HAW HAMBURG

Bachelorarbeit

Clemens Reimer

Automatisierte Ermittlung und Visualisierung des IT-Sicherheitsniveaus von IT-Systemen einer Organisation

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Clemens Reimer

Automatisierte Ermittlung und Visualisierung des IT-Sicherheitsniveaus von IT-Systemen einer Organisation

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachterin: Prof. Dr. Bettina Buth

Eingereicht am: 08.02.2022

Clemens Reimer

Thema der Arbeit

Automatisierte Ermittlung und Visualisierung des IT-Sicherheitsniveaus von IT-Systemen einer Organisation

Stichworte

IT Sicherheit, Schwachstellenmanagement, Automatisierung

Kurzzusammenfassung

Im Zuge dieser Bachelorarbeit wird eine Anwendung entwickelt, die das IT-Sicherheitsniveau einer Organisation automatisiert mittels Netzwerkscans ermittelt und die Informationen anschließend dem Benutzer zur Verfügung stellt. Es werden Basisinformationen zu den Systemen gesammelt und erkannte laufende Anwendungen werden spezifischen Tests unterzogen.

Clemens Reimer

Title of Thesis

Automated determination and visualization of the it security level of IT systems within an organization

Keywords

IT security, vulnerability management, automation

Abstract

In the course of this thesis an application is developed, which automatically determines the IT security level of an organization with the help of network scans. Afterwards, the information is provided to the user. Basic information regarding the hosts is gathered and specific tests are processed on active service applications.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Die eng vernetzte Gesellschaft und die steigende Komplexität und Konnektivität der IT-Systeme bewirken, dass große Mengen von Informationen zwischen IT-Systemen über Netzwerke ausgetauscht werden. Darunter fallen auch sensible personenbezogene Daten, die besonders schützenswert sind. Durch gesetzliche Vorgaben zur Offenlegung der Informationen bei Datenlecks durch das Bundesdatenschutzgesetz wächst zunehmend die Bedeutung der Informationssicherheit für die Gesellschaft. Regelmäßig wird in den öffentlichen Medien von Ereignissen berichtet, bei denen durch Sicherheitsvorfälle kritische personenbezogene Daten unberechtigt offengelegt werden. Für Unternehmen und Institutionen stellen solche Vorfälle ein hohes Risiko dar, da die Konsequenz eines Datenlecks neben finanziellen Schäden auch erhebliche Schäden in der Reputation der betroffenen Organisation sein kann.

So berichtete Heise online[1] am 03.05.2021 von einem Angriff auf die TU Berlin[2]. Nach dem Angriff wurden am 30.04.2021 die Server vorsorglich heruntergefahren, mit der Folge, dass die IT-Dienste der Universität vorerst nur eingeschränkt verfügbar waren. Ein Notfallstab von externen IT-Sicherheitsexperten wurde für die Analyse ins Leben gerufen, da die Vermutung bestand, dass Ransomware auf die Systeme eingespielt wurde.[16]

Am 20.05.2021 wurde daraufhin berichtet, dass es dem Angreifer gelang, das Active Directory der TU Berlin zu kopieren und damit Zugriff auf personenbezogene Daten wie Benutzernamen, kryptographisch gesicherte Passwörter, E-Mail-Adressen, Personalnummern von Beschäftigten und Matrikelnummern der Studierenden zu erhalten. Weitere Datenabflüsse konnten im Rahmen der forensischen Untersuchung nicht nachgewiesen werden.[15]

[1] https://www.heise.de
[2] https://www.tu.berlin/

Eine weitere Stellungnahme erfolgte am 03.06.2021, der zufolge die TU Berlin mit monatelangen IT-Einschränkungen rechnet bis die zentralen Dienste nach dem Ransomware-Angriff wieder zur Verfügung stehen. Da das Kern-SAP-System der Universität betroffen ist, können die Studierenden die Semester nicht wie gewohnt beenden und es muss mit anderen Hochschulen kooperiert werden, um Übergangslösungen zu schaffen. Auch das Personal selbst ist betroffen, denn die Gehälter können nicht wie gewohnt ausgezahlt werden, sondern müssen unter Vorbehalt auf Basis des Vormonats geschätzt werden.[8]

Die TU Berlin selbst nimmt auf ihrer Webseite Stellung zu dem Vorfall und berichtet, dass Dateien, die bei dem Angriff abgeflossen sind, im Darknet veröffentlicht wurden. Weiterhin arbeitet die Universität daran, die Funktionalität aller betroffenen Dienste und Prozesse wiederherzustellen, um wieder in den Normalbetrieb zurückzukehren. Dabei liegt der Fokus auf den SAP-Systemen, die für die Verwaltungsprozesse und die Services für Studium und Lehre erforderlich sind. Zusätzlich wurde eine Webseite, die „Dienste Ampel“[3], eingeführt, auf der man den aktuellen Status der einzelnen Dienste einsehen kann. Im Rahmen von Beiträgen auf dem Portal der TU Berlin wird fortlaufend über die Entwicklungen und Fortschritte berichtet.[2]

Die Konsequenzen eines solchen Angriffes sind ein enormer finanzieller Schaden und eine erhebliche Schädigung der Reputation der betroffenen Institution bzw. des Unternehmens. Um sich bestmöglich zu schützen, ist es zwingend erforderlich, das IT-Sicherheitsniveau der Organisation laufend zu kontrollieren und bestehende Schwachstellen möglichst zeitnah zu schließen. Aufgrund der Komplexität der heutigen IT-Systeme und der damit einhergehenden Vielzahl an möglichen Angriffspunkten ist eine manuelle Überprüfung in regelmäßigen Abständen ineffizient und kostspielig und es ist nicht sichergestellt, dass bei der Überprüfung dem Prüfenden selbst keine Fehler unterlaufen. Um also eine regelmäßige und ressourcenschonende Überprüfung gewährleisten zu können, ist eine Automatisierung dieses Prozesses notwendig. Die Planung und Umsetzung der Automatisierung der Werkzeugunterstützung eines solchen Prozesses soll Inhalt der vorliegenden Arbeit sein.

1.2 Ziel der Arbeit

Im Rahmen dieser Bachelorarbeit soll eine Applikation entstehen, deren Ziel es ist, IT-Sicherheitsbeauftragte dabei zu unterstützen, sich einen Überblick über das aktuelle IT-

[3] https://www.campusmanagement.tu-berlin.de/menue/dienste/zecm_dienste_ampel/

Sicherheitsniveau einer Organisation zu verschaffen und konkrete Schwachstellen aufzuzeigen. Mittels Netzwerkscan werden alle zur Organisation gehörenden Systeme ermittelt, um dann in einem weiteren Analyseprozess Informationen zu den einzelnen Systemen zu sammeln. Diese Informationen werden anschließend aufbereitet und in einer Webapplikation zugänglich gemacht.
Im Detail werden ausgehend von festgelegten IP-Adressbereichen alle aktiven Subsysteme und deren offene Ports ermittelt und für die Weiterverarbeitung bereitgestellt. Zu jeder ermittelten IP-Adresse wird ein Analyseprozess gestartet, der über das Netzwerk Informationen zu den Eigenschaften des Systems sammelt und die Ergebnisse persistiert, um eine Weiterverarbeitung der Informationen zu ermöglichen.
Nachdem der Analyseprozess abgeschlossen ist, wird die National Vulnerability Database des National Institute of Standards and Technology genutzt, um aktuelle Informationen zu bekannten Schwachstellen und deren Bewertung gemäß des Common Vulnerability Scoring System zu erhalten und darauf basierend Handlungsbedarfe zu identifizieren und zu priorisieren. Abschließend werden die gewonnenen Informationen im Rahmen der Webapplikation grafisch und tabellarisch aufbereitet, um einen kompakten Überblick über das IT-Sicherheitsniveau der Organisation zu erhalten. Bei der Ausgestaltung der Visualisierung stehen Übersichtlichkeit und Nachvollziehbarkeit der dargestellten Informationen im Fokus.

1.3 Zielgruppe der Arbeit

Diese Arbeit ist für IT-Sicherheitsbeauftrage und Interessierte am Bereich der IT-Sicherheit bestimmt. Grundkenntnisse im Bereich der IT-Sicherheit, verteilten Systemen und Rechnernetzen werden als vorausgesetzt angenommen. Es soll den Lesern eine Möglichkeit aufgezeigt werden, wie das IT-Sicherheitsniveau einer Organisation automatisiert ermittelt werden kann. Außerdem soll die grafische Visualisierung der Informationen bei der Entscheidungsfindung unterstützen, indem aufgezeigt wird, wo der zwingend notwendigste Handlungsbedarf besteht.

1.4 Aufbau der Arbeit

Die Arbeit gliedert sich in 8 Kapitel und den Anhang. Kapitel 1 führt in die Arbeit ein, grenzt das Ziel der Arbeit grob ab und beschreibt, an wen sich die Arbeit richtet.

Kapitel 2 thematisiert die notwendigen Grundlagen, die für das Verständnis der Arbeit relevant sind und ordnet die Arbeit in den Kontext des Informationssicherheitsmanagements ein. Kapitel 3 steckt die Rahmenbedingungen ab und definiert die funktionalen und nicht-funktionalen Anforderungen an die Anwendung. Das Kapitel 4 widmet sich der Konzeption der Systemarchitektur, setzt die Anwendung in einen Benutzerkontext und beschreibt das Modell des Analyseprozesses. Kapitel 5 behandelt die Implementierungsdetails des Serverteils und Clientteils. Das Kapitel 6 beschreibt, wie die Qualität der Anwendung überprüft und sichergestellt werden kann. Kapitel 7 dient der Evaluation aufgetretener Probleme und der Arbeitsergebnisse. Kapitel 8 gibt einen Ausblick darauf, wie die Anwendung sinnvoll erweitert werden kann. Schlussendlich ist im Anhang eine tabellarische Aufstellung der Metriken des Common Vulnerability Scoring System zu finden.

2 Grundlagen zur Überprüfung von IT-Sicherheitsniveaus

In dem zweiten Kapitel werden zunächst die relevantesten Begriffe geklärt, um Mehrdeutigkeiten im Kontext dieser Arbeit auszuschließen. Dem folgt die Thematisierung des Schwachstellenmanagements als Teil des Informationssicherheitsmanagements. Damit wird der Inhalt dieser Arbeit in den Kontext des Informationssicherheitsmanagements gesetzt. Als nächstes wird eine Möglichkeit, Schwachstellen zu bewerten, vorgestellt, das Common Vulnerability Scoring System. Da es auf internationaler Ebene das bedeutsamste Schema ist, wird es dieser Arbeit als Bewertungssystem zu Grunde gelegt. Abschließend wird ein Dienst des National Institute of Standards and Technology, die National Vulnerability Database, näher betrachtet, denn diese dient der Arbeit als Quelle der Information über aktuelle Schwachstellen.

2.1 Begriffsklärung

Um Mehrdeutigkeiten zu vermeiden, werden im folgendem Abschnitt Begriffe erklärt und abgegrenzt, die für diese Arbeit von hoher Relevanz sind.

2.1.1 IT-System

Ein IT-System ist ein Zusammenschluss von einzelnen Komponenten zu einem dynamischen technischen Gesamtsystem mit der Fähigkeit, Informationen zu speichern und zu verarbeiten.[5] IT-Systeme besitzen eine klare Grenze (engl. boundary), anhand welcher sich der zu schützende Bereich von dem restlichen Umfeld abgrenzen lässt.

2.1.2 Bedrohung

Alle IT-Systeme sind diversen Bedrohungen (engl. threats) ausgesetzt, welche zur Gefahr für das IT-System werden können. Das Ziel dieser Bedrohungen ist es, Schwachstellen auszunutzen, um die Sicherheitsvorkehrungen zu untergraben. Zu den möglichen Auswirkungen gehören der Verlust von: Datenintegrität, Informationsvertraulichkeit, Verfügbarkeit sowie der Authentizität von Subjekten.[5] Bedrohungen werden hinsichtlich des Ursprungs in natürliche (environmental), technische (technical) und von Menschen verursachte (man-made) unterschieden. Zusätzlich können die Bedrohungen bezüglich der Absicht in neutrale (neutral), unbeabsichtigte (unintentional), absichtliche haptische (intentional physical) und absichtliche nicht-haptische (intentional non-physical) klassifiziert werden. [12]

2.1.3 Schwachstelle, Verwundbarkeit und Exploit

Unter einer Schwachstelle (engl. weakness) eines Systems versteht man eine Schwäche des Systems oder einen Punkt, an dem das System verwundbar ist. Eine Verwundbarkeit (engl. vulnerability) ist eine Schwachstelle, mittels der sich Sicherheitsvorkehrungen eines System umgehen lassen.[5] Eine Schwachstelle kann im System, im Prozess, in der Technologie, in einer Person oder bei Kontrollen begründet liegen.[12]
Ein Exploit stellt eine Angriffsmöglichkeit dar, um durch Ausnutzung von Schwachstellen Zugang zu einem System zu erlangen. Ein Exploit kann nur die theoretische Beschreibung dieser Möglichkeit sein oder aber anwendbarer Quellcode, welcher direkt eingesetzt werden kann. Exploits werden auch für die Dokumentation von Schwachstellen verwendet, um diese dann mit Updates und Patches zu beseitigen. Für die Überprüfung von Systemen auf Anfälligkeiten gegenüber bekannter Angriffsmöglichkeiten können Exploits ebenfalls eingesetzt werden.[9]

2.1.4 Risiko

Im Rahmen der IT-Sicherheit wird unter dem Begriff Risiko (engl. risk) eine Kombination aus Eintrittswahrscheinlichkeit und potentieller Schadenshöhe bezüglich einer Bedrohung verstanden. Das Schadenspotential kann dabei quantitativ oder qualitativ bewertetet werden.[5]

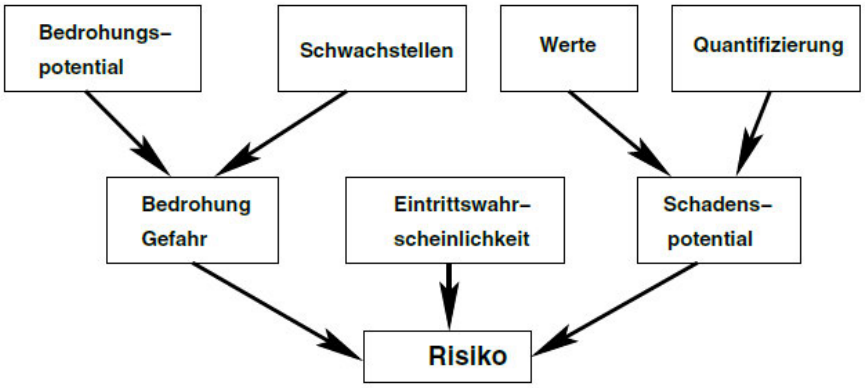

Abbildung 2.1: Zusammenhang zwischen Schwachstellen, Bedrohungen und Risiken[4]

2.2 Schwachstellenmanagement als Teil des Informationssicherheitsmanagements

Nach Klärung der grundlegenden Begriffe wird nun das Schwachstellenmanagement als Teil des Informationssicherheitsmanagements näher betrachtet, um das Verständnis zu schärfen, in welches Teilgebiet des Informationssicherheitsmanagementprozesses diese Arbeit einzuordnen ist.

