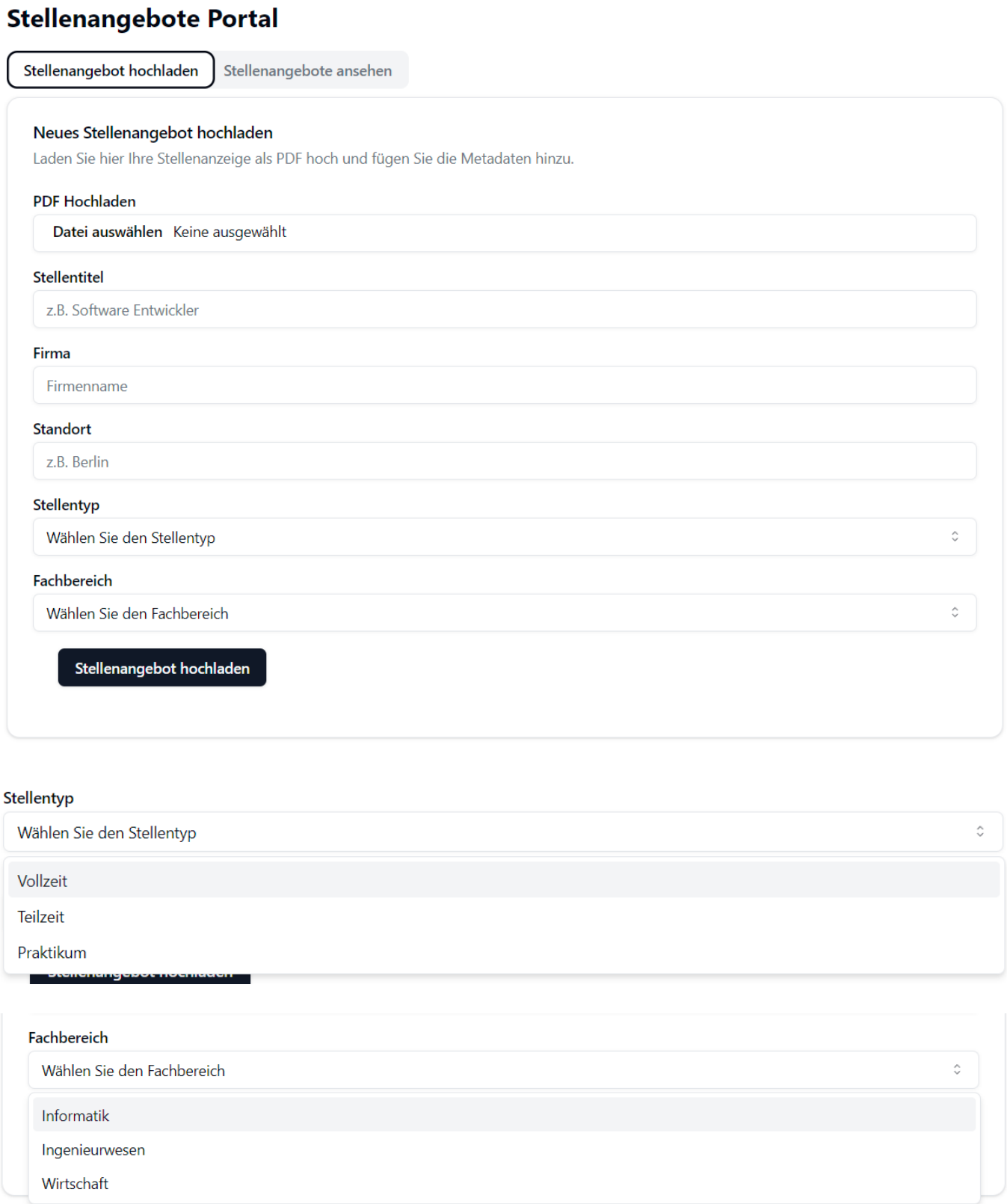


Unternehmen **(User)**, welche eine Stellenanzeige veröffentlichen wollen, rufen auf Ihrem Endgerät im Browser die Webseite des Uploadbereichs auf. (https://108.web.ide3.de/index_merged.php) Die hier adressierte Internetseite läuft auf einem **Caddy Server**, welcher in einem **Docker-Image** in einer virtuellen Maschine bereitgestellt wird. Die virtuelle Maschine hat ein **Ubuntu**-Betriebssystem und läuft über den **Proxmox**-Cluster vom FB03 der Hochschule Niederrhein. Die Internetseite ist mit **PHP** programmiert. Die User laden die Stellenanzeige im PDF-Format hoch und haben über die Benutzeroberfläche die Möglichkeit Metadaten einzustellen. Die Webseite ist so programmiert, dass der Dateityp, die Dateigröße und die Anzahl der Seiten vor dem Upload überprüft werden. Auch die Prüfung auf Schadcode wird voraussichtlich mit PHP stattfinden. Beim Hochladen der Datei durch Klick auf einen Button wird ein CAPTCHA mit Hilfe von **Cloudflare Turnstile** abgefragt, um vor Spam-Angriffen und somit Überladen des Speichers zu schützen. Bei erfolgreichem Upload werden die Datei und META-Daten im **JSON**-Format in einem dafür angelegten **Verzeichnis** auf der virtuellen Maschine gespeichert. Ein auf dem System laufender **Cron-Daemon** überprüft automatisiert in einem täglichen Intervall den Timestamp der Dateien und löscht Stellenanzeigen, die älter als 30 Tage sind, um den Speicherplatz wieder freizugeben. Die entsprechenden Mitarbeiter der HSNR **(User)**, welche für das Hochladen der Stellenanzeigen zuständig sind, greifen auf das Verzeichnis der Datenhalde zu und können diese dort herunterladen, um sie anschließend im Jobportal der Hochschule einzustellen.

Erscheinungsbild der Webseite/Mockup

Die Seite wird sehr simpel gehalten, da sie hauptsächlich eine praktische Funktion erfüllen soll, als optisch ansprechend zu sein. Direkt im oberen Abschnitt wird der Upload-Bereich für die PDF-Datei sein. Erst wenn hier die Bedingungen geprüft und die Datei akzeptiert wurde, soll der Nutzende die Eingabe der Metadaten durchführen. Bei den Metadaten werden neben den individuellen Eingaben über Textfeld von Firmenname, Standort und Stellenbezeichnung auch zwei Dropdown-Menüs mit Vorschlägen für Stellentyp (Jobs, Abschlussarbeiten, Praktika, usw.) und Fachbereich (FB01 - FB10) genutzt. Abgeschlossen wird der Vorgang

mit Klick auf die "Stellenangebot hochladen" Schaltfläche. Nach erfolgreichem Upload bekommt der Nutzer eine entsprechende Bestätigung oder bei Problemen eine Fehlermeldung angezeigt.



Infrastruktur

- **Proxmox-VM:** Eine virtuelle Maschine, die auf dem Proxmox-Node des Fachbereichs 03 läuft, wird für die Bereitstellung des Webservers und der -anwendung verwendet.

- **Docker-Container:** Der Webserver (Caddy) wird in einem Docker-Container bereitgestellt, um Isolation, Portabilität und einfache Verwaltung zu gewährleisten.

Webserver

- **Caddy:** Wird als Webserver eingesetzt, um auf HTTP/HTTPS-Anfragen zu antworten, und bietet automatisches HTTPS, was die Sicherheit erhöht.
- **Container-Setup:** Der Caddy-Container ist so konfiguriert, dass er die HTML-, CSS- und PHP-Dateien der Anwendung bereitstellt.

Webanwendung

- **Frontend:** HTML und CSS für die Benutzeroberfläche, die ein einfaches und responsives Design bietet.
- **Backend:** PHP für die serverseitige Logik, um das Hochladen von Dateien, die Verarbeitung von Metadaten und die Verwaltung des Datenpools zu unterstützen.

Komponenten des Systems

Web-Formular für die Einreichung

- **Eingabefelder:** Felder für die Eingabe von Metadaten (wie Typ der Stelle, Studiengänge, etc.) und ein Upload-Feld für das Hochladen des PDF-Dokuments.
- **Vorschlagsfunktion:** Backend-Logik in PHP zur Analyse des PDF-Dokuments und zur Generierung von Vorschlägen für Metadaten.

