Marco Madritsch

Holistischer Ansatz zur Erkennung von Phishing-Webseiten

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Studium: Masterstudium Angewandte Informatik

Alpen-Adria-Universität Klagenfurt

Begutachter
Assoc. Prof. Dipl.-Ing. Dr. Peter Schartner
Alpen-Adria-Universität Klagenfurt
Institut für Angewandte Informatik

Klagenfurt, Januar 2018

Bei Fragen, Problemen oder Anregungen kontaktieren Sie bitte die Forschungsgruppe Systemsicherheit (info@syssec.at) oder den Autor (marcomad@edu.aau.at).

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich

- die eingereichte wissenschaftliche Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe,
- die während des Arbeitsvorganges von dritter Seite erfahrene Unterstützung, einschließlich signifikanter Betreuungshinweise, vollständig offengelegt habe,
- die Inhalte, die ich aus Werken Dritter oder eigenen Werken wortwörtlich oder sinngemäß übernommen habe, in geeigneter Form gekennzeichnet und den Ursprung der Information durch möglichst exakte Quellenangaben (z.B. in Fußnoten) ersichtlich gemacht habe,
- die eingereichte wissenschaftliche Arbeit bisher weder im Inland noch im Ausland einer Prüfungsbehörde vorgelegt habe und
- bei der Weitergabe jedes Exemplars (z.B. in gebundener, gedruckter oder digitaler Form) der wissenschaftlichen Arbeit sicherstelle, dass diese mit der eingereichten digitalen Version übereinstimmt.

Mir ist bekannt, dass die digitale Version der eingereichten wissenschaftlichen Arbeit zur Plagiatskontrolle herangezogen wird.

Ich bin mir bewusst, dass eine tatsachenwidrige Erklärung rechtliche Folgen haben wird.

Marco Madritsch e.h. Klagenfurt, 26. Januar 2018

(Unterschrift)* (Ort, Datum)

*Bei der elektronischen Version ist es – aus datenschutzrechtlichen Gründen – nicht erforderlich, dass die eidesstattliche Erklärung unterschrieben wird. Sie soll mittels Kürzel „e.h.", dieses ist dem Namen nachzustellen, elektronisch gezeichnet werden.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich sowohl während der Anfertigung dieser Arbeit als auch auf dem Weg dorthin begleitet, unterstützt und motiviert haben.

Besonders bedanken möchte ich mich bei Herrn Assoc. Prof. Dipl.-Ing. Dr. Peter Schartner, der mir das Thema der Arbeit bereitgestellt und mich jederzeit mit Rat und konstruktiven Anregungen unterstützt hat.

Ein besonderer Dank gilt auch Herrn Mag. Dipl.-Ing. Dr. Marcus Hassler für seine ausgezeichnete technische Betreuung und die zahlreichen Stunden, die er für mich über die gesamte Dauer dieser Arbeit hinweg aufgebracht hat. Ohne ihn würde diese Arbeit in dieser Form nicht vorliegen.

Einen weiteren Dank möchte ich an Herrn Dr. Alejandro Correa Bahnsen für die Bereitstellung des Korpus an legitimen URLs aussprechen.

Weiters möchte ich mich noch recht herzlich bei meiner Familie und ganz besonders bei meinen Eltern bedanken, die mir nicht nur das Studium ermöglichten, sondern mir in allen Belangen meines Lebens stets mit Rat und Tat beiseitestehen und mir somit einen wichtigen Rückhalt geben.

Zu guter Letzt gilt ein besonderer Dank meiner Partnerin Kerstin, die in allen Lebenslagen hinter mit steht, mir während des gesamten Studiums auch an launischen Tagen viel Geduld entgegenbrachte, mich immer motivierte, nie an mir zweifelte und für mich stets eine bedeutende Stütze war und nach wie vor ist.