Das Ziel des Informationssicherheitsmanagements (ISM) ist es, eine Organisation bei der Erreichung ihrer Ziele unter Einhaltung der Sicherheitsanforderungen an Vertraulichkeit, Integrität und Verfügbarkeit optimal zu unterstützen. ISM ist ein komplexer kontinuierlicher Prozess, der begleitend zu allen anderen Prozessen einer Organisation gesteuert und überwacht werden muss. Die Strategien und Konzepte, die im Rahmen des Prozesses angewendet werden, müssen ständig auf Leistungsfähigkeit und Wirksamkeit überprüft werden und bei Bedarf angepasst werden. In aller Regel kommt zur Unterstützung ein Informationssicherheitsmanagementsystem (ISMS) zum Einsatz. Ein ISMS umfasst alle Methoden und Instrumente, die zur Steuerung und Lenkung der Informationssicherheit notwendig sind, um die Sicherheitsziele und Geschäftsziele der Organisation zu erreichen.[12]

Um ein wirkungsvolles Informationssicherheitsmanagement betreiben zu können, ist es erforderlich das Informationssicherheitsmanagementsystem ständig zu verbessern und zu

[4]Quelle: IT-Sicherheit Konzepte - Verfahren - Protokolle[5]

überwachen, deswegen bildet der Plan-Do-Check-Act-Zyklus die Basis des ISM. (Abb. 2.2)

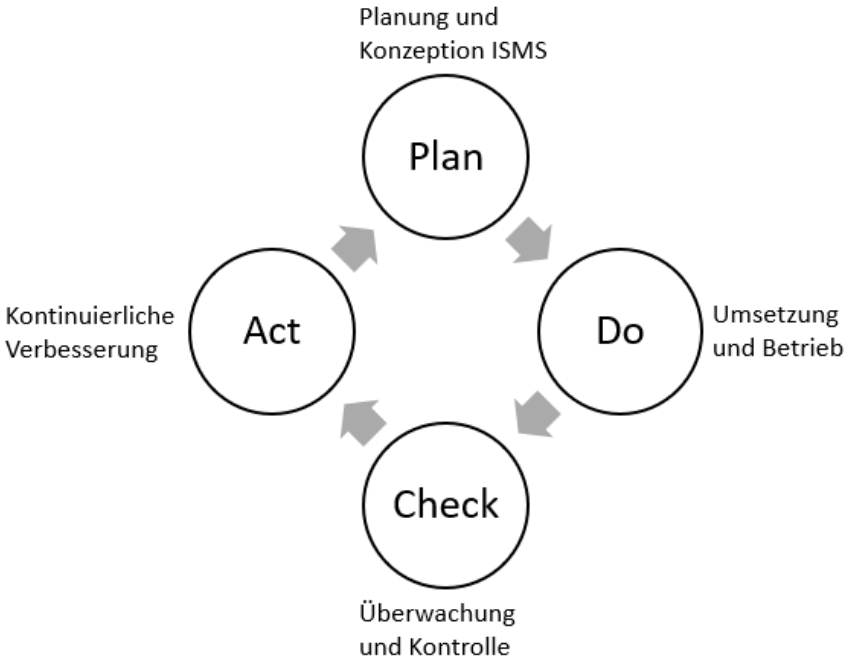

Abbildung 2.2: PDCA Zyklus für ein ISMS

In der **Plan-Phase** findet eine Risikoanalyse und Schutzbedarfsfestellung statt, um zu ermitteln, welche Sicherheitsstrategie von Nöten ist. Von der Sicherheitsstrategie wird das Sicherheitskonzept abgeleitet und es wird eine Auswahl von geeigneten Sicherheitsmaßnahmen getroffen, um das Sicherheitskonzept umzusetzen.[12]

In der zweiten Phase, der **Do-Phase**, werden dann die geplanten Strukturen etabliert und Sicherheitsmaßnahmen umgesetzt und in Betrieb genommen. Neben den technischen Aspekten stellt auch der Faktor Mensch ein wesentliches Sicherheitsrisiko dar, dem mit Sensibilisierungs- und Awarenessmaßnahmen entgegengewirkt wird.[12]

Die **Check-Phase** dient der Überprüfung der Effizienz und Wirksamkeit der Sicherheitsmaßnahmen und der Erkennung und Bearbeitung von Sicherheitsvorfällen. Um diese Überprüfung vornehmen zu können, werden neben den Ergebnissen des Monitorings die Berichte von internen und externen Prüfungen (engl. audits) genutzt. Ein Teil des IT-Sicherheitsaudits (engl. Security Audit) ist die Schwachstellenanalyse als Teil des

Schwachstellenmanagements[12], welches am Ende dieses Abschnittes genauer erläutert wird.

> „Unter dem englischen Begriff Security Audit versteht man in Deutschland generell die manuelle oder automatische Erkennung und Meldung sicherheitsrelevanter Vorgänge und Systemzustände. Sie dienen dem Schutz der Systeme, der Daten und deren Benutzer.“[12]

In der letzten Phase, der **Act-Phase**, werden eventuell festgestellte Fehler beseitigt und die bestehenden Sicherheitsmaßnahmen angepasst, um eine Verbesserung zu erzielen. Nach Beendigung kann der PDCA-Zyklus erneut durchlaufen werden, um die laufende Überprüfung des Informationssicherheitsmanagementsystems zu gewährleisten.[12]

Das Schwachstellenmanagement als Teil des Informationssicherheitsmanagements ist wiederum ein eigener PDCA-Zyklus mit dem Ziel, die Anzahl der Schwachstellen in einem System zu reduzieren und damit das Sicherheitsniveau zu erhöhen.

> „Das Schwachstellenmanagement bezeichnet die zyklische Ausübung der Identifikation, Klassifikation, Beseitigung und Abschwächung und der Prävention von Schwachstellen.“[13]

2.3 Common Vulnerability Scoring System

Nachdem das Thema dieser Arbeit in den Kontext des Informationssicherheitsmanagements gesetzt ist, wird nun verdeutlicht, wie Schwachstellen mit Hilfe des Common Vulnerability Scoring System bewertet werden können. Die Informationen dieses Abschnittes basieren alle, soweit nichts anderes angegeben ist, auf der Quelle „Common Vulnerability Scoring System version 3.1 Specification Document Revision 1“.[14]

Das Common Vulnerability Scoring System ist ein von der amerikanischen Non-Profit-Organisation Forum of Incident Response and Security Teams (FIRST)[5] verwaltetes offenes Framework, um Schwachstellen, samt ihrer technischen Eigenschaften, in Software, Hardware oder Firmware einheitlich bewerten und kommunizieren zu können. Mit der Version CVSS 3.0 hat sich CVSS zu einem internationalen Standard weiterentwickelt[7],

[5] https://www.first.org/cvss/

heute liegt mit CVSS 3.1 die aktuellste Version vor. Der Bewertung liegen drei Gruppen von Metriken zu Grunde: die Base Metric Group und zwei optionale Gruppen, die Temporal Metric Group und die Environmental Metric Group. (Abb. 2.3)

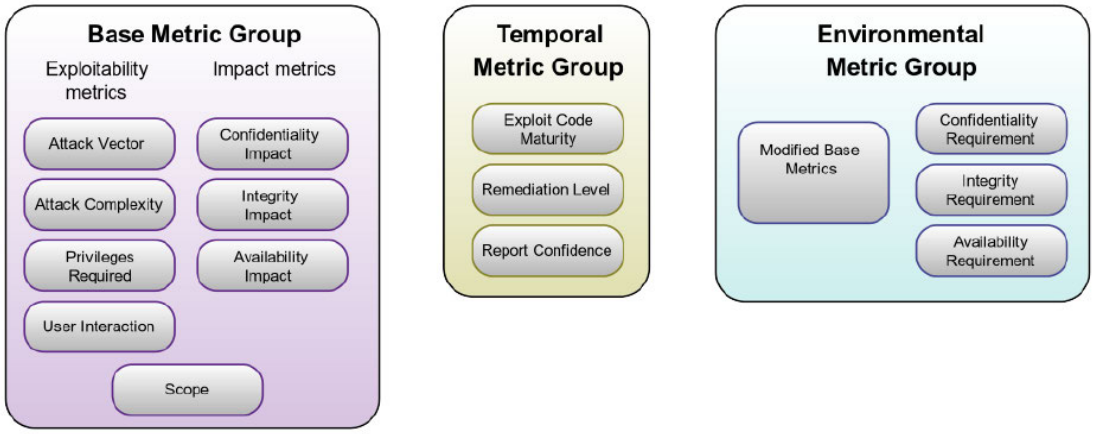

Abbildung 2.3: CVSS - Gruppen der Metriken[6]

2.3.1 Scoring

Das Ergebnis einer Bewertung durch das Common Vulnerability Scoring System ist zum einen der berechnete Score und zum anderen ein Vector-String, der Aufschluss darüber gibt, welche Metriken der Berechnung zu Grunde liegen. Der Score liegt in einem Bereich zwischen 0.0 und 10.0, wobei 10.0 das kritischste Szenario widerspiegelt. Eine Zuordnung der quantitativen zu den qualitativen Werten ist der Tabelle 2.1 zu entnehmen.

Rating	**CVSS Score**
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Tabelle 2.1: Qualitative Schweregradbewertungsskala[7]

Der Berechnung des Scores einer Schwachstelle liegen mindestens die Metriken der Base Metrics Group zu Grunde, der daraus resultierende Wert wird auch Base Score genannt.

[6] Quelle: Common Vulnerability Scoring System version 3.1 Specification Document Revision 1[14]
[7] Quelle: Common Vulnerability Scoring System version 3.1 Specification Document Revision 1[14]

Die Base Metrics Group umfasst die Exploitability Metrics und die Impact Metrics, die im folgenden Abschnitt 2.3.2 im Detail erklärt werden. Nachdem der Base Score berechnet wurde, besteht die Möglichkeit, diesen Wert durch die optionalen Metrikgruppen Temporal Metrics Group und Environmental Metrics Group bezüglich fallspezifischer Eigenschaften zu verfeinern. Der Prozess ist in der Abbildung 2.4 dargestellt.

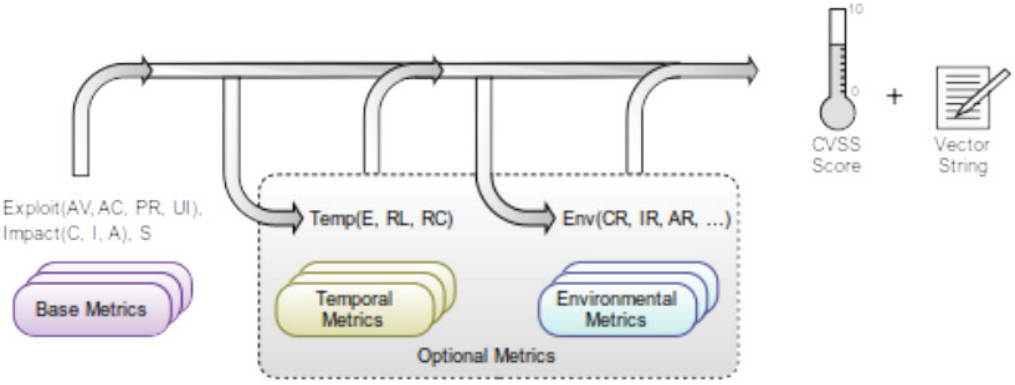

Abbildung 2.4: CVSS - Bewertungsprozess[8]

2.3.2 Metriken

Im folgenden Abschnitt werden die drei Gruppen der Metriken Base Metric Group, Temporal Metric Group und Environmental Metric Group samt der konkreten Metriken kurz erläutert.

Base Metric Group

Die Base Metric Group repräsentiert die inhärenten Eigenschaften einer Schwachstelle, wobei die Eigenschaften konstant über Zeit und Betriebsumgebung bleiben. Die Base Metric Group setzt sich aus zwei Arten von Metriken zusammen: Exploitability Metrics und Impact Metrics.

Die Exploitability Metrics beschreiben die Leichtigkeit und die technischen Mittel, die benötigt werden, um eine Schwachstelle einer Komponente auszunutzen. Man spricht im Zusammenhang einer solchen Komponente deshalb von einer vulnerable component. Der Gruppe von Exploitability Metrics liegen fünf konkrete Metriken zu Grunde, die

[8] Quelle: Common Vulnerability Scoring System version 3.1 Specification Document Revision 1[14]

wiederum verschiedene Werte annehmen können, welche der Tabelle A.1 zu entnehmen sind. Der jeweils fett gedruckte Wert in den Metrik-Tabellen hat dabei den größten Einfluss auf die Bewertung.