PDF-Upload und Sicherheitsüberprüfung

- **Dateiupload:** Implementierung eines sicheren Upload-Mechanismus in PHP, der die Größe und den Typ der hochgeladenen Dateien überprüft.
- **Schadcode-Überprüfung:** Integration einer Bibliothek (z. B. peepdf oder PDFiD) zur Analyse des PDF-Dokuments auf potenzielle Sicherheitsrisiken.

Datenpool und Verwaltung

- **Strukturierte Speicherung:** Die Metadaten als JSON-Datei und die PDF-Dateien werden lokal in einem Ordner auf der VM gespeichert.
- (**Verwaltungsschnittstelle:** Ein Backend-Interface für Administratoren, um eingereichte Stellenanzeigen zu prüfen, zu genehmigen oder abzulehnen.)

Öffentliche Stellenbörse

- (**Anzeige der Stellen:** Ein Frontend-Modul, das die genehmigten Stellenanzeigen aus der Datenbank abruft und darstellt.)
- **Automatisches Entfernen:** Implementierung eines Cron-Jobs oder eines Hintergrundprozesses, der regelmäßig die Datenbank überprüft und Anzeigen nach 4 Wochen entfernt.

Nützliche Features

Benutzerregistrierung und -verwaltung

- Ermöglicht Unternehmen, sich zu registrieren und ihre Anzeigen zu verwalten.

Statistiken und Analysen

- Bereitstellung von Statistiken über eingereichte Anzeigen, Aufrufe und Bewerbungen, um den Unternehmen wertvolle Einblicke zu geben.

Feedback-Funktion

- Möglichkeit für Unternehmen, Feedback zu geben oder Fragen zur Plattform zu stellen.

Responsive Design

- Sicherstellung, dass die Benutzeroberfläche auf verschiedenen Geräten gut aussieht und funktioniert.

Erinnerungsfunktion

- Benachrichtigungssystem: Implementierung einer Funktion, die Unternehmen vor Ablauf der Frist an ihre Stellenanzeige erinnert, möglicherweise über E-Mail-Benachrichtigungen.

Sicherheitsbetrachtungen

- Proxmox wird verwendet, da es von der Hochschule bereitgestellt wird und kostenlos verfügbar ist.

1. Die Webserver-Software Nginx wird weit verbreitet eingesetzt, unter anderem auch an der Hochschule Niederrhein. Als Open-Source-Projekt ist der Quellcode öffentlich einsehbar, was Transparenz und Sicherheit fördert. Regelmäßige Patches schließen bekannt gewordene Sicherheitslücken und sorgen für eine stetige Optimierung der Software. Dank der kostenlosen Verfügbarkeit bietet Nginx zudem eine wirtschaftlich attraktive Lösung.

2. Nach weiterer Überlegung haben wir uns für die Webserver-Software Caddy entschieden, da diese einige bedeutende Vorteile bietet. Hierzu gehört die automatische Konfiguration und Erneuerung von SSL-Zertifikaten von Let's Encrypt, ohne manuelle Eingriffe. Die Caddy-Syntax ist deutlich intuitiver und kürzer als bei Nginx, was die Wartung und Fehlerbehebung erleichtert und die Fehlerquote minimiert. Caddy kommt zudem standardmäßig mit sicheren Voreinstellungen für HTTP/2, TLS und anderen Sicherheitsfeatures. Während Nginx, das HTTP/3-Protokoll nur in der kommerziellen Version anbietet, unterstützt Caddy das Protokoll standardmäßig. Caddy lässt sich, wie auch Nginx problemlos in Docker-Containern betreiben und passt damit gut in moderne Container-Infratstrukturen.

CAPTCHA [9]

Um das System vor Spam zu schützen haben wir uns dazu entschieden, einen CAPTCHA-Dienst in unser System zu implementieren. Dabei haben wir uns für "Cloudflare Turnstile" entschieden. Der Dienst ist kostenlos, ohne Begrenzung von CAPTCHAs, lediglich gibt es eine Begrenzung von Widgets, die verwendet werden dürfen. Außerdem ist die Überprüfung sehr benutzerfreundlich, da die "Challenges" im Hintergrund laufen und der Nutzer somit nichts davon mitbekommt, oder von komplizierten Rätseln aufgehalten wird. Des weiteren werden auch weniger Daten über den Host gespeichert, was positiv für die Privatssphäre der Anwendenden ist.

Alternative Programme waren meist nur recht begrenzt kostenfrei nutzbar, oder bieten den Nutzern weniger Privatssphäre. Zu den Alternativen gehören bspw. reCAPTCHA, Friendly CAPTCHA und hCAPTCHA.

Alternativen:

- reCAPTCHA: weniger gute Privatssphäre
- Friendly CAPTCHA: kein kostenfreier Plan (Wirtschaftlichkeit)
- hCAPTCHA: begrenzte kostenfreie Nutzung

Cloudflare Turnstile

Funktionsweise: "Turnstile" bewertet das Host-System mit einem Scoring-System und schickt dem System transparent, nicht für den Nutzer sichtbar, CAPTCHAs, die gelöst werden müssen. Dabei adaptiert es die Komplexität der "Challenges" je nach dem Score des Nutzers, wie wahrscheinlich es sich um einen Bot handelt. Der Score wird aus Aspekten wie Reputation der IP-Adresse, Support von Web-APIs und menschlichen Verhaltensmustern, berechnet.

Vorteile:

- Keine Begrenzung von Anfragen (auf 10 Widgets begrenzt)
- Bessere Privatssphäre
- Benutzerfreundlichkeit (läuft transparent im Hintergrund, ohne Rätsel lösen für die Nutzer)

Sicherheitsüberprüfung der PDF-Dateien

Im Fokus für diesen Entwurf waren die Tools PDFiD und peepdf.

PDFiD [4] PDFiD ist ein reines PDF-Analyse-Werkzeug, was PDFs auf ihre Merkmale und Objekte analysiert und diese auflistet. So besteht hier vorerst keine Sicherheitsbetrachtung, da es sich nur um eine Auflistung der PDF-Struktur und -Elemente handelt. Es kann jedoch zur Sicherheitsüberprüfung von Dokumenten genutzt werden, da es einen klaren und ganzheitlichen Überblick über die Inhalte der PDF bietet, selbst wenn diese verschleiert wurden.

peepdf [5] peepdf ist ein Python-Tool zur Analyse und Auswertung von PDFs im Hinblick auf Security. Genau wie PDFiD bietet es eine Auflistung über die in einer PDF vorkommenden Merkmale und Objekte. Diese Funktionalität wird jedoch erweitert, indem peepdf die Analyse von einzelnen Datenstrukturen wie Streams und JavaScript ermöglicht. Darüberhinaus können mit dem Tool auch PDF-Dateien erstellt und modifiziert werden, mit der möglichen Einbindung von JavaScript und der Obfuskation von Inhalten.

Bewertung Für unser Projekt ist peepdf PDFiD überlegen, da es über die Funktion zur Analyse der PDF-Struktur hinaus eine tiefere Analyse mit Fokus auf Sicherheitsaspekte anbietet. Jedoch gibt es auch bei der Umsetzung von peepdf eine gewisse Hürde. Der erste Schritt in beiden Programmen stellt die Analyse der Struktur da, woraufhin man über peepdf weitere Merkmale inspizieren kann. Für die weitere Analyse müssen jedoch neue Commands eingegeben werden, die je nach Strukturausgabe unterschiedlich sind. So unterscheidet sich die Analyse einer Annotation von der eines Streams. Zur Analyse von PDFs mit potenziell schädlichen Inhalten ist ein Verständnis der Struktur erforderlich, das im Rahmen dieses Projekts nicht automatisiert werden kann.