Kurzfassung

Phishing zählt heutzutage zu einer äußerst ernstzunehmenden Bedrohung unserer Gesellschaft und betrifft Personen sowohl im privaten als auch geschäftlichen Umfeld. Mit Hilfe von gefälschten Nachrichten und manipulierten Webseiten versuchen Angreifer hierbei das Opfer zur Preisgabe von sensiblen Informationen oder zur Durchführung einer bestimmten Handlung zu animieren. Da die Verbreitung und Beliebtheit derartiger Angriffe zusammen mit dem damit verbundenen Ausmaß an entstehenden Schäden jährlich enorm steigt, besteht gleichzeitig die Notwendigkeit zur Entwicklung von verbesserten Methoden und Ansätzen zu deren automatisierten Erkennung. Diese Arbeit soll einerseits einen Überblick über das Thema Social Engineering im Allgemeinen und Phishing als eine Ausprägung davon im Speziellen verschaffen und beschäftigt sich andererseits mit der Konzipierung und prototypischen Entwicklung eines holistischen Ansatzes zur Erkennung von Phishing-Webseiten. Die Architektur besteht aus einer clientseitigen Browser-Erweiterung für den Mozilla Firefox und eines serverseitigen RESTful Webservices. Im Zuge des Aufrufs einer Webseite durch den Browser sendet die Erweiterung die angeforderte URL an den Server und triggert somit den Analyseprozess. Dieser besteht aus insgesamt sechs übergeordneten Analyseschritten und bewertet die zur übermittelten URL dazugehörige Webseite einerseits anhand von bestimmten Heuristiken und andererseits anhand eines Klassifikators sowie des individuellen Surfverhaltensmusters des jeweiligen Benutzers. Das dazu berechnete Gesamtergebnis ordnet die Seite einem grünen, gelben oder roten Ampelsymbol zu, welches schlussendlich dem Benutzer im Browser als Maß für die ausgehende Phishing-Bedrohung visuell angezeigt wird. Der im Zuge der Arbeit implementierte Prototyp umfasst den Aufbau der Architektur sowie die Umsetzung der ersten vier Prozessschritte des gesamten Konzepts bzw. Ansatzes. Die Ergebnisse der Evaluierung zeigten hier mit einer Genauigkeit von 84,17 %, bei einer Sensitivität von 78,13 % und einer Spezifität von 90,22 % bereits gute Resultate, wobei der Prototyp das vorhandene Potenzial des Konzepts bei Weitem nicht völlig ausschöpft.

Inhaltsverzeichnis

1 Einleitung

Die heutzutage zunehmend verteilte und vernetzte Kommunikation bietet zahlreiche neue Möglichkeiten und hat enorme Auswirkungen auf unser privates als auch geschäftliches Umfeld. Vor allem das Internet hat sich in den letzten Jahren als das beliebteste und meistgenutzte Medium für den Informationsaustausch etabliert. Durch dessen Aufkommen entstanden eine Menge an neuen Kommunikationskanälen, welche sich mittlerweile zu nicht mehr wegzudenkenden Bestandteilen unseres Alltags entwickelt haben. Ob E-Mail, soziale Netzwerke oder Cloud-Services – sie alle bewirkten einen Paradigmenwechsel in der gemeinsamen Nutzung von Daten und der Art und Weise wie wir kommunizieren. Benutzer teilen und veröffentlichen über diese Kanäle tagtäglich massenweise Informationen aller Art, wobei oftmals die Sicherheit und Privatsphäre vernachlässigt bzw. außer Acht gelassen werden. In vielen Fällen lediglich mittels einer E-Mail-Adresse oder eines virtuellen Profils als Identifikationsmerkmal ausgewiesen, vertrauen Personen ihrem Gegenüber und geben sensible Informationen preis. Zwar steigt die Sicherheit dieser Kanäle aufgrund diverser technischer Maßnahmen (wie z.B. Verwendung sicherer Protokolle, Verschlüsselung oder Zertifikate) zunehmend, jedoch ist jede noch so ausgeklügelte Sicherheitsmethode gegen jegliche Angriffe, welche sich an die Benutzer persönlich richten, zwecklos. Bekannt unter dem Namen Social Engineering, handelt es sich hierbei um einer der beliebtesten und am weitest verbreiteten Methoden, um an sensible und vertrauliche Daten zu gelangen. Das Ziel derartiger Angriffe ist mit Hilfe von Methoden der Beeinflussung und der Überzeugungskunst Personen bewusst zu manipulieren bzw. zu täuschen, um sie so zur Preisgabe von Informationen zu animieren. Dadurch können sogar die besten Sicherheitsbarrieren eines Systems in kürzester Zeit mit nur geringem Aufwand gebrochen werden, da Social Engineering Methoden auf das schwächste Glied der Sicherheitskette abzielen – den Faktor Mensch [KHHW15].