Die Metrik Attack Vector (AV) beschreibt dabei den Rahmen, in dem das Ausnutzen der Schwachstelle möglich ist. Unter der Attack Complexity (AC) Metrik fließen Bedingungen ein, die außerhalb der Einflussmöglichkeiten des Angreifers liegen, welche erfüllt sein müssen, um einen erfolgreichen Angriff durchzuführen. Priviliges Required (PR) beschreibt die Anforderungen an bestimmte Rechte, die vor dem erfolgreichen Ausnutzen der Schwachstelle erlangt worden sein müssen. Ob die Aktivität eines anderen Nutzers als dem Angreifer erforderlich ist, um eine erfolgreichen Angriff durchzuführen, fließt mit der User Interaction (UI) Metrik mit ein. Die Scope (S) Metrik lässt einfließen, ob ein Angriff auf eine verwundbare Komponente Auswirkungen auf Ressourcen einer Komponente außerhalb des Sicherheitsumfangs der angegriffenen Komponenten hat.

Die Impact Metrics stellen die direkten Konsequenzen auf die Vertraulichkeit, Integrität und Verfügbarkeit einer erfolgreichen Ausnutzung einer Schwachstelle dar. Da hier die Folgen für eine Komponente gemeint sind, spricht man von einer impacted component. Eine Übersicht über die Metriken und deren mögliche Werte zeigt die Tabelle A.1.

Die erste Metrik Confidentiality (C) bezieht das Ausmaß einer erfolgreichen Ausnutzung einer Schwachstelle auf die Vertraulichkeit von Informationen der impacted component mit ein. Die zweite Metrik Integrity (I) beschreibt den Einfluss eines solchen Angriffes auf die Glaubwürdigkeit und Richtigkeit von Informationen. Die letzte unter der Kategorie impact metrics einfließende Metrik ist Availability (A). Hier wird nicht die Wirkung auf die Informationen, sondern die Auswirkung auf die Verfügbarkeit von Informationen, gemessen.

Temporal Metric Group

Die optionale Temporal Metric Group setzt sich aus drei Metriken zusammen. Eine Übersicht ist der Tabelle A.2 zu entnehmen.

Die Exploit Code Maturity (E) Metrik misst dabei die Wahrscheinlichkeit eines Angriffes auf eine Schwachstelle und basiert typischerweise auf dem Reifegrad der Angriffstechniken und der Verfügbarkeit des Quellcodes der Schadsoftware. Unter der Metrik Remediation Level (RL) wird mit einbezogen, ob derzeit bereits Patches oder temporäre Workarounds

für das Schließen der Schwachstelle bestehen. Die letzte Metrik Report Confidence (RC) berücksichtigt die Sicherheit, mit der eine Schwachstelle beschrieben werden kann.

Environmental Metric Group

Mit der optionalen Environmental Metric Group wird den Analysten die Möglichkeit an die Hand gegeben, die Bewertung basierend auf der Betriebsumgebung einer Organisation und dem jeweiligen Einsatzszenario anzupassen. Es besteht die Möglichkeit, die Base Metric Group Werte mit Hilfe der Modified Base Metrics zu überschreiben. Außerdem können besondere Ansprüche an die Vertraulichkeit, Integrität und Verfügbarkeit mit Hilfe der Security Requirements Group zum Ausdruck gebracht werden. Die komplette Aufstellung ist der Tabelle A.3 im Anhang zu entnehmen.

2.3.3 Vector String

Um die Kommunikation und den Austausch von den Informationen einfach zu gestalten, wird im Rahmen des CVSS ein Vector String verwendet. Dieser Vektor enthält alle Informationen, welche Metriken in die Berechnung des Scores einer Schwachstelle mit eingeflossen sind. Die jeweils eingeflossenen Metriken werden mit dem dementsprechenden Kürzel referenziert und zusätzlich werden die Kürzel für den der Berechnung zu Grunde liegenden Wert angefügt. Ein Beispiel für einen solchen Vektor ist in der Abbildung 2.5 skizziert und die gesamten Kürzel sind in den Tabellen A.0.1 im Anhang aufgelistet.

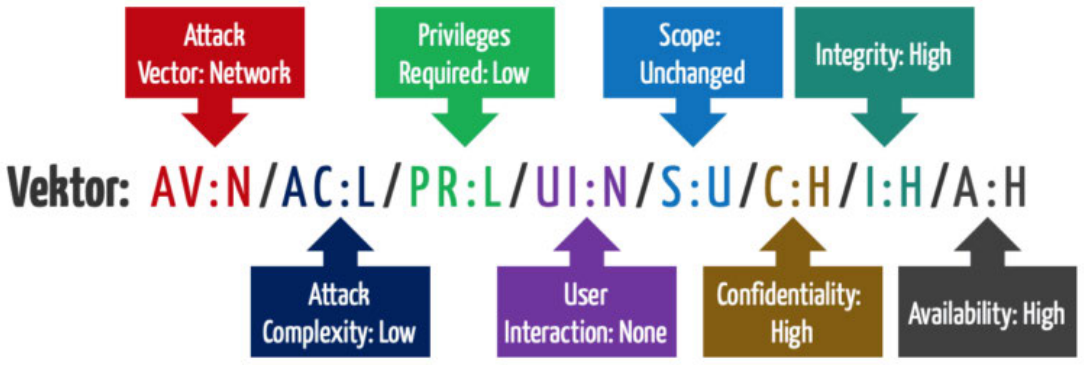

Abbildung 2.5: CVSS - Repräsentation des Vektor Strings[9]

[9] https://www.johner-institut.de/blog/iec-62304-medizinische-software/cvss-common-vulnerability-scoring-system/

2.4 National Vulnerability Database / Common Vulnerabilities and Exposures

Eine Bewertung einer Schwachstelle durch ein Bewertungssystem ist die Grundlage, um die Kritikalität von Schwachstellen kommunizieren zu können. Deswegen gibt es einen Dienst, der von dem National Institute of Standards and Technology angeboten wird, die National Vulnerability Database. Diese Datenbank stellt alle relevanten Informationen über aktuell bekannte Schwachstellen in Software, Hardware und Firmware zur Verfügung. Die Schwachstellen werden mit Hilfe des Common Vulnerability Scoring System bewertet und erhalten eine eindeutige Kennung, die CVE-ID. Die aktuellste Schnittstelle zur National Vulnerability Database stellt eine REST-Schnittstelle dar. Über diese ist es möglich, Anfragen an die National Vulnerability Database zu stellen und als Antwort wird ein JSON-Objekt nach dem normierten Schema[10] mit den angeforderten Informationen zurückgegeben.

Common Vulnerabilities and Exposures, kurz CVE, ist ein Standard, um die global eindeutige Referenzierung von Schwachstellen und Expositionen in verschiedenen Schwachstellendatenbanken und die automatisierte Schwachstellenanalyse mit Analysetools zu ermöglichen. In dem CVE-Katalog werden keine Informationen zum Risiko, zur Auswirkung und zu den detaillierten technischen Informationen einer Schwachstelle bereitgestellt. Der CVE-Katalog enthält lediglich die CVE-ID, einen Statusindikator, Datumsangaben und eine kurze Beschreibung der Schwachstelle. Zusätzlich stellt der CVE-Katalog Referenzen zu den dazugehörigen Schwachstellenberichten bereit, z.B. der Bewertung durch das National Institute of Standards and Technology in der National Vulnerability Database.[4]

[10] https://csrc.nist.gov/schema/nvd/feed/1.1/nvd_cve_feed_json_1.1.schema

3 Rahmenbedingungen und Anforderungen

Nach der Klärung der Grundlagen werden in diesem Abschnitt zuerst die Rahmenbedingungen des Projektes und anschließend die funktionalen und nicht-funktionalen Anforderungen an die Applikation geklärt, um die Zielsetzung so weit zu konkretisieren, dass mit der Entwicklung begonnen werden kann.

3.1 Rahmenbedingungen

Das Projekt soll als Client-Server-Applikation realisiert werden, um die Anforderungen an die Betriebsumgebung gering zu halten.
Als Programmiersprache des Serverteils wird Python[11] zum Einsatz kommen. Python Anwendungen laufen auf jeglichen System ohne Einbußen in Performanz und aufgrund der steigenden Popularität gibt es zahlreiche Bibliotheken. Python ist im Jahr 2021 vor C und Java die beliebteste Programmiersprache.[12]

Der Clientteil wird in der Programmiersprache JavaScript[13] realisiert. Neben der guten Performanz bietet auch JavaScript eine breit aufgestellte Community mit vielen Bibliotheken. Heutzutage wird JavaScript als client-seitige Programmiersprache in 97.3% aller Webprojekte eingesetzt.[14]

[11] https://www.python.org/
[12] https://statisticstimes.com/tech/top-computer-languages.php
[13] https://www.javascript.com
[14] https://w3techs.com/technologies/details/cp-javascript

3.2 Funktionale Anforderungen

3.2.1 Analyse einer Domain und Auflösen der IP Adressen

Der Benutzer soll die Möglichkeit haben, Domainnamen an die Applikation zu übergeben. Es werden daraufhin alle dazugehörigen Subdomains und mittels des Domain Name System die dazugehörigen Informationen, speziell die IP-Adressen, ermittelt und für die Weiterverarbeitung bereitgestellt.

3.2.2 Stammdatenbank

Zusätzlich besteht die Möglichkeit, eine Stammdatenbank zu befüllen, welche die Ansprechpartner mitsamt Kontaktdaten für die ermittelten IP-Adressen bzw. Domains beinhaltet. Die Befüllung dieser Datenbank soll mittels Import einer CSV-Datei möglich sein.

3.2.3 Gutartiges Verhalten der Applikation

Es soll für jede ermittelte IP-Adresse ein Verarbeitungsprozess angestoßen werden, der sich prinzipiell gutartig verhalten soll. D.h. es sollen nur protokollkonforme Scanverfahren zum Einsatz kommen und Internetdienste werden nicht aggressiv genutzt, um Denial-of-Service Probleme zu minimieren. Die Nutzung von Exploits ist nicht vorgesehen, um schädliche Veränderungen von Anwendungsdaten auszuschließen.

3.2.4 Scan der Ports

Der Verarbeitungsprozess sieht es zunächst vor, die Erreichbarkeit der IP-Adresse zu prüfen und im positiven Fall soll anschließend die Ermittlung der offenen Ports erfolgen. Über die Benutzeroberfläche soll es möglich sein, sowohl die TCP-Portbereiche als auch die UDP-Portbereiche zu konfigurieren. Auf Basis der Konfiguration werden die entsprechenden Ports geprüft. Sollte die IP-Adresse nicht erreichbar sein, soll ein neuer Versuch zu einem späteren Zeitpunkt erfolgen.

3.2.5 Analyse offener Ports

Wenn ein offener Port ermittelt wurde, soll dieser im Rahmen des nächsten Verarbeitungsschrittes genutzt werden, um weitere Informationen über die Anwendung, die über diesen Port bereitgestellt wird, zu sammeln. Hierbei soll nach Möglichkeit das Betriebssystem ermittelt werden und es sollen vor allem die Angaben aus den Dienste-Bannern Berücksichtigung finden.

Prüfung bei offenem SSH-Port

Wird ein offener SSH-Port gefunden, soll geprüft werden, ob das Passwortverfahren oder das Public-Key-Verschlüsselungsverfahren für die Authentifizierung unterstützt wird. Das Programm soll hier nicht den Brute Force Ansatz verfolgen und die Anmeldung mit einem Passwort versuchen, sondern es soll ermittelt werden, ob die Anmeldung mit einem Passwort prinzipiell möglich wäre. Diese Prüfung soll für eine Liste von Accounts vorgenommen werden. Diese Liste soll über die Benutzeroberfläche verwaltbar sein.

Prüfung bei offenem SMTP-Port

Wird ein offener SMTP-Port gefunden, sollen die entsprechenden MX-Records ermittelt werden.

Prüfung bei offenem HTTP- oder HTTPS-Port

Im Falle, dass ein offener HTTP- oder HTTPS-Port gefunden wird, soll geprüft werden, ob eine HTML-Datei ohne weiteren Pfad oder Dateinamen in der URL heruntergeladen werden kann. Anschließend soll die Serverantwort im Hinblick auf den Einsatz des Basic Authentication-Verfahrens und eines Redirect geprüft werden. Im Falle eines Redirect soll die Zieladresse und bei Basic Authentication das Realm gespeichert werden. Die HTML-Datei selbst soll ebenfalls gespeichert werden. Die gespeicherten HTML-Dateien sollen relativ simpel dahingehend untersucht werden, ob Hinweise auf Impressumsinformationen und Datenschutzerklärungen in Form von Links vorhanden sind.

3.2.6 Schwachstellenanalyse

Für jede über einen Port bereitgestellte ermittelte Anwendung soll ein Link zu einem passendem Suchergebnis in der National Vulnerability Database bereit gestellt werden, um komfortabel an detaillierte Informationen zu den aktuellen Schwachstellen zu gelangen. Hierfür kann, wenn vorhanden, die Common Platform Enumeration (CPE) genutzt werden, welche ein Standard für eine einheitliche Namensgebung bei informationstechnischen Systemen, Plattformen und Softwareprodukten darstellt. Im Falle, dass keine CPE vorliegt, soll der Produktname der Anwendung für die Suche in der National Vulnerability Database verwendet werden.

3.2.7 Aufbereitung der Daten als Dashboard

Die gewonnenen Informationen sollen aufbereitet und für die Ausgabe bereitgestellt werden. Dies soll in Form eines Dashboards realisiert werden. Hierbei sollen mindestens für jede Domain die Ansprechpartner und deren Kontaktmöglichkeiten gelistet werden, sofern diese als Stammdaten hinterlegt wurden. Wenn vorhanden, sollen in dieser Übersicht das jeweils zugehörige Impressum und die Datenschutzerklärung verlinkt werden. Bei weiterem Interesse soll es möglich sein, Detailinformationen zu dem angefragten Hostnamen zu erhalten.