Mögliche Lösung Es stellt sich die Frage, ob der Scanner, der die Anhänge von E-Mails an Hochschul-Adressen automatisch analysiert, weiterhin von Nutzen sein könnte. Alternativ zu einer internen Lösung über

peepdf könnten die Eingaben der Stellenanzeigenanbieter oder zumindest die eingereichten PDF-Dateien an ein E-Mail-Postfach gesendet werden. Dadurch werden sie automatisch wie zuvor auf schädliche Inhalte überprüft, müssen jedoch von den ankommenden E-Mails auf die Datenhalde transferiert werden. Für eine weitere Sicherstellung werden eine weitere Anzahl an Tools auf ihre Funktionalität, ihre Operation innerhalb unseres Projektes und ihre Wirtschaftlichkeit analysiert. Darunter QuickSand [6], PDFExaminer Tool [3] und verschiedene Scan-APIs wie von Cloudmersive [1], VirusTotal [7] und IPQS [2].

Testplan

Bedeutung des Testplans

- Die Entwicklung des Stellenbörse-Systems erfordert einen einen durchdachten Testansatz, der verschiedene Aspekte der Anwendung berücksicht. Anders als bei einfachen Webanwendungen geht es hier um ein System, welches zusätzlich sensible Daten verarbeitet und eine hohe Zuverlässigkeit erfordert. Fehler könnten nicht nur zu Frustration bei den Nutzern führen, sondern auch Sicherheitsrisiken bergen.
- Unit-Tests bilden das Fundament der Teststrategie, da insbesondere bei der Verarbeitung von PDF-Dateien und der Metadatenextraktion Fehler den gesamten Prozess beeinträchtigen können. Die Integrationstests stellen sicher, dass die Komponenten vom Upload bis zur automatischen Archivierung funktionieren.
- Die Sicherheitstests haben einen hohen Stellenwert, da PDF-Dateien potenzielle Risiken bergen können, wie die Ausführung von JavaScript-Code (Schadcode ausführen), eingebettete Objekte beinhalten (.exe-Dateien ausführen), automatisches Aufrufen von URLs, XML-basierte Angriffe und weitere, daher ist die Absicherung essentiell. Gleichzeitig müssen Performance-Tests die Stabilität unter Last gewährleisten
- Usability-Tests stellen sicher, dass das System intuitiv bedienbar ist.

Unit Tests

- Bei der Verarbeitung von PDF-Dateien und Metadaten müssen wir sicher sein, dass jede einzelne Komponente zuverlässig funktioniert. PHP-Komponenten wie die PDF-Verarbeitung, die korrekte Speicherung der ausgewählten Metadaten und das JSON-basierte Speichersystem werden in diesem Schritt geprüft. Besonders wichtig ist die Validierung der Upload-Funktion und der korrekten Übernahme der vom Unternehmen ausgewählten Kategorisierungen der Stellenanzeigen. In der Praxis bedeutet das, dass wir für Funktionen wie die PDF-Größenprüfung oder die Metadatenextraktion einzelne Testfälle anwenden. Diese Tests stellen sicher, dass jede Komponente auch isoliert zuverlässig arbeitet. Die Tests sind schnell angewendet und geben uns sofort Feedback, damit wir Fehler korrigieren können.

Integrationstests

- Der nächste Schritt bezieht sich auf die Integrationstests. Die Integrationstests überprüfen das Zusammenspiel des Caddy-Webservers im Container mit der PHP-Anwendung. Der gesamte Prozess vom Upload einer Stellenanzeige, der Auswahl der Metadaten über die vordefinierten Dropdown-Menüs bis zur automatischen Löschung nach vier Wochen muss reibungslos funktionieren. Dabei wird besonderes Augenmerk auf die korrekte Funktionsfähigkeit gelegt.

Sicherheitstests

- Die Sicherheitstest haben durch das potenzielle Hochladen von schädlichen Inhalten und diversen webbasierten Angriffen einen hohen Stellenwert in dieser Arbeit. Die implementierten Schutzmaßnahmen wie Größenlimitierung, Mime-Type-Überprüfung und Schadcode-Analyse müssen gründlich getestet werden. Auch die automatische HTTPS-Konfiguration des Caddy-Servers wird auf ihre Wirksamkeit überprüft.

Testplan Checkliste

Prüfung	**Ja**	**Nein**	**Erwartetes Ergebnis**
Funktionalität			
Ist die Website erreichbar?			Ja
Funktioniert das Hochladen der PDF-Datei problemlos (mit Auswahl von Kategorien)?			Ja
Werden die Metadaten für die PDF-Dateien automatisch und korrekt generiert?			Ja
Kann auf die hochgeladenen PDF-Dateien zugegriffen werden?			Ja
Funktioniert die automatische Löschung nach 28 Tagen?			Ja
Funktioniert das automatische Backup-System?			Ja
Sicherheitsaspekte			Ja
Funktioniert die PDF-Größenbegrenzung? (Max.5MB)			Ja
Können ausschließlich PDF-Dateien hochgeladen werden?			Ja
Werden schädliche PDF-Dateien erkannt?			Ja
Lädt und funktioniert das Captcha?			Ja
Werden neue PDF-Dateien umbenannt, falls sie bereits mit gleichem Namen existieren?			Ja
Ist die Seite vor XSS-Angriffen geschützt?			Ja

Alle Punkte sollten bei einem korrektem aufsetzen der Website mit "JA" beantwortet werden können. Sollte dies nicht der Fall sein, so funktioniert das System nicht wie vorhergesehen.

(Definition: Ein Testplan umfasst Ziele, Zeitplan, Ergebnisse und Ressourcen, die zum Erreichen dieser Ziele erforderlich sind.)

Sicherheitsanforderungen

Bedrohungsmodellierung

STRIDE-Bedrohungsmodell

Das Stride-Bedrohungsmodell ist ein Modell zur Kategoriesierung von IT-Sicherheitsirisiken, welche wir nutzen, um potenzielle Schwachstellen in unserem System systematisch zu identifizieren

- Es eignet sich für kleine bis mittlere Projekte und Teams und wurde als gute Wahl aufgrund der Zugänglichkeit und Praktibilität empfunden. Zusätzlich fokussiert sich STRIDE stark auf Webanwendungen.

Identitätsvortäuschung

- Bedrohungen:
  - Vortäuschung echter Unternehmen
  - Einstellung gefälschter Stellenanzeigen
- Gegenmaßnahmen
  - Implementierung von Cloudflare Turnstile CAPTCHA (wird Implementiert)
  - Zugriffsbeschränkung pro IP-Adresse (Möglichkeit weiterer Gegenmaßnahmen)
  - E-Mail-Verifizierung für Unternehmenseinreichungen (Möglichkeit weiterer Gegenmaßnahmen)
  - Domänenvalidierung für Unternehmens-E-Mails (Möglichkeit weiterer Gegenmaßnahmen)

Manipulation

- Bedrohungen:
  - Modifikation hochgeladener PDF-Dateien
  - Manipulation der Metadaten
  - Unbefugte Änderung gespeicherter Stellenanzeigen
- Gegenmaßnahmen (Möglichkeiten von Gegenmaßnahmen)
  - Dateiintegritätsprüfung mittels Hashwerten
  - Sichere Dateispeicherung mit entsprechenden Berechtigungen
  - Eingabevalidierung für alle Formularfelder

Nicht-Abstreitbarkeit

- Bedrohungen
  - Abstreiten von Stellenanzeigeeinreichungen
  - Behauptungen unbefugter Änderungen
- Gegenmaßnahmen (Möglichkeiten von Gegenmaßnahmen)
  - Umfassendes Protokollierungssystem
  - E-Mail-Bestätigung für alle Einreichungen

Informationspreisgabe

- Bedrohungen
  - Unbefugter Zugriff auf gespeicherte PDF-Dateien
  - Datenlecks
- Gegenmaßnahmen
  - Strikte Zugriffskontrollen
  - Sichere Dateipfade

    - Eingeschränkte Fehlermeldungen

Denial of Service

- Bedrohungen
    - Überlastung des Upload-Systems
    - Erschöpfung des Speicherplatzes
    - Serverressourcenverbrauch
- Gegenmaßnahmen
    - Dateigrößenbeschränkung (5MB)
    - Upload-Ratenbegrenzung
    - Ressourcenüberwachung
    - Cloudflare DDoS-Schutz

Rechteerweiterung

- Bedrohungen
    - Unbefugter Administratorzugriff
    - Systembefehlsausführung durch PDFs
- Gegenmaßnahmen
    - Rollenbasierte Zugriffskontrolle
    - Eingabevalidierung
    - Principle of Least Privilige