Mittlerweile gibt es zahlreiche verschiedene Vorgehensweisen und Ansätze des Social Engineerings. Phishing zählt hierbei zu einer der am weitest verbreiteten und zugleich auch effektivsten Methoden dieser Art. Laut der Anti-Phishing Working Group (APWG) erfreuen sich diese Angriffe jährlich immer größerer Beliebtheit, wobei allein im Jahr 2016 die Anzahl um 65 % gegenüber dem Vorjahr stieg [AaMa17]. Bei einem Phishing-Angriff gibt sich der Angreifer typischerweise in einem elektronischen Kommunikationsverkehr als eine vermeintlich vertrauenswürdige Instanz aus, um so an sensible Daten zu gelangen oder das Opfer zu einer bestimmten Handlung zu animieren. Dazu sendet er meist eine gefälschte E-Mail, welche entweder einen schadhaften Anhang oder einen Link auf eine manipulierte Phishing-Webseite beinhaltet, an seine Zielpersonen und hofft dadurch deren Interesse zu wecken. Sobald das Opfer in Folge auf den schadhaften Link klickt, wird entweder eine beliebige Art von Malware am Rechner installiert oder es erscheint eine Phishing-Webseite, welche zur Eingabe von vertraulichen Daten (wie

z.B. Anmeldedaten) auffordert. Um derartigen Angriffen vorzubeugen, ist in erster Linie das Sicherheitsbewusstsein jedes Einzelnen und somit die Benutzerbildung bzw. -aufklärung essentiell. Wenn sich jeder der Gegenwärtigkeit und ausgehenden Gefahr von Phishing-Angriffen bewusst wäre und dementsprechend verdächtigen Nachrichten, vor allem von unbekannten Personen, mit Vorsicht und einer gesunden Skepsis gegenüberstehen würde, könnten diese Trickbetrüge nahezu im Keim erstickt werden. Da dies jedoch leider nicht der Fall ist und die Erfolgsaussichten sogar relativ hoch sind, existieren aus technischer Sicht mittlerweile zahlreiche unterschiedliche Lösungen zu deren automatisierter Erkennung. Mit Hilfe von E-Mail-Filtern wird einerseits versucht, einen Angriff bereits anhand der dazugehörigen Phishing-Nachricht frühzeitig erkennen und verhindern zu können. Wenn dies jedoch nicht gelingt und der Benutzer schließlich auf den schadhaften Link klickt, kann der Angriff andererseits noch anhand der Untersuchung der erscheinenden Phishing-Webseite entdeckt und unterbunden werden.