3.2.8 Statistische Auswertungen

Der Fokus bei der Dashboard-Ansicht soll auf die kritisch eingestuften Sachverhalte gesetzt werden und die Schlüsselinformationen sollen möglichst kompakt und geeignet dargestellt werden. In der Dashboard-Übersicht sollen die nachfolgend aufgelisteten statistischen Auswertungen in Form von Diagrammen oder anderen angemessenen Möglichkeiten erfolgen:

- Verteilung der geprüften Systeme auf aktive und inaktive
- Verteilung der gefundenen Serviceanwendungen
- Verteilung der geprüften Benutzernamen auf die Loginverfahren Passwort und Public-Key
- Verteilung der Betriebssysteme

- Anzahl der Systeme mit und ohne Ansprechpartner
- Anzahl der Domainnamen mit und ohne Ansprechpartner
- Anzahl der Domainnamen mit und ohne Basic Authentication
- Anzahl der Subdomainnamen mit und ohne Ansprechpartner
- Anzahl der Subdomainnamen mit und ohne Basic Authentication
- Anzahl der HTTP-Services
  - mit Impressum und Datenschutzerklärung
  - mit Impressum
  - mit Datenschutzerklärung
  - ohne Impressum und Datenschutzerklärung
- Anzahl der HTTPS-Services
  - mit Impressum und Datenschutzerklärung
  - mit Impressum
  - mit Datenschutzerklärung
  - ohne Impressum und Datenschutzerklärung

3.2.9 Visualisierung der Domainstruktur

Um die hierarchischen Zusammenhänge der Domains zu visualisieren, sollen die Domainnamen und Subdomainnamen besonders kenntlich gemacht werden, so dass zwischen den beiden einfach unterschieden werden kann.

3.2.10 Export der Daten

Es soll möglich sein, die gesammelten Informationen im JSON-Format exportieren zu können.

3.3 Nicht-Funktionale Anforderungen

3.3.1 Modularisierung der Applikation

Die Applikation soll möglichst modular aufgebaut sein, um eine einfache Möglichkeit zur Erweiterung zu bieten. Auch die Wiederverwendbarkeit von einzelnen Komponenten soll dadurch gefördert werden.

3.3.2 Effiziente Verarbeitungsprozesse

Die Verarbeitungsprozesse sollen möglichst effizient gestaltet werden, um die Dauer des Analyseprozesses zu minimieren und Redundanzen im Verarbeitungslauf zu vermeiden.

3.3.3 Integration bestehender Softwarelösungen

Wenn bereits Softwarelösungen existieren, die als Teil der Applikation eingesetzt werden können, um die Anforderungen an die Applikation zu erfüllen, sollten diese genutzt werden, um Ressourcen zu sparen.

3.3.4 Benutzbarkeit

Die Benutzung der Applikation soll intuitiv und einfach sein.

3.3.5 Dokumentation

Die Dokumentation spielt eine besonders wichtige Rolle, da das Projekt durch verschiedene Personen weiterentwickelt werden soll.

4 Konzeption

Nachdem die Anforderungen an die Applikation aufgestellt wurden, wird in diesem Abschnitt zunächst die Systemarchitektur vorgestellt, die Anwendung in einen Benutzerkontext gesetzt und nachfolgend der Analyseprozess näher betrachtet.

4.1 Systemarchitektur

Die Applikation wird als Webapplikation nach dem Client-Server-Modell realisiert, denn dieses Modell eignet sich dazu, Verteilte Systeme zu strukturieren. Wie in Abb. 4.1 zu erkennen ist, nimmt der Dienstnutzer (Client) eine angebotene Funktionalität vom Diensterbringer (Server) über ein Rechnernetz hinweg in Anspruch. Der Client und Server weisen jeweils separate Adressräume auf. Zur Erfüllung des angeforderten Dienstes kann der Server weitere Diensterbringer mit einbeziehen. Somit lässt sich das Client-Server-Modell flexibel auf große, hierarchische Systeme anpassen.[11]

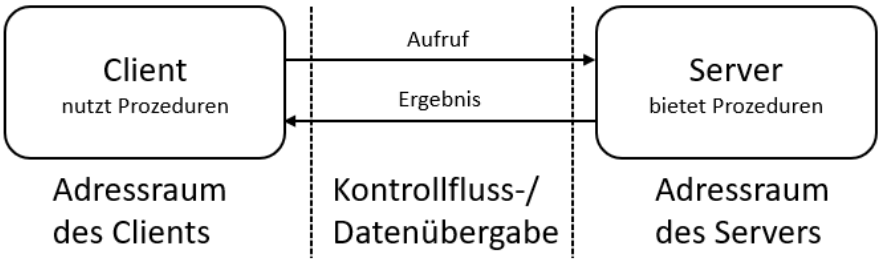

Abbildung 4.1: Client-Server-Modell[15]

Eine der in Kapitel 3 beschriebenen Anforderungen ist, dass eine Erweiterbarkeit bzw. Modifikation der Applikation ohne große Umstände möglich sein soll. Aus diesem Grund bietet sich eine erweiterte Form des Client-Server-Modells an, die 3-Schichten-Architektur (engl. three tier architecture), die in Abb. 4.2 dargestellt ist. Die drei Schichten bilden die

[15]nach Verteilte Systeme - Grundlagen und Basistechnologien von Schill[11]

Präsentationsschicht, die Verarbeitungsschicht und die Persistenzschicht. Die klar definierten Schnittstellen zwischen den einzelnen Schichten machen es möglich, dass einzelne Komponenten einfach ausgetauscht werden können, ohne große Anpassungen an anderen Schichten vornehmen zu müssen.[11]

Die Präsentationsschicht stellt in der 3-Schichten-Architektur die Benutzerschnittstelle der Applikation dar. Neben der reinen Darstellung kann diese Schicht Vorverarbeitungsfunktionalitäten bereitstellen. Die Verarbeitungsschicht beinhaltet die Anwendungslogik der Applikation und ist zuständig für die serverseitigen komplexen Verarbeitungsprozesse. Die Persistenzschicht verwaltet die Zugriffe auf die Datenbestände durch den Server.[11] Die Struktur der 3-Schichten-Architektur unterstützt durch die klare Trennung der Zuständigkeiten das Single Responsibility Principle der objektorientierten Programmierung.

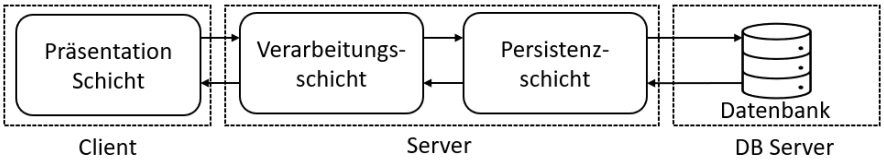

Abbildung 4.2: 3-Schichten-Architektur

4.2 Kontextsicht

Ein Benutzer interagiert mit dem System indem er die Applikation konfiguriert. Es gibt drei Fremdsysteme, die während des Verarbeitungsprozesses angesprochen werden. Zum einen die National Vulnerability Database, die verwendet wird, um die weiterführenden Informationen zu den aktuellen Schwachstellen von Produkten zu erhalten. Zum anderen die Domains einer Organisation, welche während des Verarbeitungsprozesses analysiert werden sollen. Das dritte Fremdsystem ist der Dienst der Seite *https://crt.sh*, welcher im Rahmen des Analyseprozesses genutzt wird, um die Subdomains möglichst unaufdringlich zu ermitteln. Dargestellt ist der Kontextrahmen der Applikation in der Abb. 4.3.

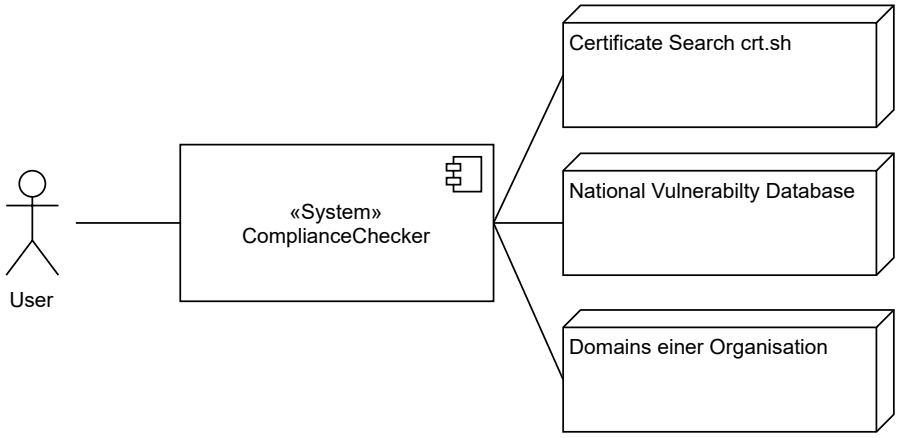

Abbildung 4.3: Kontextsicht

4.3 Analyseprozess

Um die Applikation zu nutzen, wird das System mit den Werten für die aktuelle Prüfung konfiguriert. Danach kann der Analyseprozess gestartet werden. Als erstes werden die aktiven Systeme im Netzwerk mittels Pingcheck identifiziert und es wird versucht, den Hostnamen zu ermitteln. Anschließend werden alle Domains und Subdomains ermittelt. Nun wird für alle aktiven Systeme ein Analyseprozess gestartet, der die jeweils offenen Ports und dazu gehörenden Servicebanner ermittelt. Im Anschluss werden in Abhängigkeit der ermittelten Services die jeweils servicespezifischen Analysen gestartet. Nachfolgend sind die möglichen servicespezifischen Analysen und deren Ergebnisse kurz zusammengestellt.

1. **SSH Service**
   Für die Liste der konfigurierten Benutzernamen wird geprüft, ob das Passwort- oder Public-Key-Verfahren verwendet wird.

2. **SMTP Service**
   Die entsprechenden MX-Records werden ermittelt.

3. **HTTP/HTTPS Service**
   Kann eine HTML-Datei heruntergeladen werden, wird der Inhalt der Seite nach Datenschutzerklärungs- und Impressumsinformationen durchsucht. Zusätzlich wird auf ein Redirect und das Basic Authentication-Verfahren getestet.

Abschließend werden die Ergebnisse persistiert, um eine spätere Bereitstellung zu ermöglichen. Dargestellt ist der Prozess in der Abb. 4.4.

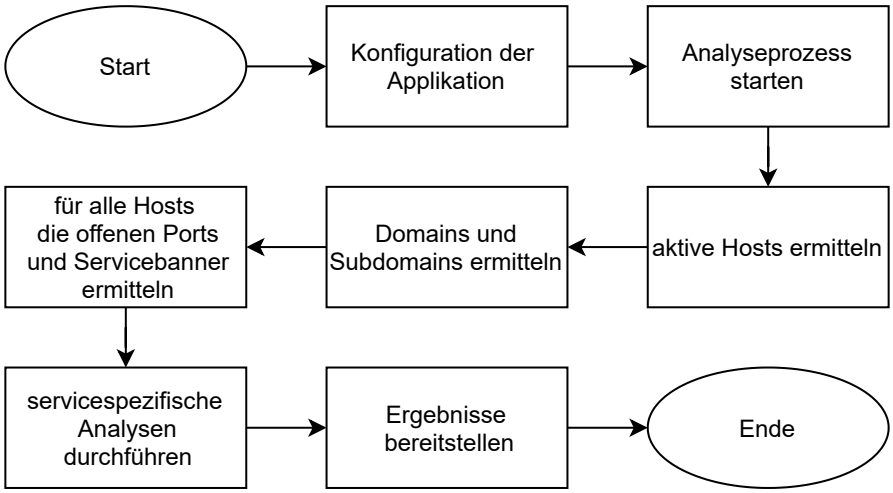

Abbildung 4.4: Analyseprozess

5 Implementierung

Nachdem die Konzeption abgeschlossen ist, wird in diesem Kapitel die eigentliche Implementierung in den Fokus genommen und beschrieben. Wie der Abbildung zur Drei-Schichten-Architektur 4.2 zu entnehmen ist, gliedert sich die gesamte Anwendung in einen Clientteil und einen Serverteil.
Zunächst wird die Implementierung des Serverteils thematisiert. Hier werden im Detail zuerst die im Serverteil eingesetzten Technologien genannt, dann die Struktur der Applikation vorgestellt. Im Anschluss wird die Serviceschicht betrachtet, um anschließend die Implementierung der einzelnen Module aufzuzeigen.
Bezüglich des Clientteils werden die eingesetzten Technologien genannt, die Lösung für das Zustandsmanagement vorgestellt, um abschließend die Benutzeroberfläche zu thematisieren.

5.1 Serverteil

5.1.1 Auswahl der eingesetzten Technologien

Die Umsetzung sowohl der Verarbeitungsschicht als auch der Persistenzschicht wird laut Rahmenbedingung in Python unter Verwendung des Frameworks Django[16] realisiert. Django unterstützt mit seiner Funktionalität bei Zugriffen auf die Datenbank, der Modellierung von Datenmodellen, der Pflege der Datenbanktabellen durch Migrationsanweisungen und mit einem Testkit bei der Überprüfung der Funktionalität. Zusätzlich wird die Erweiterung, das Django Rest Framework[17] (DRF), zum Einsatz kommen. Das DRF ermöglicht es, komfortabel eine auf REST basierende Applikation zu entwickeln und bildet die Schnittstelle zwischen Client und Server hervorragend ab.

[16] https://www.djangoproject.com/
[17] https://www.django-rest-framework.org/

Zusätzlich wird das Tool „Nmap: the Network Mapper“[18] eingebunden, welches für das Projekt von hoher Relevanz ist. Denn Nmap bildet viele Funktionalitäten ab, die im Rahmen der Entwicklung erforderlich sind. Dazu gehört das Ermitteln von offenen Ports, die Analyse von Ports und die Beschaffung von Servicebannerinformationen.
Um das Durchsuchen von HTML-Seiten nach bestimmten Inhalten zu ermöglichen, wird auf die Bibliothek „Beautiful Soup“[19] zurückgegriffen. Das Tool verfügt über eine gute Dokumentation und ist leicht in das Projekt zu integrieren.
Zur dauerhaften Speicherung der Ergebnisse wird die MySQL-Datenbank MariaDB[20] eingesetzt. MariaDB ist eine weit verbreitete, frei verfügbare relationale Datenbank, die über eine ausführliche HTML-Dokumentation verfügt und durch das Framework Django optimal unterstützt wird.