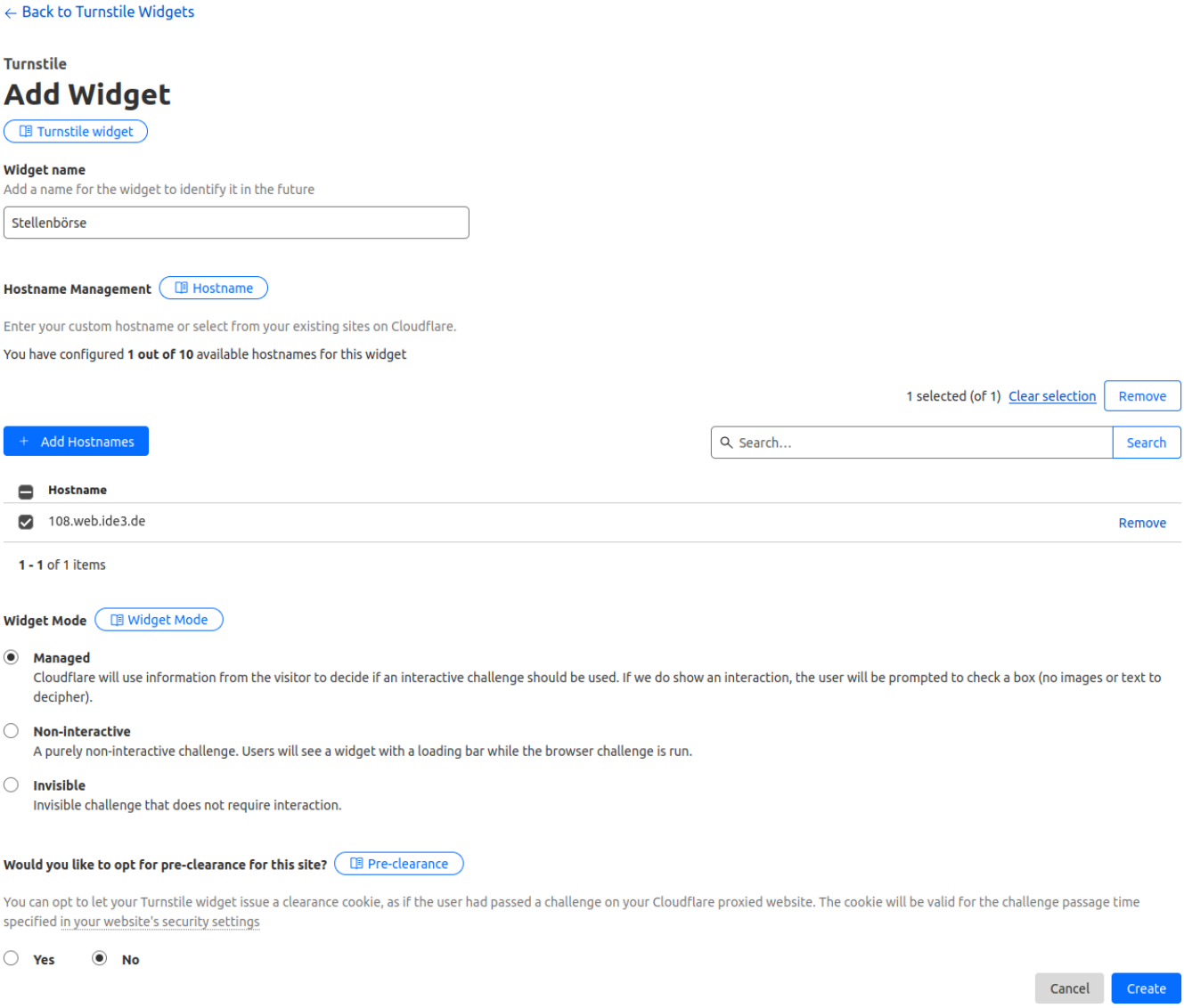
Realisierung (Meilenstein 2)

Arbeitsumgebung

Die Arbeitsumgebung ist, wie unter dem Punkt "Systemarchitektur" des "Meilenstein 1" erläutert, realisiert worden. Im Folgenden wird die Realisierung der einzelnen Aspekte genauer erörtert.

Implementierung "Cloudflare Turnstile" [8]

Konfigurieren und Erstellen eines Turnstile-Widgets als Captcha:



Nun ist der Sitekey und Secret-Key für das Widget einsehbar.

Widget in Webseite einbinden

1. Folgendes Skript wurde in HTMLs "head"-Element eingefügt:

```
<script src="https://challenges.cloudflare.com/turnstile/v0/api.js" async defer>
</script>
```

2. Widget an die gewünschte Stelle einbinden

```
<div id="cfCaptcha" class="cf-turnstile" data-sitekey="0x4AAAAAAAzudMkcbeUQtbl5">
</div>
```

3. Rendern der Integration auf Client-Seite

Implizites Rendering: Wir haben uns für das implizite Rendering entschieden, da wir eine statische Webseite verwenden und das Widget nicht nur zu gewünschten Zeitpunkten gerendert werden soll.

Validierung der serverseitigen Response

Turnstile Tokens müssen mithilfe von siteverify verifiziert werden, um zu überpüfen, dass dieser noch nicht verwendet wurde und noch valide ist.

Dem siteverify endpoint muss die Response und der Secret-Key zusammen mit dem Sitekey übergeben werden.

Mithilfe der Funktion `"checkCaptcha():bool` in der Datei index.php führen wir die Validierung durch und erhalten die Antwort ob der Token verifiziert wurde.

Zurücksetzen des Widgets bei Inaktivität

```
<script>
    // Function to execute every 30 seconds
    function resetTurnstile() {
        if (typeof turnstile !== 'undefined') {
            turnstile.reset(cfCaptcha);
        }
    }
    setInterval(resetTurnstile, 30000);
</script>
```

Automatisierte Löschung

In der Datenhalde abgelegte PDF-Dateien, welche älter als 30 Tage sind, sollen automatisiert gelöscht und der Speicher somit bereinigt werden. Hierzu ist die regelmäßige Prüfung der Timestamps der Dateien nötig. Dies lässt sich durch das Erstellen eines Skripts in Kombination mit dem Cron-Daemon bewerkstelligen. Cron ist ein Dienst welcher automatisiert wiederkehrende Aufgaben erledigen kann und ist in Ubuntu in der Regel bereits vorinstalliert (`cron` und `anacron`) [10].

Um zu überprüfen ob der Cron-Dienst ausgeführt wird nutzen wir folgenden Konsolenbefehl: `systemctl status cron` Wir erhalten jedoch folgende Fehlermeldung *Unit cron.service could not be found*. Dieselbe Meldung erhalten wir bei Überprüfung des Anacron-Dienst per `systemctl status anacron`.

Die beiden Dienste sind im gewählten Ubuntu Image nicht installiert. Die Installationen lassen sich durch folgende Befehle durchführen: `sudo apt install cron sudo apt install anacron`

Sollten die Dienste nach der Installation nicht automatisch starten, ist dies manuell durchzuführen: `sudo systemctl start cron sudo systemctl start anacron`

Nach erneutem Prüfen bekommen wir eine positive Rückmeldung vom System, dass die Dienste wie gewünscht ausgeführt werden. Durchgeführt wird die automatische Prüfung/Löschung immer morgens um 6:25 Systemzeit.

Skript für Datenlöschung

Das folgende Skript trägt den Namen "*datenbereinigung*".

```
#!/bin/bash
Speicher="/home/user/caddy_test/public/uploads"
find "$Speicher" -iname "*.pdf" -type f -mtime +30 -delete
```

`Speicher` dient hier als Variable für das Verzeichnis, unter dem die hochgeladenen Bewerbungen im PDF-Format liegen.

`find` erhält durch die zuvor definierte `Speicher`-Variable das benötigte Verzeichnis in welchem es suchen soll. `-iname "*.pdf"` fungiert als Filter und prüft case-insensitiv nur Dateien mit Endung .pdf. `-type f` beschränkt die Suche auf Dateien und ignoriert somit Verzeichnisse. `-mtime +30` wählt nur Dateien aus, welche älter (+) als 30 Tage (time) sind. `-delete` löscht alle durch den vorherigen Suchfilter gefundenen Dateien.