Die vorliegende Arbeit soll einerseits einen Überblick über Phishing und die möglichen Gegenmaßnahmen dazu verschaffen und stellt andererseits einen holistischen Ansatz zur Erkennung von Phishing-Webseiten vor. Dazu wird einführend in Kapitel 2 das Thema Social Engineering und dessen Facetten näher betrachtet. Darauf aufbauend beschreibt Kapitel 3 ausführlich den typischen Ablauf eines Phishing-Angriffes und fasst die gängigsten Vorgehensweisen eines Angreifers zusammen. Um einen möglichst optimalen Schutz gewährleisten zu können, werden außerdem noch sowohl technische als auch nichttechnische Gegenmaßnahmen dazu angeführt. Bevor in Kapitel 4 ein auf Whitelist-, Blacklist- und Heuristik-basiertes Konzept zur Erkennung von Phishing-Webseiten vorgestellt wird, werden zuvor einige bereits vorhandene Lösungen und Ansätze präsentiert. Anschließend beschreiben die beiden Kapitel 5 und 6 eine prototypische Implementierung des vorgeschlagenen Konzepts in Form eines serverseitigen RESTful Webservices sowie einer clientseitigen Browser-Erweiterung für den Mozilla Firefox. Zur Feststellung der Effektivität des Prototyps wurde zusätzlich eine Evaluierung durchgeführt, welche zusammen mit allen Ergebnissen in Kapitel 7 wiederzufinden ist. Zum Abschluss beinhaltet Kapitel 8 neben einem kurzen Fazit noch mögliche Verbesserungsvorschläge bzw. Erweiterungen des Prototyps.

2 Grundlagen

Dieses Kapitel soll als Einstieg dienen und einen grundlegenden Überblick über das dem Phishing zugrunde liegenden Thema Social Engineering bereitstellen. Dazu wird zuerst der Begriff Social Engineering selbst näher erläutert und anschließend die Motivation bzw. das Ziel dahinter hervorgehoben. Daraufhin werden die Erfolgsfaktoren eines derartigen Angriffes sowie die damit verbundenen Bedrohungen und Angriffspunkte betrachtet. Abschließend werden noch zwei Modelle zur Beschreibung eines Social Engineering Angriffes sowie die dazu verwendeten Methoden und Techniken dargestellt.

2.1 Social Engineering

Der Begriff Social Engineering wird auch heute noch weitgehend missverstanden, was zu zahlreichen verschiedenen Ansichten und Meinungen über seine Bedeutung führt. Die Auffassungen reichen dabei angefangen von einfachem Lügen, um kostenlos Waren oder Dienstleistungen zu ergaunern, über Tools, welche Kriminelle und Trickbetrüger verwenden, bis hin zu einer mystischen Kunst, welche Praktizierenden die gedankenmanipulierenden Fähigkeiten eines Zauberkünstlers oder Illusionisten verleiht [HaWi10].

Tatsächlich macht jeder gewöhnliche Alltagsmensch tagtäglich in nahezu allen Lebenssituationen von Social Engineering Gebrauch. Sei es ein kleines Kind beim Einkaufen, um dessen Willen in der Süßigkeitenabteilung durchzusetzen, ein Mitarbeiter bei der Bitte um eine Lohnerhöhung oder die Vorgehensweise eines Arztes, Rechtsanwaltes oder Psychologen, um Informationen von Patienten oder Klienten zu bekommen – es wird wahrhaftig in jeder menschlichen Interaktion verwendet. In Folge bedeutet das leider auch, dass Social Engineering ebenfalls gegenwärtig ist, wenn Kriminelle, Trickbetrüger und dergleichen im Zuge einer Straftat andere Menschen zur Preisgabe von sensiblen Informationen zu verleiten versuchen. Wie jedes Tool oder Werkzeug ist auch Social Engineering weder „gut“ noch „böse“, sondern dessen moralische Vertretbarkeit hängt von der jeweiligen Verwendung ab [HaWi10].

2.1.1 Definition

Auch in der Literatur lassen sich zahlreiche verschiedene Definitionen des Begriffs Social Engineering finden. Mitnick und Simon [MiSi06] beschreiben ihn etwa als die gezielte Manipulation oder Täuschung von Personen durch die Verwendung von Methoden der Beeinflussung und der Überzeugungskunst. Damit versucht ein Social Engineer sich eine falsche Identität anzueignen, um so entweder mit oder ohne der Verwendung von technischen Hilfsmitteln an vertrauliche Informationen zu gelangen.