5.1.2 Paketsicht

Wie in Abb. 5.1 ersichtlich ist, wird der Serverteil in vier Pakete unterteilt. Es wird eine Struktur geschaffen, welche bei der Kapselung der Funktionalität unterstützen und die Übersichtlichkeit verstärken soll.

Das Paket `analysis` bildet die zentrale Rolle des Verarbeitungsprozesses ab. Zur Erfüllung des Zweckes werden die Funktionalitäten der Pakete `basedata`, `configurations` und `tools` genutzt, welche nachfolgend im Detail vorgestellt werden. In Django werden die Pakete als Apps realisiert, die ihre eigene Strukturen enthalten. So verfügt jede App über die folgenden Dateien: `models.py` für Datenmodelle, `serializers.py` für Serializers, `urls.py` in Verbindung mit `views.py` für REST-Endpunkte und zwei Konfigurationsdateien `admin.py` und `apps.py`. Aufgrund der Komplexität der Applikation wurden die `models.py`, `serializers.py`, `urls.py` und `views.py` nicht als einzelne Dateien sondern als gleichnamige Ordner umgesetzt, die jeweils dann mehrere separate Dateien beinhalten, um die Übersichtlichkeit und Wartbarkeit zu unterstützen. Die Konfigurationsdatei `admin.py` steuert die Sichtbarkeit der jeweiligen App im internen Adminbereich und die `apps.py` wird für die Einbindung in die Hauptapplikation über die Datei `settings.py` benötigt.

[18] https://nmap.org/
[19] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[20] https://mariadb.org/

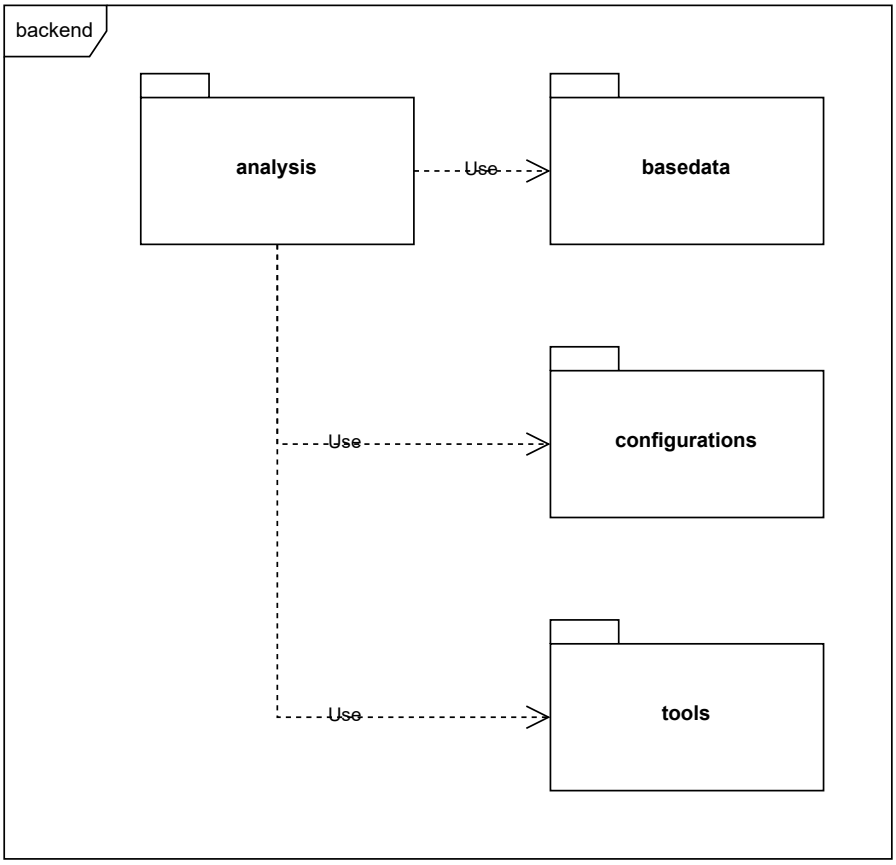

Abbildung 5.1: Paketsicht: Serverteil

5.1.3 Serviceschicht

In diesem Abschnitt werden die Implementierungsdetails der Serviceschicht im Paket `analysis` thematisiert, welche für eine Djangoapplikation eine Besonderheit darstellt. Normalerweise erfolgt die Kommunikation mit der Datenbank innerhalb des Paketes ausgehend von den Views über die Models. Wie in Abbildung 5.2 dargestellt ist, wird im Paket `analysis` eine Serviceschicht zwischen die Views und Models integriert, die darüber hinaus für die Kommunikation mit dem Paket `tools` zuständig ist. Dadurch wird zum einen erreicht, dass die Views und Models nur die notwendigen Funktionalitäten beinhalten und nicht die Geschäftslogik. Zum anderen wird die Schnittstelle zur Kommunikation mit dem Paket `tools` klar durch die Serviceschicht abgebildet. Implizit erreicht man zusätzlich, dass die intern verwendeten Datenstrukturen auch nicht an die Tools übergeben werden müssen, was eine Förderung der Kapselung und der Wiederverwendbarkeit zur

Folge hat.
Im Rahmen des Projektes sind die folgenden fünf Services entstanden:

1. Der `ScanService` wird bei einem neu gestarteten Analyseprozess zuerst angesprochen und erzeugt ein neues `Scan` Objekt, generiert die Http-Antworten und erzeugt die Statistiken des jeweiligen Scanvorgangs.

2. Der `ExplorationService` ist verantwortlich für das Scannen des Netzwerkes und die Ermittlung der aktiven Hosts. Zur Erfüllung dieser Aufgaben wird das Tool `Resolver` eingesetzt, welches im späteren Verlauf vorgestellt wird.

3. Der `DomainDiscoveryService` hat die Aufgabe die Domains der Organisation und die dazugehörigen Subdomains zu ermitteln. Auch dieser Service greift auf Funktionalitäten des `Resolver` Tools zurück, um das Ziel zu erreichen.

4. Der `ServiceDiscoveryService` bildet die Schnittstelle zum `Scanner` Tool. Er ist dafür zuständig, auf aktiven Hostsystemen laufende Services zu identifizieren und Basisinformationen zu sammeln.

5. Der `PortService` hat zur Hauptaufgabe, im Falle eines gefundenen offenen Ports mit laufendem Service, einen spezifischen Analyseprozess zu starten, der Detailinformationen sammelt. Kommuniziert wird mit den Tools `Resolver`, `LoginChecker` und `HTMLParser`.

5.1.4 Tools

Der folgende Abschnitt stellt die einzelnen Tools vor, die im Rahmen des Projektes entwickelt wurden, um eine Übersicht über die Verwendung und deren Funktionalitäten zu geben.

tools.Resolver

Der `Resolver` bildet die folgenden drei Aufgaben ab: Ermitteln aktiver Hosts, Ermitteln von zur Organisation gehörenden Domains und Subdomains sowie das Testen von bestimmten Bedingungen bezüglich einer ermittelten Domain.
Für die **Ermittlung aktiver Hosts** wird basierend auf den im Paket `configurations` hinterlegten IP-Adressbereichen für jede Adresse ein Ping-Check mittels NMAP

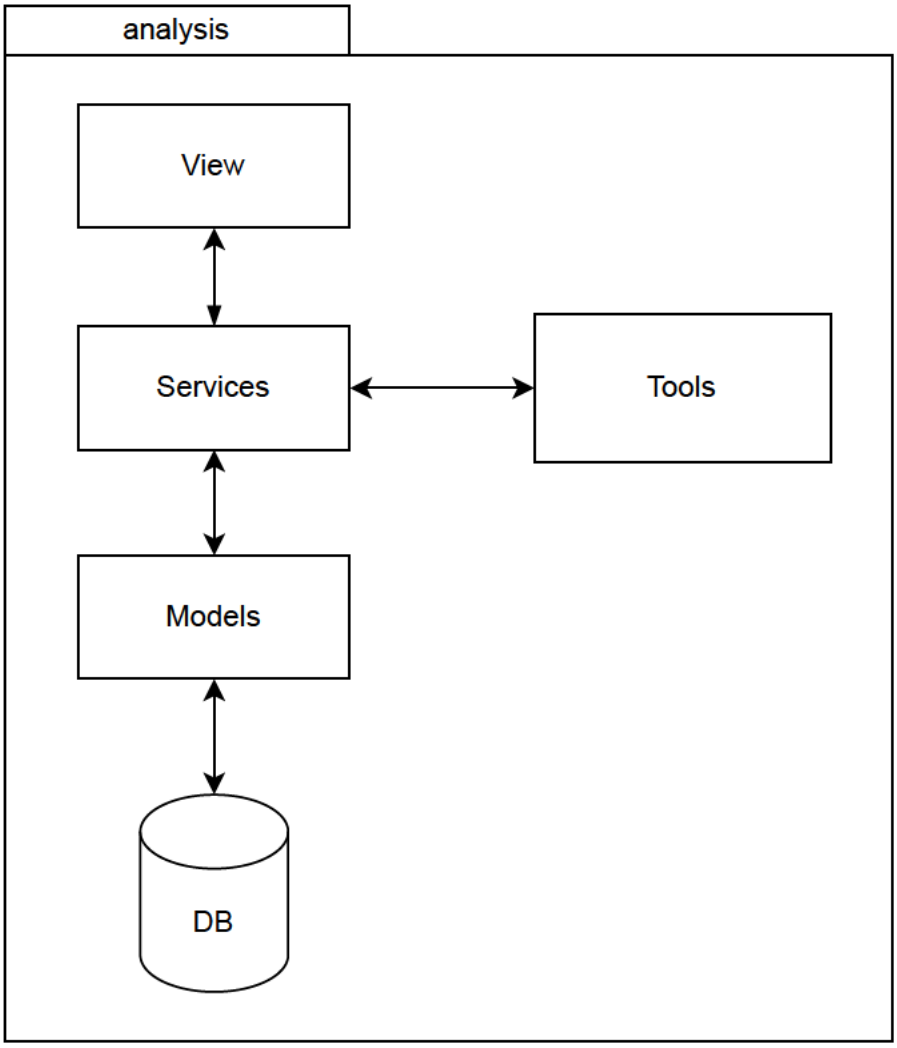

Abbildung 5.2: Serviceschicht Paket: `analysis`

durchgeführt, um die aktiven Hosts zu identifizieren. Im Detail wird ein Python NMAP-Wrapper `python3-nmap`[21] eingesetzt und die Methode `nmap_ping_scan` der Klasse `NmapScanTechniques` verwendet, welche intern den Befehl `nmap -sP <target-ip>` verwendet.

Um die Aufgabe alle **Subdomains und Domains von einer Organisation ermitteln** zu lösen, stellt sich zunächst die Frage, welcher Ansatz verfolgt wird. Die in dieser Arbeit betrachteten Optionen sind der Brute Force-Ansatz, die Nutzung von Suchmaschinen und das Scannen der Certificate-Transparency-Logs. NMAP stellt ein DNS-Brutescript

[21] https://pypi.org/project/python3-nmap/

zur Verfügung, welches es ermöglicht, Subdomains „erraten“ zu können. Da eine der Anforderungen der Arbeit ein gutartiges Verhalten der Applikation ist, kann man diesen Ansatz ausschließen, da Fremdsysteme zu exzessiv genutzt werden und z.B. Überlastungen verursachen können, womit der Ansatz nicht vertretbar wäre. Eine andere Möglichkeit ist es, Suchmaschinen zu nutzen, um die Ergebnisse einer Anfrage auszuwerten und sich die gewünschten Informationen zu extrahieren. Auch dieses Verfahren wird in Hinblick auf die Gutartigkeit ausgeschlossen, da viele Anfragen an Webservices gestellt werden müssen und sich dadurch die gleiche Problematik ergibt. Letztendlich wird in der Applikation deshalb das dritte Verfahren „Scannen der Certificate-Transparency-Logs“ eingesetzt. Bei dem Certificate-Transparency-Log handelt es sich um ein im Jahre 2013 von Google[22] gestartetes Projekt, um das Surfen im Internet sicherer zu machen. Wenn ein Zertifikat durch eine Certificate Authority (CA) ausgestellt wird, wird in dem öffentlichen Certificate-Transparency-Log der entsprechende Eintrag mit den Zertifikatsinformationen eingetragen. Dadurch lässt sich die Historie und der Bestand der ausgestellten Zertifikate nachvollziehen.[6]
Dieses Log wird in der Applikation über den Dienst des Webservices Certificate Search[23] über eine REST-Schnittstelle in Anspruch genommen. Für jede in dem Paket `configurations` hinterlegte Domain einer Organisation wird nur eine Anfrage an den Webservice gestellt. In Kauf genommen wird dabei, dass nicht zertifizierte Domains nicht gefunden werden können, da diese nicht im Certificate-Transparency-Log erfasst werden. Allerdings wird davon ausgegangen, dass die Webseiten einer Organisation in der Regel zertifiziert sind.
Die letzte Funktionalität, die das Tool `Resolver` umfasst, ist **das Testen von bestimmten Bedingungen bezüglich einer ermittelten Domain**. Der `Resolver` bietet die Möglichkeit zu testen, ob eine Domain eine Subdomain einer Domain ist, ob ein Redirect konfiguriert ist und ob das Basic Authentication Verfahren verwendet wird.

tools.Scanner

Die einzige Aufgabe des Scanner Tools ist es, die Servicebanner bei potentiellen offenen Ports zu ermitteln. Hier wird auch auf NMAP zurückgegriffen, um das Ziel zu erreichen. Mit dem Befehl `nmap <target> -oX <output> -sV -sS -sU -p <ports>` ist