Der Zeitraum der Löschung kann durch einfaches Anpassen des Parameters `-m` geändert werden. Um die Prüfung des Skripts auf Funktionalität zu vereinfachen, werden weitere Zeitstempel dokumentiert:

`-mmin +1` Löscht Dateien, welche älter als eine Minute sind. `-mhour +1` Löscht Dateien, welche älter als eine Stunde sind. `-mtime +1` Löscht Dateien, welche älter als einen Tag sind. `-mweek +1` Löscht Dateien, welche älter als eine Woche sind.

Das erstellte Skript prüft zusammengefasst im Speicherverzeichnis der Datenhalde alle Dokumente auf ihren Zeitstempel. Sollten Dokumente älter als 30 Tage sein, werden diese gelöscht. Eine tägliche Prüfung reicht für die Bereinigung der Datenhalde aus. Hierfür existiert ein eigenes Verzeichnis für tägliche automatisierte Aufgaben. `/etc/cron.daily/` Das Skript wird durch Speichern im entsprechenden Verzeichnis vom Cron-Dienst ausgeführt und erledigt somit nun automatisiert seine Aufgabe. Um von Cron ausgeführt zu werden, muss dem Skript zuvor noch die benötigte Berechtigung mit `sudo chmod +x datenbereinigung` zugeteilt werden.

Testen des Skripts auf Funktion

Es werden Testdateien (PDF) im Speicherverzeichnis der Datenhalde abgelegt. Das Skript wird umgeschrieben und soll die Löschung bei Dateien durchführen, welche älter als eine Minute sind(`-mmin +1`). Die täglichen Cron-Skripte können durch den Befehl `run-parts /etc/cron.daily` manuell ausgeführt werden. Die zuvor abgelegten Dateien wurden nach manueller Ausführung des Skripts erfolgreich gelöscht. Auch die Automatisierung durch Cron wurde durch Ablegen neuer Testdateien bestätigt, da diese am folgendem Tag nach 6:25 aus dem Verzeichnis erfolgreich gelöscht wurden.

Textfelder

```
<div class="form-group">
    <label for="firmenname">Firmenname:</label>
    <input type="text" name="firmenname" id="firmenname" required>
</div>
```

- Die Textfelder werden als `<input type="text">` definiert.
- Jedes Feld hat einen `name` (für die PHP-Verarbeitung) und eine `id`.
- Das `required`-Attribut macht das Feld zu einem Pflichtfeld.
- Die Werte werden später in PHP über `$_POST['firmenname']` ausgelesen.

Dropdown-Menüs

```
// PHP-Arrays für die Dropdown-Optionen
$stellentypen = ["Jobs", "Praxisphasen & Praktika", "Abschlussarbeiten", ...];

// HTML-Implementierung
<select name="stellentyp" id="stellentyp" required>
    <?php foreach ($stellentypen as $typ): ?>
        <option value="<?php echo htmlspecialchars($typ); ?>">
            <?php echo htmlspecialchars($typ); ?>
        </option>
    <?php endforeach; ?>
</select>
```

- Die Auswahl im Dropdown Menü wird aus PHP-Arrays gebildet.
- Die foreach-Schleife erstellt für jeden Array-Eintrag eine Auswahlmöglichkeit.
- `htmlspecialchars()` verhindert XSS-Angriffe, indem spezielle Zeichen, die für einen Angriff genutzt werden können, in HTML-Entities umgewandelt werden. Beispielsweise `<` wird zu `&lt;`, folglich wird aus `<script>böser_code()");</script>' = &lt;script&gt;böser_code()&lt;/script&gt;`, somit wird der Code nur als Text angezeigt und ist harmlos.

Button

```
<button type="submit" <?php echo $is_button_enabled ? '' : 'disabled'; ?
>>Hochladen</button>
```

- Submit-Button zum Absenden des Formulars.
- PHP-Variable `$is_button_enabled` legt fest, ob der Button aktiv ist.

Datei-Upload

```
// HTML-Teil
<input type="file" name="pdf_file" id="pdf_file" accept=".pdf" required>

// PHP-Verarbeitung
if (isset($_FILES['pdf_file'])) {
    $file = $_FILES['pdf_file'];
    if ($file['size'] > MAX_FILE_SIZE) {
        $message = '<div class="error">Die Datei ist zu groß!</div>';
    } else {
        $pdf_filename = uniqid() . '_' . basename($file['name']);
```

```
        $upload_path = UPLOAD_DIR . $pdf_filename;

        if (move_uploaded_file($file['tmp_name'], $upload_path)) {
            // Upload erfolgreich
        }
    }
}
```

- `<input type="file">` ermöglicht die Dateiauswahl.
- `accept=".pdf"` beschränkt den Upload auf PDF-Dateien.
- `$_FILES` enthält Upload-Informationen.
- `move_uploaded_file()` verschiebt die Datei in den Upload-Ordner.

Formularverarbeitung

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Datenvalidierung
    $formData = [
        'firmenname' => filter_var($_POST['firmenname'], FILTER_SANITIZE_STRING),
        // weitere Felder...
    ];