[22] https://www.google.de/
[23] https://crt.sh/

es möglich, ein solches Servicebanner zu ermitteln. Für eine Zieladresse `<target>` und eine definierte Menge an Ports `<ports>` wird nun versucht, nähere Informationen über einen eventuell laufenden Service zu erhalten. Der Parameter `-sV` bewirkt das Serviceinformationen gesammelt werden, `-sU` bezieht UDP-Ports in den Scanprozess mit ein und `-sS` steuert dabei das Vorgehen des Scanverfahrens. In diesem Fall wird auf ein TCP-SYN-Scan gesetzt, der schnell und unauffällig ist, da keine TCP Verbindung aufgebaut, sondern lediglich auf die Antwort gewartet wird.[10] Das Problem bei diesem Ansatz besteht darin, dass NMAP für diesen Befehl mit Rootrechten ausgeführt werden muss. Da der Prozess automatisiert abläuft, besteht nicht die Möglichkeit mit einer Passworteingabe das Problem zu lösen. Ein anderer Weg wäre es, die Applikation mit Rootrechten zu betreiben, was grundlegend unerwünscht ist. Eine weitere Option ist es, diesen Dienst in eine separate Applikation auszulagern, die dann mit Rootrechten betrieben wird, um dann per REST die Kommunikation zwischen den beiden Applikationen zu gestalten. Da dieses Verfahren eine höhere Komplexität mit sich bringt, wird auch diese Möglichkeit nicht näher in Betracht gezogen. Die gewählte Lösung ist es, den Befehl ohne Passworteingabe ausführbar zu machen. Hierfür wird das Shellscript `scan_privileged.sh` angelegt und in der Projektstruktur im Ordner `scripts` abgelegt. Das Shellscript enthält lediglich den o.a. NMAP-Befehl. Nun wird ein Visudo-Eintrag erzeugt mit `<Benutzer> ALL=(ALL) NOPASSWD: <path>/backend/scripts/scan_privileged.sh`, der es dem Benutzer erlaubt, bestimmte Befehle ohne Passwort auszuführen. Dabei ist darauf zu achten, dass die Rechte der Datei nach dem Aufsetzen der Applikation an den Benutzer `root` übergeben werden, damit das Skript nicht geändert werden kann und somit jegliche Befehle zur Ausführung gebracht werden können. Die unter `<output>`, in diesem Fall `scripts/output`, abgelegte XML-Datei wird mit dem `NmapParser` der Bibliothek `libnmap` eingelesen und ausgewertet. Der Scanner liefert eine Liste mit den Services samt der dazugehörigen Informationen zurück.

tools.LoginChecker

Der `LoginChecker` kommt zum Einsatz, wenn vom `Scanner` ein laufender SSH-Dienst ermittelt wurde. Die Aufgabe ist es nun, für die im Paket `configurations` hinterlegten Benutzerkonten zu prüfen, ob für ein Benutzerkonto das Public-Key-Verfahren konfiguriert oder der Login mittels Passwort möglich ist. Die erste Option ist es, den Login einfach zu versuchen und das Ergebnis auszuwerten. Da aber die Gutartigkeit der Applikation eine hohe Priorität hat, wird diese Möglichkeit nicht in Betracht gezogen.

Um eine SSH-Verbindung aus der Applikation heraus aufbauen zu können, wird auf die in Python integrierte Bibliothek `subproccess` zurückgegriffen. Mit der Methode `get-statusoutput` wird gepüft, ob die Anmeldung ohne Passwort für den Benutzer möglich wäre. Der Befehl lautet: `ssh <user>@<host> -o PasswordAuthentication=no -o StrictHostKeyChecking=accept-new`.
Wichtig ist, dass bei der Automatisierung der zusätzliche Parameter `StrictHost-KeyChecking` auf `accept-new` gesetzt wird, da sonst die neuen Fingerabdrücke der Systeme manuell bestätigt werden müssen. Daraufhin wird das Ergebnis überprüft und abgeklärt, welche Verfahren unterstützt werden. Wird ausschließlich das Public-Key-Verfahren unterstützt, ist das Ergebnis positiv zu bewerten und es wird `True` zurückgegeben. Andernsfalls wird der Fall negativ bewertet und der Rückgabewert ist dementsprechend `False`.

tools.HTMLParser

Wenn ein laufender HTTP-Service ermittelt wurde, ist es die Aufgabe des `HTMLParser`, auf den Webseiten nach den Links zum Impressum und der Datenschutzerklärung zu suchen und die Informationen zur Verfügung zu stellen. Für die Umsetzung kommt zum einen die Bibliothek `requests`, die als Standard in Python integriert ist und zum anderen die Bibliothek `BeautifulSoup` [24] zum Einsatz. Nachdem mittels `requests` der Inhalt der Seite ermittelt wurde, kann der Quellcode der Seite durch den in `BeautifulSoup` integrierten HTMLParser analysiert werden. Für das Impressum wird nach den Schreibweisen „Impressum“ und „Imprint“ gesucht. Für die Datenschutzerklärung nach „Datenschutzerklärung“, „Datenschutzerklaerung“ und „Datenschutzerklärung“. Rückgabewert ist ein Tupel, welches die URL, den Seiteninhalt und das Suchergebnis für das Impressum und die Datenschutzerklärung beinhaltet.

tools.CSVFileHandler

Die Methode `import_file(path, auto_clear)` des `CSVFileHandler` ist für das Importieren von Kontaktdaten im CSV-Format zuständig. Um die Datei zu lesen wird die Standardpythonbibliothek `csv` eingesetzt, welche einen Reader zu Verfügung stellt, mit dem man die einzelnen Zeilen einer Datei auslesen kann. Die folgende Auflistung stellt das Schema für einen Datensatz eines Kontakts dar.

[24] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- Vorname - Pflichtfeld
- Nachname - Pflichtfeld
- E-Mail-Adresse - Pflichtfeld, einzigartig
- Telefonnummer - Optional
- Kommentar - Optional
- Domains - Mehrfacheingabe durch Verwendung eines Leerzeichen möglich
- IP-Adressbereiche - Mehrfacheingabe Nutzung eines Leerzeichen möglich

Als Parameter werden der Methode der Dateipfad `path` und ein Boolean `auto_clear` übergeben, welches regelt, ob die Datei nach der Verarbeitung gelöscht werden soll. Im Rahmen der Verarbeitung wird eine `Contact`-Instanz und ggf. weitere `DomainResponsibility`- und `IpRangeResponsibility`-Instanzen, welche die Verantwortung eines Kontakts für Domains und IP-Adressbereiche repräsentieren, erzeugt und persistiert.

PortConfigUtil

Als Helfermodul ist das `PortConfigUtil` entstanden, welches nützliche Funktionalitäten im Umgang mit den Portkonfigurationen bietet. Die statische Methode `convert_to_list(port_configs)` erwartet als Parameter ein `QuerySet` und sorgt dafür, dass aus diesem eine sortierte Liste mit Ports (integer) zurückgegeben wird. Die statische Methode `merge_ranges(base_class, standard_class)` wird benötigt, wenn die Standardports der aktuellen Konfiguration der Applikation hinzugefügt werden sollen. So werden die einzelnen Portbereiche der `base_class` mit denen der `standard_class` in den Bereichen, in denen es Überschneidungen gibt, zu einem Bereich zusammengefasst. Eine weitere statische Methode lautet `get_portranges_as_string(port_configs)`. Sie gibt für ein übergebenes `QuerySet` von `PortConfig` einen `String` zurück, welcher die einzelnen Ports durch ein Komma separiert enthält. Die letzte Methode ist `create_port_string()`, welche aus allen konfigurierten `TCPConfigs` und `UDPConfigs` einen für NMAP benötigten `String` des folgenden Formats erzeugt:

$$T : TCPPort_1, ..., TCPPort_N, U : UDPPort_1, ..., UDPPort_N$$

IpRangeUtil

Als letztes Tool ist das Helfermodul `IpRangeUtil` zu nennen, welches mit der Methode `validate_ip_address(ip)` die Möglichkeit zur Validierung einer IPv4-IP-Adresse bietet. Zurückgegeben wird ein `Boolean`.

5.2 Clientteil

5.2.1 Auswahl der eingesetzten Technologien

Im Rahmen dieser Arbeit wird die Präsentationsschicht laut Rahmenbedingungen in JavaScript unter Benutzung des Frameworks React[25] implementiert. Neben der guten Perfomanz von React bietet das Framework ein ausgeprägtes Ökosystem an Bibliotheken. Zusätzlich wird in dieser Arbeit Redux[26] verwendet, welches für die Zustandsverwaltung der Applikation eingesetzt wird. Redux ist eine weit verbreitete Möglichkeit, die Zustandsverwaltung in JavaScript Anwendungen zu handhaben und ist sehr gut skalierbar. Auch die Zustände von komplexen Anwendungen lassen sich mit Redux übersichtlich verwalten.
Um die Kommunikation des Clienteils mit dem Serverteil zu realisieren, wird auf die Bibliothek „axios“[27] zurückgegriffen. Für axios spricht die weite Verbreitung, die gute Integrierbarkeit und die gute Leserlichkeit der HTTP-Requests.
Das Design der Benutzeroberfläche wird mit Hilfe des Bootstrap[28] Frameworks realisiert. Neben der guten Dokumentation spricht auch die einfache Integration in Projekte für Bootstrap. Zusätzlich wird das Template „Spacelab“ von Bootswatch[29] eingebunden, um die Bootstrap Elemente optisch anzupassen und ein minimalistisches schlankes Design zu ermöglichen.
Für die Erstellung von Graphen wird auf die Bibliothek „Google Charts“[30] gesetzt. Diese Bibliothek bietet optisch ansprechende Graphen und die Integration ist leichter als bei Alternativen, die auf dem Markt existieren.

[25] https://reactjs.org/
[26] https://redux.js.org/
[27] https://www.npmjs.com/package/axios
[28] https://getbootstrap.com/
[29] https://bootswatch.com/
[30] https://www.react-google-charts.com/

5.2.2 Zustandsverwaltung

Grundsätzlich basiert React auf dem Prinzip des unidirektionalen Datenflusses. Wie in Abbildung 5.3 dargestellt ist, wird ausgehend von einer Aktion der Zustand einer Komponente verändert. Die Veränderung des Zustandes hat Auswirkungen auf die Ansicht und löst ein erneutes Rendering der betroffenen Komponente aus. Somit verwaltet der Zustand der Komponente alle Werte, die für das Anzeigen der Komponente relevant sind.[1]

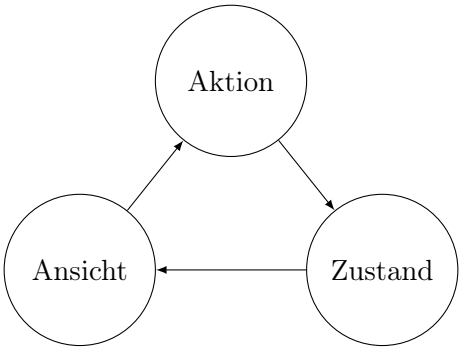

Abbildung 5.3: Datenfluss in React

In Hinblick auf die Komplexität der Anwendung, gerade unter Berücksichtigung der potentiellen Erweiterung durch andere Studierende, ist es notwendig den globalen Zustand der Applikation effizient zu managen. Hier wird auf die relevanteste der auf dem Markt existierenden Bibliotheken gesetzt, Redux. Durch die klaren Strukturen lassen sich neue Komponenten einfach integrieren und zeitgleich wird durch Redux das Single Point of Control Pattern gefördert. Redux basiert auch auf dem Prinzip des unidirektionalen Datenflusses, allerdings wird das Modell um den `Store` und die `Reducer` erweitert. Der `Store` verwaltet dabei den applikationsweiten Zustand und ist somit der einzige Anlaufpunkt, wenn es um den globalen Zustand geht. Dieser stellt die Methoden `dispatch` zum Verändern und `getState` zum Lesen von Daten zu Verfügung. Die `Reducer` wiederum übernehmen die Aufgabe, bei Änderungen den derzeitigen Zustand in den neuen Zustand zu überführen und anschließend an den `Store` zu übergeben. Um dabei zu bestimmen, auf welche Art der aktuelle Zustand verändert werden soll, wird dem Reducer die Art `Type` und der Inhalt `Payload` als Objekt übergeben. Mit diesen beiden Informationen lässt sich dann der neue Zustand bilden.[3] Wie Abbildung 5.4 verdeutlicht, bestimmt der Zustand die Ansicht, welche durch Aktionen den Reducer benachrichtigt.

Dann erhält der `Store` durch den `Reducer` einen neuen Zustand, womit sich der Kreislauf wieder schließt.

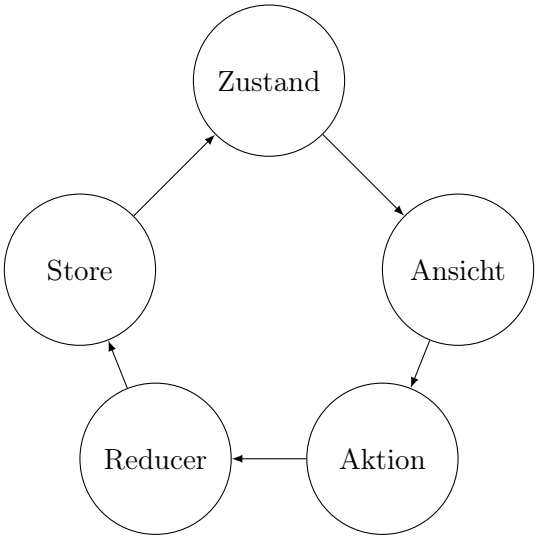

Abbildung 5.4: Datenfluss in React mit Redux

5.2.3 Benutzeroberfläche

Das Design der Applikation soll minimalistisch und funktional sein. Deswegen wurde auf das Bootstrap-Template „Spacelab“ von Bootswatch zurückgegriffen, welches einen solchen Stil unterstützt. Grundsätzlich soll das Design der Benutzeroberfläche der Darstellung eines Dashboards folgen, d.h. Übersichtlichkeit und Informationsgehalt stehen im Fokus. Im Folgenden wird die Umsetzung der Kernelemente der Benutzeroberfläche, die Statistikkomponente und die Hosteintragskomponente näher betrachtet.

Die Statistikkomponente wurde mittels der Bibliothek React-Google-Charts[31] realisiert. Für die wichtigsten Statistiken kommen Kuchendiagramme zum Einsatz, wie in Abbildung 5.5 dargestellt. Die vier Statistiken sind die Hosts mitsamt ihres Status, die gefundenen Services, die SSH-Statistiken und die Verteilung der Hosts auf die Betriebssysteme. Um die Übersichtlichkeit zu fördern, wurde auf weitere Diagramme verzichtet. Stattdessen wurden die übrigen Informationen in Tabellen strukturiert. Zusätzlich umfassen die Statistiken eine Aufteilung zu den Hosts mit und ohne Ansprechpartner, den Domains- und Subdomains mit und ohne Ansprechpartner und zu den HTTP- und

[31] https://react-google-charts.com/

HTTPS-Webservices. Bei den HTTP- und HTTPS-Webservices wird gezeigt, ob auf den Webseiten Impressum und/oder die Datenschutzerklärung vorhanden sind.

Abbildung 5.5: Statistiken

Bei der Hosteintragskomponente soll auch die Übersichtlichkeit im Fokus stehen ohne dabei an Informationsgehalt zu verlieren. Aufgrund der Menge an Informationen ist es nicht ganz einfach, dabei die Übersichtlichkeit zu wahren. Letztendlich werden Icons samt Tooltips eingesetzt, um den Informationsgehalt ohne Verluste transportieren zu können. In der Abbildung 5.6 sieht man einen beispielhaften Eintrag für ein Hostsystem. In der Spalte Hosts sind die Basisinformation zu den Systemen vorhanden. Dazu gehören die IP-Adresse, der Hostname, das Betriebssystem und der verantwortliche Ansprechpartner, der in diesem Fall nicht vorhanden ist. Unter der Spalte Services finden sich die Informationen zu allen zur Zeit aktiven Services wieder. Auf der linken Seite finden wir den Eintrag zu Protokoll, Port und Servicetyp. Darunter wird der Produktname, optional die Produktversion und die optionalen CPE-IDs gelistet. Der rechte Teil der Spalte sieht je nach Art des Services anders aus.

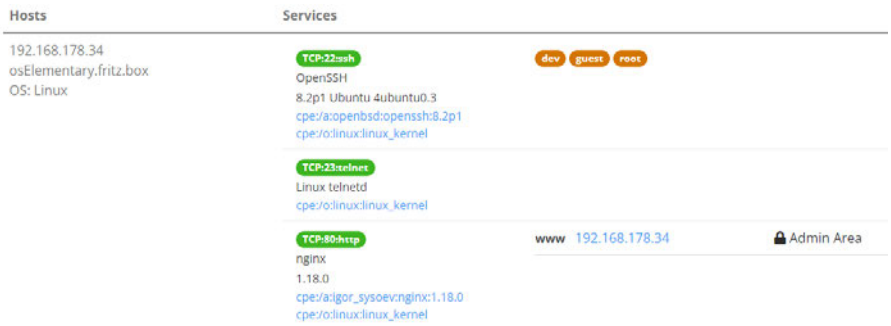

Abbildung 5.6: Hosteintrag

In der Beispieldarstellung 5.6 sehen wir oben die Darstellung eines SSH Tests. Es wird geprüft, ob der Login auf dem Server mit verschiedenen Accounts mittels Passwort oder Public-Key-Verfahren möglich ist. Der Accountname in grün spiegelt wider, dass das sichere Public-Key-Verfahren konifguriert ist, wobei orange dafür steht, dass der Benutzer unbekannt oder ein Login per Passwort möglich ist. Im unteren Fall wird erkenntlich, dass ein HTTP-Webservice läuft, der keiner Domain zugeordnet ist und eine durch Basic Authentication geschützte Seite zur Verfügung stellt, wobei der Realmname Admin Area lautet. Durch Einhaltung der Gestaltprinzipien, wie z.B. Gruppierung, Einsatz von Icons, Farbgebungen und dem Prinzip der Geschlossenheit, kann gerade in dieser Ansicht eine gute Übersichtlichkeit erhalten werden, ohne dabei an Informationsgehalt zu verlieren.

6 Qualitätssicherung

Nachdem die Implementierungsdetails der Applikation erläutert wurden, widmet sich dieses Kapitel dem Thema der Qualitätssicherung. Zuerst wird die zu Grunde liegende Testumgebung, dann das Testverfahren im Rahmen des Serverteils und abschließend das Testverfahren des Clientteils vorgestellt.

6.1 Testumgebung

Um die Applikation während der Entwicklung testen zu können, ohne produktiv eingesetzte Systeme zu stören oder zu behindern, wurde mit einer Testumgebung gearbeitet. Die Testumgebung fällt im Rahmen dieser Arbeit nur sehr beschränkt aus, da nur begrenzte Ressourcen zur Verfügung stehen. Neben dem Entwicklungssystem, auf dem die Anwendungen und die Datenbank lokal betrieben werden, existieren zwei weitere Systeme, die das Netzwerk bilden. Zum einen handelt es sich um ein Linuxsystem, welches im lokalen Netzwerk integriert ist. Auf diesem System wurden verschiedene Services installiert, um die Funktionalitäten der Software testen zu können. Es wird ein OpenSSH[32] Dienst, ein Telnetdienst und ein Webserver mit Nginx[33] ohne Domain betrieben. Zum anderen wird auf einen Rootserver zurückgegriffen, der bereits als Webserver eingesetzt wird. Dieser wurde um spezifische Testwebseiten erweitert, die in der Tabelle 6.1 näher erläutert werden. Hier kommen SSL-Zertifikate zum Einsatz und der Webserver bietet seine Dienste über Port 80 und 443 an. Neben einem OpenSSH Dienst wird noch ein SMTP-Dienst mit Postfix[34] betrieben.

[32] https://www.openssh.com/
[33] https://www.nginx.com/
[34] http://www.postfix.org/

Subdomain	Zweck
ba-test.reimsen.de	Webseite mit Basic Authentication zum Testen des Tools `Resolver`
play.reimsen.de	Webseite mit Links zu Impressum und Datenschutzerklärung zum Testen des Tools `HTMLParser`
redirect-test.reimsen.de	Webseite mit aktiver Weiterleitung zum Testen des Tools `Resolver`

Tabelle 6.1: Aufstellung der Testwebseiten

6.2 Serverteil

Um die nötige Qualität der Serverapplikation sicherzustellen wurde auf Unittests mit Hilfe der Pythonstandardbibliothek `unittest` gesetzt. Organisiert wurden die Tests in einem Ordner `tests` und die Dateien müssen der Namenskonvention `test_*.py` entsprechen, um gefunden zu werden. Um eine Einschätzung über die Testabdeckung zu erhalten, wird die Bibliothek `coverage`[35] eingesetzt. Diese ermöglicht es eine HTML-Auswertung in den Ordner `htmlcov` zu erzeugen, welche eine Aufschlüsselung über die abgedeckten Codezeilen liefert. Derzeit liegt die gesamte Testabdeckung bei 95%. Der Fokus liegt dabei auf den kritischen Teilen der Applikation wie den Tools und den Services. Ein gewisser Anteil wird allerdings von Django automatisch mitgetestet.

6.3 Clientteil

Die Funktionalität des Clientteils wurde durch Benutzertests überprüft. Es wurde versucht durch unübliche Eingaben oder Klickstrecken, Konstellationen zu erzeugen, die das Programm in einen Fehlerzustand überführen. Da das erwartete Verhalten bekannt ist, konnten so Diskrepanzen zwischen dem Soll- und Istzustand aufgedeckt werden.

6.4 Bereitstellung der Applikation

Im Rahmen eines Projektes eines anderen Studenten, der sein Masterstudium absolviert, soll die Applikation für eine produktive Umgebung bereitgestellt werden. Die Bereitstel-

[35] https://pypi.org/project/coverage/

lung soll über Docker[36] erfolgen, da so eine einfache Integration in bestehende Systemstrukturen sichergestellt und die Unabhängigkeit von Softwareanforderungen gewährleistet ist. Sowohl die Clientanwendung, Serveranwendung und das Datenbanksystem laufen jeweils in einem seperaten Dockercontainer. Die Kommunikation zwischen den Containern wurde mit Docker Compose[37] realisiert.
Im Rahmen der Bereitstellung sind Fehler aufgedeckt worden, die in einer lokalen Testumgebung verborgen blieben. Nach Rücksprache konnten die Fehler behoben werden und die Bereitstellung der Applikation war erfolgreich.
Außerdem wurde durch diesen Test die Erkenntnis gewonnen, dass im Rahmen einer produktiven Umgebung innerhalb eines Dockercontainers die Anwendungen keinen expliziten `sudo` Aufruf benötigen. Damit wird das im Abschnitt 5.1.4 unter dem Punkt „tools.Scanner“ beschriebene Berechtigungsproblem obsolet.
Aber auch durch die Bedienung eines weiteren Benutzers, der in den Entwicklungsprozess selbst nicht eingebunden ist, kamen neue Fehler an die Oberfläche, die durch Ergänzung von zusätzlicher Validierung in der Präsentationsschicht gelöst werden konnten.

6.5 Dokumentation

Gute Software erfordert eine gute Dokumentation, um unter anderem Entwicklern den Einstieg in das Projekt zu erleichtern. Um eine gute Dokumentation zu gewährleisten und eine komfortable Möglichkeit zu bieten, diese zu pflegen, wird auf eine automatisch erzeugte HTML-Dokumentation gesetzt. Als Bilbiothek wurde auf Sphinx[38] zurückgegriffen. Dieses Tool erzeugt auf Basis von `docstrings` eine HTML-Dokumentation. In der Serveranwendung im Ordner `_docs` befinden sich die relevanten Konfigurationsdateien. Im Unterordner `_build` liegen dann die generierten Dokumentationen. Sphinx ist einfach in das Projekt zu integrieren und bietet mit dem Read-the-Docs-Theme[39] eine optisch ansprechende und übersichtlich gestaltete Oberfläche. Für alle Tools, Views und Services wurden `docstrings` erstellt, die später in die Dokumentation einfließen. Die Dokumentation selbst ist als passwortgeschützter Bereich in der Webapplikation über die Schaltfläche „Dokumentation“ zu erreichen.

[36] https://docs.docker.com/
[37] https://docs.docker.com/compose/
[38] https://www.sphinx-doc.org/en/master/
[39] https://sphinx-rtd-theme.readthedocs.io/en/stable/

7 Evaluation

Nach dem Thema Qualitätssicherung des Projektes widmet sich dieses Kapitel der Evaluation der Ergebnisse der Arbeit. Zuerst werden die im Laufe des Projektes entstandenen Probleme thematisiert, um anschließend zu überprüfen, ob die ursprünglich gesetzten Ziele erreicht wurden.

7.1 Probleme

Am Anfang des Projektes war die Erfahrung mit Python und Django nur begrenzt, so dass die Umsetzung relativ schleppend voranging und teilweise suboptimale Lösungen entwickelt wurden. Durch das intensive Auseinandersetzen mit der Programmiersprache und dem Framework wurden später neue Erkenntnisse gewonnen, die dann zu einer Überarbeitung der bereits bestehenden Implementierung geführt haben.
Viele Probleme ergaben sich auch aus der Sensibilität des Themas IT-Sicherheit und des damit verbundenen Schwachstellenmanagements. Die Informationsbeschaffung über den Zustand eines Systems zu gutartigen Zwecken, z.B. die offenen Ports und Detailinformationen zu laufenden Anwendungen, liegt dicht bei den Vorbereitungen für einen Angriff auf ein System.
Demnach gestaltete es sich als unmöglich, während der Implementierung Tests an produktiv laufenden Systemen vorzunehmen. Das Problem konnte durch den Aufbau einer Testumgebung zwar gelöst werden, war aber mit viel Aufwand verbunden, da das Vorwissen zu den einzelnen Serviceanwendungen, z.B. Mail-Server, begrenzt war. Demnach musste eine Einarbeitung in Themengebiete erfolgen, die für die Lösung der eigentlichen Fragestellung nur nebensächlich notwendig war.
Aber selbst die eigens verwalteten Systeme stuften die Scans im frühen Stadium der Entwicklung als bösartig ein, was zur Folge hatte, dass Sicherheitsmaßnahmen griffen. Zum Beispiel wurde die lokale IP-Adresse des Entwicklungsystems nach einem Scanvorgang von dem Zugriff auf den Testserver ausgeschlossen.

Auch bei der Nutzung von Drittanbieter Services, hier die Nutzung des Services von Certificate Search (crt.sh), traten ähnliche Probleme auf. Während der Testphase des Tools `resolver` wurden die Dienste zu häufig in Anspruch genommen, so dass Anfragen der IP-Adresse des lokalen Entwicklungssystems abgelehnt wurden. Vermutlich wurde seitens Certificate Search eine Einstufung als DDoS-Attacke vorgenommen und das Systems hat sich dagegen abgesichert. In der weiteren Entwicklung wurde mit einem fest hinterlegten Ergebnis gearbeitet, um diese Art von Problem zu vermeiden.

Die Verwendung der Bibliothek `python3-nmap`[40] war nicht so gut einsetzbar wie angenommen. Das Problem entstand durch die Automatisierung des kompletten Scanprozesses, der keine Benutzerinteraktion für eine Passworteingabe vorsieht. Bestimmte Befehle sind aber nur mit Rootrechten ausführbar und somit konnten diese Befehle mit `python3-nmap` nicht ausgeführt werden. Es stellte sich heraus, dass Nmap auch sehr gut direkt über die Anwendung mittels der internen Pythonbibliothek `subprocess` benutzt werden kann und die in Abschnitt 5.1.4 beschriebene Lösung für das Berechtigungsproblem eingebunden werden konnte.

Zusammenfassend lässt sich sagen, dass das Projekt sehr viele Detailprobleme enthielt, die es spezifisch zu lösen galt. Auch an dieser Stelle war eine Auseinandersetzung mit vielen Teilgebieten notwendig, da der damalige Wissensstand nicht ausreichte, um die Probleme direkt zu durchdringen und zu lösen. Durch die am Anfang erwähnte Sensibilität des Themas, gestaltete sich die Informationsbeschaffung zu den einzelnen Problemen als mühsam.