    // Speichern in JSON-Datei
    if (saveToDatahalde($formData)) {
        $message = '<div class="success">Erfolgreich gespeichert!</div>';
    }
}
```

- Prüft ob das Formular abgeschickt wurde (POST).
- Validiert und bereinigt Eingabedaten.
- Speichert Daten in der JSON-Datei.
- Gibt Erfolgsmeldung oder Fehlermeldung aus.

Zusammenspiel der Funktionalitäten

- Zu Beginn füllt der Benutzer das Formular aus und wählt die gewünschte PDF-Datei.
- Nach dem Klicken auf den "Hochladen"-Button werden die Daten per `POST` gesendet.
- PHP validiert dabei alle Eingaben, die PDF-Datei wird hochgeladen und mit der entsprechenden Funktion geprüft.
- Alle Daten werden als JSON-Datei in der "Datenhalde" gespeichert.
- Zuletzt wird eine Erfolgs- oder Fehlermeldung angezeigt.

PDF-Sicherheitsüberprüfung

Problematik

Bei dem Vergleich von fünf kostenfreien Tools und drei APIs stellte sich heraus, dass es keine für uns passende Lösung gibt, die schädliche PDFs erkennt, unschädliche nicht als schädlich markiert und dem Datenschutz

folgt. Folglich eine kurze Analyse der verschiedenen Tools, die im Umfang dieses Projektes untersucht wurden. Ein Fokus wird darauf gelegt, wie gut sie für dieses Projekt einzusetzen sind.

Vorgehen

Zur Überprüfung der Tools wurden 16 verschiedene PDF-Dateien genutzt. Fünf davon waren schädliche Dateien, die wir im Rahmen eines anderen Moduls (BCSM 404 Sichere Anwendungssysteme) zur Untersuchung bearbeitet hatten. Zusätzlich haben wir eine eigene schädliche PDF zum Testen verwendet. Die Angriffsarten:

- Zwei PDFs mit Code Execution
- Zwei PDFs mit URI Invocation
- Eine PDF mit XML External Entity Attack
- Eine PDF mit Deflate Bomb

Zur Überprüfung von normalen PDFs ohne schädliche Inhalte wurden zehn zufällig gewählte Stellenausschreibungen von der Hochschulwebseite [13] verwendet.

Diese 16 PDFs wurden dann von den verschiedenen Tools überprüft und die Resultate in einer Tabelle aufgeführt. Die Ergebnisse dazu sind in den einzelnen Abschnitten zu finden.

peepdf-3 [5]

peepdf-3 von digitalsleuth ist eine aktualisierte Version von peepdf [11], welches Jose Miguel Esparza (jesparza) 2011 veröffentlich hat. Es ist ein PDF-Analysetool, welches in Python 3 geschrieben wurde. Bei der Analyse von PDF-Dateien gibt es alle Elemente aus, die potenziell schädlich sein können. Dabei findet jedoch keine genauere Analyse statt, ob die Elemente wirklich schädlich sind.

Stärken:

- Erkennt 4 von 6 schädlichen PDFs.
    - Erkannt: Code Execution 1 & 2, URI Invocation 1 & 2
    - Nicht erkannt: Deflate Bomb (Error), XML External Entity Attack
- Gibt potenziell schädliche Elemente aus.

Schwächen:

- Führt bei Untersuchung der Deflate Bomb zu einer Ressourcenauslastung von 100%.
- Bei 2 von 16 untersuchten PDFs entstehen Errors.
- Keine Analyse, ob die gefundenen Elemente wirklich schädlich sind.
- 2 von 10 normalen PDFs werden als potenziell schädlich wahrgenommen.

Obwohl peepdf-3 eine Großzahl an potenziell schädlichen Elementen in den Dateien erkennt, ist die False Positive und Error-Rate am höchsten aus den überprüften Tools. Die hohe Ressourcenauslastung beim Untersuchen der Deflate Bomb ist nicht optimal, jedoch kann dieses Problem leicht umgangen werden, indem eine Zeitbegrenzung eingeführt wird nach der die Untersuchung unterbrochen und die Datei als nicht brauchbar eingestuft wird.

PDFiD [4]

PDFiD ist eins von vier PDF-Bearbeitungs- und Analysetools von Didier Stevens, die seit mindestens 2008 existieren. Es analysiert PDFs und gibt alle in ihnen vorkommenen Elemente aus, jedoch ohne eine Bewertung im Sinne der Sicherheit. Aus diesem Grund findet es im Rahmen dieses Projekts keinen Gebrauch mehr. Es würde sich jedoch für diejenigen anbieten, die einen Parser brauchen, auf dem externe Sicherheitsanalysesoftware angewendet wird.

PDFExaminer

PDFExaminer ist ein PHP PDF-Analyse Tool von tylabs aus dem Jahr 2021, welches die verschiedenen Elemente von PDFs analysiert, diese nach Sicherheit bewertet und am Ende in der Variable `is_malware` eine vollständige Bewertung der analysierten Datei ausgibt. Das Tool wurde von QuickSand abgelöst, welches jedoch ein Pythontool ist.

Stärken:

- 0 von 10 normalen PDFs werden als schädlich erkannt.
- Gibt klar aus, ob es sich um Malware handelt (`is_malware`).

Schwächen:

- Erkennt 2 von 6 schädlichen PDFs.
    - Erkannt: Code Execution 1 & 2
    - Nicht erkannt: Deflate Bomb, URI Invocation 1 & 2, XML External Entity Attack

Das Tool bietet für dieses Projekt die richtige Funktionalität an, vor allem durch die Variable `is_malware`. Aufgrund der niedrigen Anzahl der True Positives lohnt es sich jedoch nicht zur Analyse von schädlichen Inhalten, die über simple Code Executions hinausgehen.

QuickSand

Die Weiterführung von PDFExaminer, nun jedoch in Python geschrieben, ist QuickSand. Dieses Tool analyisiert die einzelnen Elemente von PDF-Dateien und bewertet diese nach Schädlichkeit. Zuletzt wird eine Einstufung der Gefährdung durch die Datei über die Variable `warning:` ausgegeben. Diese Variable zeigt von 1 bis 10 an, wie gefährlich die Datei ist.

Stärken:

- 1 von 10 normalen PDFs wird als schädlich erkannt.
- Gibt eine Gewichtung der Gefährdung durch die Datei aus.

Schwächen:

- Erkennt 2 von 6 schädlichen PDFs.
    - Erkannt: Code Execution 1, URI Invocation 1

Auch das neue Analysetool von tylabs erfüllt alle theoretischen Anforderungen des Projekts. Eine klare Einstufung, wie gefährlich eine Datei ist, bietet eine Anpassung an die Risikobereitschaft beziehungsweise erlaubt eine Lockerung der Regeln, falls das Programm schnell eine niedrige Gefährdung feststellt. Da jedoch zu viele schädliche Dateien unentdeckt bleiben, ändert auch das überzeugende Featureset nichts an der Nutzbarkeit.

Simple-PDF-Analyzer [12]

Der Simple-PDF-Analyzer (kurz SPA) wurde 2021 von 0xE0L veröffentlicht, wertet PDFs aus und gibt für die Sicherheitsanalyse interessante Elemente aus. Weiterhin kann SPA FlateDecode Streams lesbar machen. Es findet keine Bewertung hinsichtlich der Sicherheit statt, weswegen SPA, wie PDFiD, für dieses Projekt keinen Gebrauch finden wird.

APIs

Die von uns analysierten APIs weisen mehrere für dieses Projekt bedeutende Probleme auf. Zum einen ist der Datenschutz nicht unbedingt gewährleistet, was im Falle der PDFs, die später auf dem Jobportal der Hochschule veröffentlicht werden sollen, jedoch nicht allzu gravierend ist. Dazu kommen sie häufig mit einer Begrenzung an Dateien, die in einem bestimmten Zeitraum hochgeladen werden können. Um diese Limits zu erhöhen, ist ein Abonnement nötig. Da im Rahmen dieses Projekts keine monetären Ausgaben getätigt werden sollen, stellen diese APIs keine Alternative zu den Analysetools dar.

Hochschul E-Mail-Server

Auch ein Versenden der PDF-Dateien an eine E-Mail-Adresse, die über die Hochschule gehostet wird, führt zu keiner Erkennung der schädlichen PDFs. Somit findet entweder keine Überprüfung der Dateien statt, oder sie ist ungenauer als die der genutzten Tools und stellt keine Alternative dar. Wäre eine solche Überprüfung vorhanden und würde diese zuverlässige Ergebnisse liefern, würden sich jedoch neue Hürden stellen. Die Dateien müssten automatisiert über den Mail-Server hochgeladen und verschickt werden und müssten bei der Empfänger-Adresse automatisiert heruntergeladen und zur Verfügung gestellt werden.

Fazit

Zwei der fünf Tools (PDFiD und Simple-PDF-Analyzer) dienen hauptsächlich als PDF-Parser, wodurch ein Programm geschrieben werden müsste, welches die analysierten Elemente genauer untersucht und mit bekanntem PDF-Schadcode vergleicht. Das liegt jedoch außerhalb des Rahmen dieses Projekts. Von den drei übrigen haben zwei (PDFExaminer und QuickSand) eine Erkennungsrate von 33% aller getesteten schädlichen Dateien, was zu niedrig ist. Die letzte Alternative (peepdf) erkennt schädliche PDFs am zuverlässigsten, jedoch auch nicht mit einer hohen Präzision. Zudem erzeugt peepdf die meisten False Positives und gibt die meisten Errors aus. Weder die getesteten APIs noch eine Weiterleitung der Dateien über den Hochschul E-Mail-Server sind eine Möglichkeit.

Da aktuell keine Lösung die Anforderungen erfüllt, gibt es drei Wege, fortzufahren: Es könnte eine der kostenpflichtigen APIs getestet und bei erfolgreichem Test genutzt werden. Alternativ könnte ein eigenes Tool entwickelt werden, das entweder eigenständig oder mithilfe eines existierenden Parsers Elemente mit bekanntem Schadcode vergleicht. Schließlich könnte auch ohne eine Sicherheitsüberprüfung fortgefahren werden.

Test

Systemvoraussetzungen & Vorbereitung

- Der Tester muss mehrere aktuelle Browserversionen installieren, um eine umfassende Prüfung sicherzustellen.
- Folgende Testdateien werden vorbereitet:

    - Eine Standard-PDF mit 1-2 Seiten für Grundfunktionen unter 5MB
    - Eine PDF-Datei über 5 MB zur Überprüfung des Upload-Limits
    - Eine PDF mit Sonderzeichen im Dateinamen
    - Mehrere potenziell schädliche PDFs zur Sicherheitsüberprüfung (Deflate Bomb, Malicious Scripts, etc.)
- Sollte das erwartete Ergebniss nicht vollständig erfüllt werden, so wird dies notiert und dokumentiert. Zusätzlich wird in der Checkliste in der jeweiligen Kategorie "NEIN" angekreuzt.

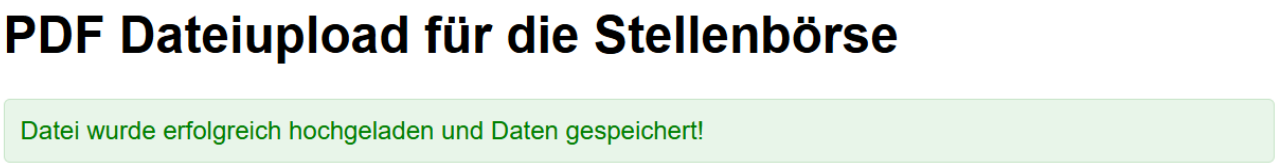
Detaillierte Testszenarien Funktionalität

Erreichbarkeit und Grundfunktionen

- Der Tester ruft die Webseite (https://108.web.ide3.de/index_merged.php) in verschiedenen Browsern (z. B. Mozilla Firefox, Google Chrome, Microsoft Edge, etc.) auf und überprüft, ob die Seite vollständig und korrekt lädt.
- Die Erreichbarkeit wird auf Desktop-PCs und mobilen Geräten getestet.
- Erwartetes Ergebnis: Die Seite ist auf dem PC, sowie auf dem Mobilgerät erreichbar. Alle Elemente laden korrekt und sind auf Ihren vorhergesehenen Positionen (siehe Bild). Das Ergebnis wird dokumentiert.

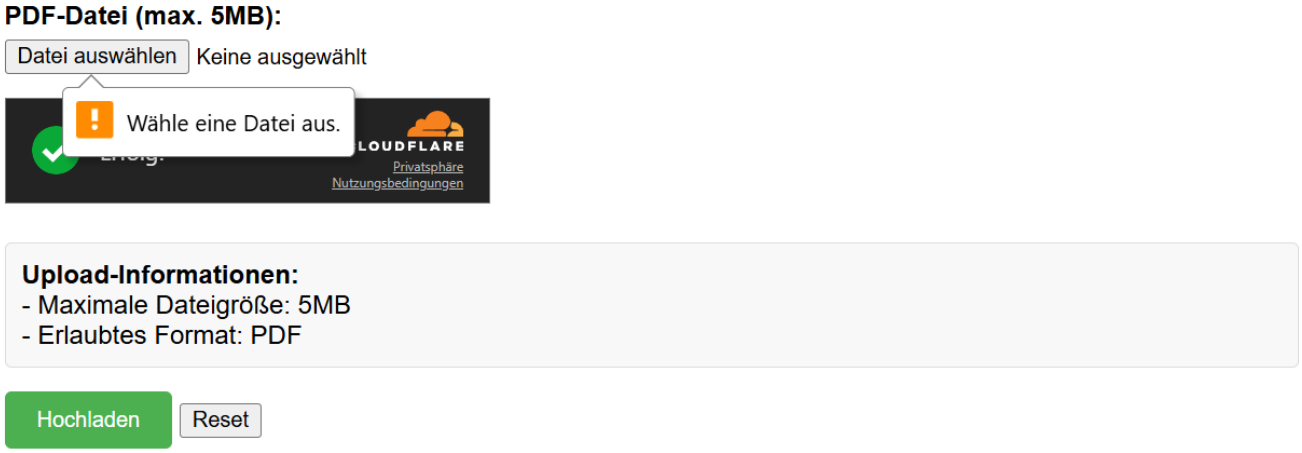
PDF-Upload-Prozess

- Alle verfügbaren Kategorien werden systematisch durchgeklickt.
- Der Upload wird mit verschiedenen Kategorien getestet.
- Pflichtfelder werden auf Vollständigkeit und Funktionalität geprüft.
- Erwartete Ergebnisse: Jede Kategorie lässt sich auswählen. Der Upload funktioniert unabhängig der gewählten Kategorien. Bei erfolgreichem Upload wird die Meldung "Datei wurde erfolgreich hochgeladen und Daten gespeichert" in grüner Schrift angezeigt.



Bei nicht ausgefüllten Pflichtfeldern wird kein Upload durchgeführt. Stattdessen wird der Nutzer darauf hingewiesen, die Pflichtfelder auszufüllen.

Bei fehlender Datei wird der Warntext "Wähle eine Datei aus" angegeben.



Die PDF sollte nach Upload im Dateiserver sichtbar sein.

- Das Ergebnis wird dokumentiert.

Metadaten-Generierung

- Nach jedem erfolgreichem Upload werden die automatisch generierten Metadaten auf Korrektheit und Vollständigkeit überprüft.
- Erwarteter Wert: Die Metadaten werden als JSON-Datei im Dateiserver abgespeichert und geben die korrekten Informationen der Datei wieder. Prüfbare Werte:
  - Timestamp
  - Firma
  - Standort
  - Stelle
  - Typ
  - Fachbereich
  - Dateiname