7.2 Fazit

Ziel der Arbeit war es, eine Applikation zu entwickeln, die automatisiert das Sicherheitsniveau einer Organisation ermittelt und die Informationen im Rahmen einer Webapplikation bereitstellt. Die in Kapitel 3 gestellten funktionalen Anforderungen sind vollumfänglich umgesetzt und die Applikation lässt sich in einer Dockerumgebung bereitstellen.
Eine der in Kapitel 3 gestellten nicht-funktionalen Anforderung, Effiziente Verarbeitungsprozesse, ist nur teilweise erfüllt. Es wurde bei der Umsetzung zwar darauf geachtet, keine redundanten Arbeiten zu erledigen, aber es wurde kein Fokus auf Parallelisierung bzw.

[40] https://pypi.org/project/python3-nmap/

Multi-Threading gelegt.
Abschließend lässt sich sagen, dass ein solider Grundstein für eine Weiterentwicklung durch weitere Studierende gelegt wurde. Der Einsatz in der HAW ist geplant und soll durch einen Masterstudenten in das Informationssicherheitsmanagement integriert werden.
Die Umsetzung des Projektes war herausfordernd und interessant. Auch der Hintergrund, dass die Software produktiv eingesetzt werden könnte, sorgte für die Motivation eine optimale Lösung zu finden. Die Kollaboration mit dem Masterstudenten hat die Gesamterfahrung bereichert und für einen kritischen Austausch gesorgt.

8 Ausblick

Nachdem die Evaluierung vorgenommen wurde, wird in diesem Abschnitt ein Ausblick gegeben, wie die Applikation erweitert und verbessert werden könnte.

Der Kern der Applikation ist es, das Sicherheitsniveau einer Organisation zu analysieren und darzustellen. Bisher ist es möglich, über das Produkt zu den produktspezifischen Einträgen von Schwachstellen bei der National Vulnerability Database zu gelangen. In späteren Versionen kann ein Zuordnen von erkannten Produkten auf konkrete Schwachstellen einen erheblichen Mehrwert für den Anwender generieren, da die manuelle Tätigkeit auf ein Minimum beschränkt werden kann. Die Informationen könnten direkt über die vom National Institute of Standards and Technology bereitgestellte REST-Schnittstelle abgefragt und innerhalb der eigenen Anwendung dargestellt werden.

Die Applikation kann mehr Information sammeln, wenn man die Anwendung derart erweitert, so dass weitere spezifische Prüfungen im Falle eines gefundenen Services ausgeführt werden. Als relevante Services wären an diesem Punkt IMAP- und POP-Services zu nennen.

Betrachtet man die Applikation hinsichtlich der Performanz eines Durchlaufs, wird deutlich, dass Verbesserungspotential vorhanden ist. In dieser ersten Version wurde das Thema Parallelisierung zunächst vernachlässigt. Python bringt aber alle Voraussetzungen mit, um mit Multiprocessing die Performanz der Applikation zu verbessern. Großes Optimierungspotential besteht in dem Schritt der Serviceanalyse der gefundenen Hostsysteme. Hier kann man mehrere Hostsysteme parallel scannen und damit die erforderliche Zeit eines Scanprozesses drastisch reduzieren.

Ein weiterer Verbesserungsansatz bezieht sich auf die Robustheit der Applikation. Grundsätzlich sollte mehr Validierung vorgenommen und in Fehlerfällen sollte ein Aufräumprozess gestartet werden, der die Daten von ggf. schon geschriebenen Daten bereinigt und dafür sorgt, dass die Datenbank sich stets in einem konsistenten Zustand befindet.

Auch im Clientteil können „Quality of Life“-Features einen deutlichen Mehrwert für die Benutzererfahrung bringen. Zum Beispiel können Filteroptionen bei der Kontaktansicht das Pflegen der Kontakte effizienter gestalten, da man eine gezielte Suche vornehmen kann und nicht gezwungen ist, bis zum gesuchten Ziel zu scrollen. Für die Kontaktansicht und die Ergebnisansicht sollten die Ergebnisse auf verschiedene Seiten verteilt werden - Stichwort Pagination - um lange Listen zu vermeiden, die sich ggf. negativ auf die Performanz der Benutzeroberfläche auswirken können.

Ein weitere Verbesserungsmöglichkeit besteht in der Information des Benutzers zu einem laufenden Scanprozess. Es könnte die zu erwartende Restzeit ausgegeben werden, um besser einschätzen zu können, wann der Scanvorgang beendet sein wird. Zusätzlich könnten bereits nach dem Beenden eines Teilschrittes des Analyseprozesses, Informationen an den Benutzer zurückgegeben werden. Auch Fehler die während eines Scanvorganges auftreten, sollten in der Benutzeroberfläche klar ersichtlich sein.

In späteren Versionen ist es denkbar, auch die Benutzeroberfläche automatisiert zu testen. Eine Möglichkeit würde Selenium[41] darstellen, welches auch in Python integrierbar ist.

[41] https://pypi.org/project/selenium/

Literaturverzeichnis

[1] Abramov, Dan ; authors the Redux documentation: *Redux - A predictable state container for JavaScript apps.* – URL https://redux.js.org/. – besucht am 30.01.2022

[2] Berlin, TU: *Informationen zu eingeschränkten IT-Services an der TU Berlin.* – URL https://www.tu.berlin/themen/einschraenkung-it-services/. – besucht am 30.01.2022

[3] Bieh, Manuel: *React lernen und verstehen.* – URL https://lernen.react-js.dev/. – besucht am 30.01.2022

[4] Corporation, The M.: *Documents and Guidance.* 03 2021. – URL https://cve.mitre.org/about/documents.html#cve_list. – besucht am 30.01.2022

[5] Eckert, Claudia 1.: *IT-Sicherheit Konzepte - Verfahren - Protokolle.* 10. Auflage. De Gruyter Oldenbourg, 2018 (De Gruyter Studium). – URL https://doi.org/10.1515/9783110563900

[6] Google: *Certificate Transparency Project - Working together to detect maliciously or mistakenly issued certificates..* – URL https://certificate.transparency.dev/. – besucht am 30.01.2022

[7] Incident Response, FIRST F. of ; Teams, Security: *Common Vulnerability Scoring System version 3.1 User Guide Revision 1.* https://www.first.org/cvss/v3-1/cvss-v31-user-guide_r1.pdf. 2021. – besucht 30.01.2022

[8] Krempl, Stefan: *Cyberangriff: TU Berlin rechnet mit monatelangen IT-Einschränkungen.* 06 2021. – URL https://www.heise.de/news/Cyberangriff-TU-Berlin-rechnet-mit-monatelangen-IT-Einschraenkungen-6061688.html. – besucht am 30.01.2022

[9] Luber, Stefan: *Definition Exploit (Ausnutzung von Schwachstellen)*. 06 2017. – URL https://www.security-insider.de/was-ist-ein-exploit-a-618629/. – besucht am 30.01.2022

[10] Lyon, Gordon: *Nmap security scanner - documentation*. – URL https://nmap.org/docs.html. – besucht am 30.01.2022

[11] Schill, Alexander ; Springer, Thomas (Hrsg.): *Verteilte Systeme Grundlagen und Basistechnologien*. 2. Aufl. 2012. Springer Berlin Heidelberg, 2012 (eXamen.press). – URL http://dx.doi.org/10.1007/978-3-642-25796-4

[12] Sowa, Aleksandra 1.: *Management der Informationssicherheit Kontrolle und Optimierung*. Springer Vieweg, 2017 (Studienbücher Informatik). – URL http://dx.doi.org/10.1007/978-3-658-15627-5

[13] Steinke, Michael: *Schwachstellenmanagement in Hochschulnetzen am Beispiel des Münchner Wissenschaftsnetzes*. Geschwister-Scholl-Platz 1, 80539 München, Universität München, Diplomarbeit, Dezember 2015

[14] Together, FIRST Improving S.: *Common Vulnerability Scoring System version 3.1 Specification Document Revision 1*. https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf. 2021. – besucht 30.01.2022

[15] Wilkens, Andreas: *Angriff auf IT der TU Berlin: Daten sind abgeflossen*. 05 2021. – URL https://www.heise.de/news/Angriff-auf-IT-der-TU-Berlin-Daten-sind-abgeflossen-6050753.html. – besucht am 30.01.2022

[16] Wilkens, Andreas: *IT-Dienste der TU Berlin angegriffen*. 05 2021. – URL https://www.heise.de/news/IT-Dienste-der-TU-Berlin-angegriffen-6034861.html. – besucht am 30.01.2022

A Anhang

A.0.1 Common Vulnerability Scoring System - Metriken

Tabelle A.1: CVSS - Base Metrics Group

		Metrik	Wertebereich
Base Metric Group	Exploitability	Attack Vector (AV)	**Network (N)**
			Adjacent (A)
			Local (L)
			Physical (P)
		Attack Complexity (AC)	**Low (L)**
			High (H)
		Privileges Required (PR)	**None (N)**
			Low (L)
			High (H)
		User Interaction (UI)	**None (N)**
			Low (L)
			High (H)
		Scope (S)	**Changed (C)**
			Unchanged (U)
	Impact	Confidentiality Impact (C)	**High (H)**
			Low (L)
			None (N)
		Integrity Impact (I)	**High (H)**
			Low (L)
			None (N)
		Availability Impact (A)	**High (H)**
			Low (L)
			None (N)

zu 2.3.2 Base Metric Group

Tabelle A.2: CVSS - Temporal Metrics Group

		Metrik	Wertebereich
Temporal Metric Group		Exploit Code Maturity (E)	**High (H)**
			Functional (F)
			Proof-of-Concept (P)
			Unproven (U)
			Not Defined (X)
		Remediation Level (RL)	**Unavailable (U)**
			Temporary Fix (T)
			Official Fix (O)
			Not Defined (X)
		Report Confidence (RC)	**Confirmed (C)**
			Reasonable (R)
			Unknown (U)
			Not Defined (X)

zu 2.3.2 Temporal Metric Group

Tabelle A.3: CVSS - Environmental Group Metrics

		Metrik	Wertebereich
Environmental Metric Group	Base Modifiers	Attack Vector (AV)	Not Defined (MAV:X)
			Network (MAV:N)
			Adjacent (MAV:A)
			Local (MAV:L)
			Physical (MAV:P)
		Attack Complexity (AC)	Not Defined (MAC:X)
			Low (MAC:L)
			High (MAC:H)
		Privileges Required (PR)	Not Defined (MPR:X)
			None (MPR:N)
			Low (MPR:L)
			High (MPR:H)
		User Interaction (UI)	Not Defined (MUI:X)
			None (MUI:N)
			Required (MUI:R)
		Scope (S)	Not Defined (MS:X)
			Changed (MS:C)
			Unchanged (MS:U)
	Impact	Confidentiality Impact (C)	Not Defined (MC:X)
			None (MC:N)
			Low (MC:L)
			High (MC:H)
		Integrity Impact (I)	Not Defined (MI:X)
			Low (MI:L)
			High (MI:H)
		Availability Impact (A)	Not Defined (MA:X)
			Low (MA:L)
			High (MA:H)
	Security Requirements	Confidentiality Requirements (CR)	Not Defined (CR:X)
			Low (CR:L)
			Medium (CR:M)
			High (CR:H)
		Integrity Requirements (IR)	Not Defined (IR:X)
			Low (IR:L)
			Medium (IR:M)
			High (IR:H)
		Availability Requirements (AR)	Not Defined (AR:X)
			Low (AR:L)
			Medium (AR:M)
			High (AR:H)

zu 2.3.2 Environmental Metric Group

Glossar

Basic Authentication Basic Authentication ist eine Methode zur Zugriffsauthentifizierung im Rahmen einer HTTP-Transaktion, um sich mittels Benutzernamen und Passwort zu authentifizieren.

Brute Force Unter der Brute Force Methode wird ein Ansatz verstanden, bei welchem sich durch Ausprobieren von diversen Passwörtern, Zugriff zu einem System verschafft wird.

Dashboard Das Dashboard bezeichnet im Kontext von Informationsmanagementsystemen eine grafische Benutzeroberfläche, die zur Visualisierung von Daten eingesetzt wird.

Denial-of-Service Unter Denial-of-Service wird eine Nichtverfügbarkeit eines Internetdienstes basierend auf gezielt verursachten Anfragen verstanden, welche zu einer Überlastung führen.

Domain Name System Das Domain Name System ist einer der wichtigsten Dienste in vielen IP-basierten Netzwerken. Seine Hauptaufgabe ist die Beantwortung von Anfragen zur Namensauflösung.

MX-Record Ein Mail Exchange Record ist ein Eintrag im Domain Name System und gibt an, unter welchem vollständig qualifiziertem Domainnamen der Mail-Server einer Domain erreichbar ist.

Public-Key Bei dem Public Key-Verschlüsselungsverfahren wird ein Klartext mit Hilfe eines öffentlichen Schlüssels verschlüsselt. Der verschlüsselte Text kann dann nur mit dazugehörigen Private Key wieder entschlüsselt werden.

Ransomware Der Begriff Ransomware stammt aus dem Englischen von ransom für „Lösegeld“ und beschreibt eine spezielle Form eines Trojaners, der zur Erpressung der Betroffenen eingesetzt wird. Hierbei werden die vertraulichen Daten des Betroffenen durch Verschlüsselungstechniken unbrauchbar gemacht, um diese dann erst nach Zahlung des Lösegelds wieder verfügbar zu machen.

Redirect Ein Redirect beschreibt eine URL-Weiterleitung, d.h. der Benutzer wird bei Aufruf einer URL automatisch zu einer bestimmten URL weitergeleitet.

Akronyme

BSI Bundesamt für Sicherheit in der Informationstechnik

CPE Common Platform Enumeration

CT Certificate Transparency

CSV Comma Seperated Values

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

DRF Django Rest Framework

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IMAP Internet Message Access Protocol

IoT Internet of Things

ISM Informationssicherheitsmanagement

ISMS Informationssicherheitsmanagementsystem

JSON JavaScript Object Notation

KPI Key Performance Indicator

NIST National Institute of Standards and Technology

NMAP Nmap: the Network Mapper

NVD National Vulnerability Database

POP Post Office Protocol

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

SSL Secure Sockets Layer

rDNS Reverse DNS Lookup

REST Representational State Transfer

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

URL Uniform Resource Locator

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original