```
[
    {
        "timestamp": "2024-11-26 14:16:02",
        "firma": "Musterfirma",
        "standort": "Musterort",
        "stelle": "Muster",
        "typ": "Praxisphasen & Praktika",
        "fachbereich": "Fachbereich 01 Chemie",
```

```
        "pdf_filename": "6745d82218e43_Hallo.pdf"
    },
    {
        "timestamp": "2024-11-26 17:00:21",
        "firma": "wqe",
        "standort": "qwe",
        "stelle": "qwe",
        "typ": "Nebenjobs (in der Region)",
        "fachbereich": "Fachbereich 08 Wirtschaftswissenschaften",
        "pdf_filename": "6745fea59abbc_Grundlagen des BCM.pdf"
    }
]
```


Beispiel von Metadaten von 2 verschiedenen Uploads.

- Das Ergebnis wird dokumentiert.

Zugriffsrecht

- Es wird überprüft, ob auf die hochgeladene PDF-Datei, die sich auf dem Datei-Server befindet, zugegriffen und sie geöffnet werden kann.
- Das Ergebnis wird dokumentiert.

Zeitgesteuerte Systemfunktionen

- Eine Testdatei wird hochgeladen und der Zeitstempel genau dokumentiert.
- Grundsätzlich wird nach exakt 30 Tagen die automatische Löschung durchgeführt. Testweise kann die automatische Löschung auf einen kurzfristigen Zeitraum geändert werden, um bei einer Neuaufsetzung die Funktionalität zu überprüfen. Dazu wird im Skript "datenhaldebereinigen" (zu finden im Pfad `/etc/cron.daily/datenhaldebereinigen`) folgender Code geändert.

```
#!/bin/bash
Speicher="/home/user/caddy_test/public/uploads"
find "$Speicher" -iname "*.pdf" -type f -mtime +30 -delete
```

`-mtime +30` kann durch `-mmin +5` ersetzt werden, um die automatische Löschung auf 5 Minuten zu setzen. So würde das Skript nach der Änderung aussehen:

```
#!/bin/bash
Speicher="/home/user/caddy_test/public/uploads"
find "$Speicher" -iname "*.pdf" -type f -mmin +5 -delete
```

Nun sollte eine hochgeladene PDF, die älter als 5 Minuten ist, automatisch gelöscht werden.

- Erwartetes Ergebnis: PDF-Dateien werden im benutzerdefinierten Zeitraum automatisch gelöscht. Das Ergebnis wird dokumentiert.

Passwortgeschützte PDFs

- Eine passwortgeschützte PDF wird hochgeladen.
- Erwartetes Ergebnis: Die PDF wird erfolgreich hochgeladen. Es ist noch keine Funktion implementiert, die kennwortgeschützte PDFs erkennt. In Zukunft kann die Funktionalität so erweitert werden, dass passwortgeschützte PDFs abgelehnt werden.

Sicherheitsaspekte

Dateigrößenmanagement

- PDF über 5 MB wird hochgeladen und getestet.
- Erwartetes Ergebnis: Es wird die Warnung "Fehler beim Hochladen der Datei!" angezeigt. Der PDF-Upload wurde unterbunden und die PDF findet sich NICHT im Datenserver wieder.

- Das Ergebnis wird dokumentiert.

Dateityp-Validierung

- Nur PDF-Uploads werden akzeptiert.
- Andere Dateitypen müssen konsequent blockiert werden.
- Testweise versuchen, ein JPG-, PNG- und Text-Dokument hochzuladen.
- Erwartetes Ergebnis: Es wird auf "Datei auswählen" geklickt. Der Explorer öffnet sich, um eine PDF-Datei auszuwählen. Es sollten keine anderen Dateitypen zur Auswahl stehen. Zusätzlich wird versucht, per Drag-and-Drop eine Datei eines anderen Typs in das Feld 'Datei auswählen' zu ziehen. Der Dateiname erscheint. Klickt man nun auf Hochladen, so sollte die Fehlermeldung "Die hochgeladene Datei ist keine gültige PDF-Datei!" erscheinen.

- Das Ergebnis wird dokumentiert.

Sicherheitsscans

- Die speziell präparierten PDFs mit potenziell schädlichem Inhalt werden hochgeladen, um die Erkennungsmechanismen zu überprüfen.
- Erwartetes Ergebnis: Es werden 3 Warnungen angezeigt. Zusätzlich kommt die Fehlermeldung, dass die CAPTCHA-Verfizierung fehlgeschlagen ist.

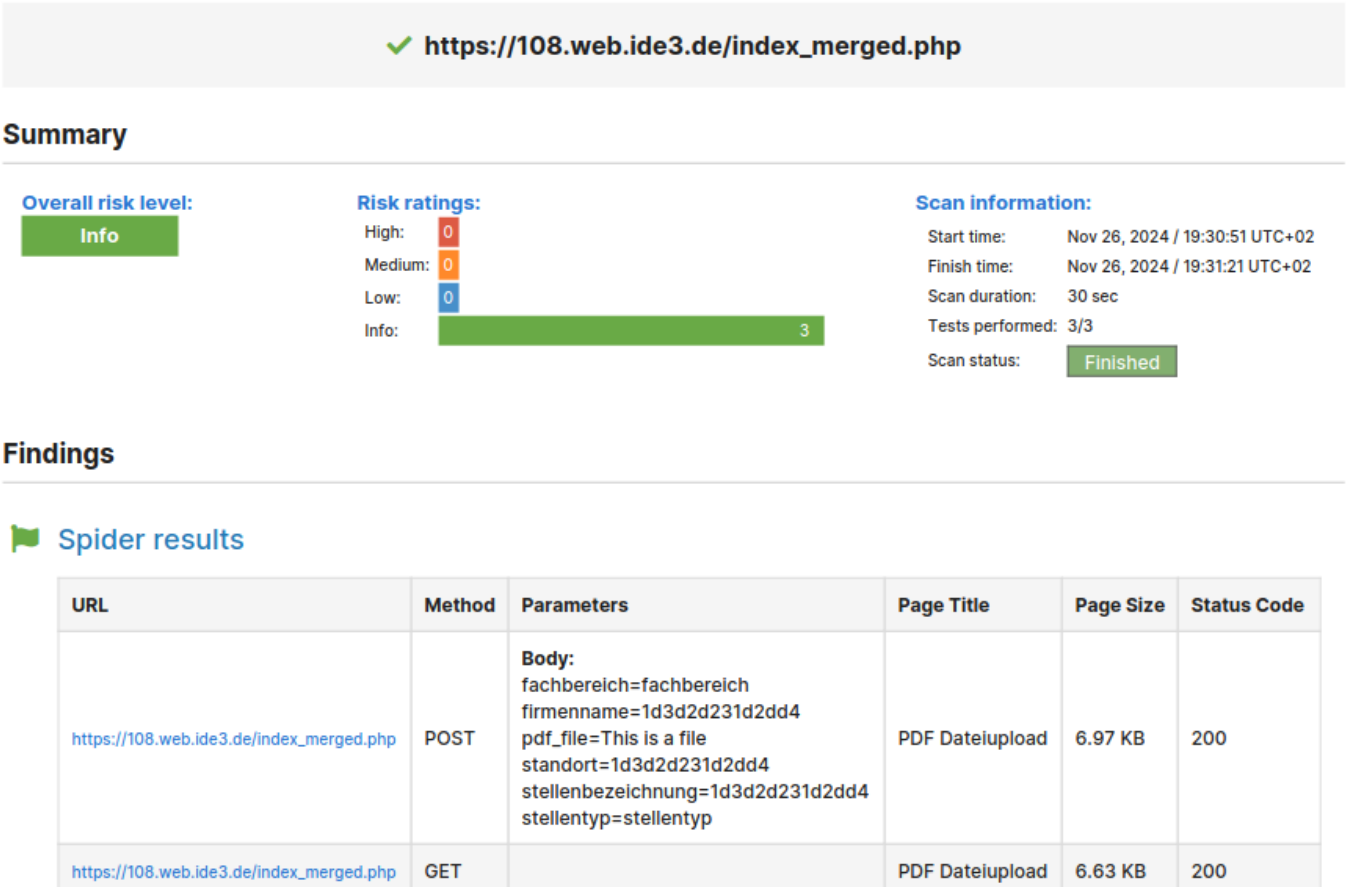
- Das Ergebnis wird dokumentiert.

Dateinamenskonventionen

- Funktion zur Umbennenung von Duplikaten mit identischem Dateinamen wird getestet. Dazu wird zweimal eine PDF-Datei mit gleichem Namen hochgeladen.
- Die PDF-Dateien erhalten nach dem Upload einen zufällig generierten Hash, der vorne am Dateinamen angehängt wird.
- Erwartetes Ergebnis: Zwei gleiche PDF-Dateien mit gleichem Namen haben auf dem Dateiserver unterschiedliche Namen, da sie einen zufälligen Hash im Namen zugeordnet bekommen.

Websicherheit

- Die Sicherheit der Seite vor XSS-Angriffen wird mit dem Online-Tool "https://pentest-tools.com/" getestet. Es sollten keine Risiken vorhanden sein. (STAND: 26.11.2024)



- Das Ergebnis wird dokumentiert.

Dokumentation und Abschlussbewertung

- Alle Testergebnisse sollen nachvollziehbar dokumentiert werden.
- Fehler werden bestenfalls mit Screenshot-Dokumentation festgehalten.
- Erstellung eines Fehlerberichtes, sollte das System nicht wie vorhergesehen funktionieren.
- Ein Gesamtbericht wird mit allen Testergebnissen verfasst. Sind alle Anforderungen und Funktionen erfüllt, so kann das System in Betrieb genommen werden.

Wirtschaftlichkeit

Für die Wirtschaftlichkeit müssen vergleichsweise wenig Faktoren berücksichtigt werden, da ein Großteil der benötigten Systeme auf Seiten der Hochschule schon vorhanden ist und für andere Funktionalitäten genutzt wird. Auch ohne die Realisierung dieses Projekts verfügt die Hochschule über VMs, die über Proxmox bereitgestellt werden, besitzt eine eigene, gehostete Domain und führt Wartungsarbeiten an den genutzten Diensten durch. Für dieses spezifische Projekt würden hauptsächlich Wartungskosten und darin inbegriffene Anpassungskosten aufkommen. So müssten die Systeme in regelmäßigen Abständen überprüft und gewartet werden, um alle Funktionalitäten zu gewährleisten. Und im Falle, dass zum Beispiel der kostenlose CAPTCHA-Dienst von Cloudflare eingestellt werden würde, müsste das Projekt so angepasst werden, dass ein anderer Dienst verwendet wird.

Quellenverzeichnis

---

1. Cloudmersive, aufgerufen am 20.10.24
2. IPQS, aufgerufen am 20.10.24
3. PDFExaminer Tool, aufgerufen am 20.10.24
4. PDFiD, aufgerufen am 20.10.24
5. peepdf-3, aufgerufen am 20.10.24
6. QuickSand, aufgerufen am 20.10.24
7. VirusTotal, aufgerufen am 20.10.24
8. Cloudflare Turnstile Implementierung Dokumente
9. CAPTCHA-Dienst Vergleich
10. Cron, aufgerufen am 11.11.24
11. peepdf, aufgerufen am 25.11.24
12. Simple-PDF-Analyzer, aufgerufen am 25.11.24
13. Stellenbörse Hochschule Niederrhein, aufgerufen am 24.11.2024