Mann [Mann12] beschreibt Social Engineering hingegen zunächst als Praxis zur Erlangung von vertraulichen Informationen durch die Manipulation von legitimen Benutzern. Die Intention bzw. das Ziel hinter einer Manipulation einer anderen Person muss jedoch nicht immer die direkte Preisgabe von Informationen sein. Beispielsweise erlangt ein Angreifer durch die Täuschung eines Wachmannes, um so Zutritt zu einem Gebäude zu erhalten, keinerlei sensible Daten, jedoch verhilft ihm diese Manipulation auf dem Weg dorthin. Aus diesem Grund konkretisiert Mann Social Engineering als eine durch Täuschung hervorgerufene Manipulation von Personen, damit diese entweder vertrauliche Informationen preisgeben oder eine bestimmte Handlung ausführen.

Hadnagy [HaWi10] beschreibt Social Engineering wiederum als die Kunst bzw. sogar die Wissenschaft, Menschen in geschickter Art und Weise zu bestimmten Handlungen in einem Aspekt ihres Lebens zu manövrieren. Er präzisiert diese Definition und ist schlussendlich der Ansicht, dass Social Engineering der Vorgang der Manipulation einer Person ist, um anschließend eine bestimmte Maßnahme ergreifen zu können, welche im Interesse der Zielperson liegen kann oder auch nicht. Dazu zählt er das Erlangen von Informationen, das Erhalten von Zutritt zu einem Gebäude oder Zugang zu einem System, oder das Anregen einer Person zu einer bestimmten Handlung. Zur Erläuterung der positiven bzw. negativen Anwendung von Social Engineering vergleicht er die Vorgehensweise eines Arztes, Psychologen oder Therapeuten im Gegensatz derer eines Trickbetrügers. Erstere greifen auf Social Engineering Ansätze zurück und „manipulieren“ somit ihre Patienten, um aber schlussendlich Handlungen zu ihrem Wohle ergreifen zu können. Ein Trickbetrüger hingegen verwendet Aspekte des Social Engineerings, um sein Opfer zu einer mit einem bestimmten Nachteil behafteten Handlung zu animieren. In beiden Fällen kann das Vorgehen weitgehend ident, das Ziel und Endergebnis jedoch komplett konträr sein. Hadnagy betont vor allem, dass Social Engineering nicht nur diese eine Handlung ist, in welcher man Personen täuscht, lügt oder eine andere Rolle spielt. Eine köstliche Mahlzeit besteht in der Regel auch nicht nur aus einer einzigen Zutat, sondern ergibt sich erst durch die sorgfältige Hinzunahme und Mischung von mehreren einzelnen Zutaten. Gleichermaßen ist Social Engineering viel mehr eine Sammlung von mehreren Fertigkeiten, welche richtig kombiniert und eingesetzt einen guten Social Engineer auszeichnen.

Trotz der vielen unterschiedlichen Sichtweisen sind sich alle einig, dass Social Engineering im Allgemeinen eine auf die menschliche Tendenz zu vertrauen basierende Manipulation von Personen ist. Ziel eines derartigen Angriffes ist meist das Erlangen von Informationen, um mit deren Hilfe einen unberechtigten Zugang zu einem System und der darin befindlichen Daten zu erhalten [Gran01].

2.1.2 Motivation und Ziele

Die Motivation hinter einem Social Engineering Angriff unterscheidet sich grundsätzlich nicht von jener eines herkömmlichen Hacker-Angriffes. Der Angreifer kann dabei unter anderem folgende Ziele verfolgen (vgl. [Gran01] [Nohl08]):

- Einen unautorisierten Zugang zu einem System oder Informationen zu erlangen, um damit einen Betrug oder andere Verbrechen begehen zu können
- In ein fremdes Netzwerk eindringen
- Ein fremdes Netzwerk oder System lahmlegen
- Industriespionage

- Identitätsdiebstahl
- Rufschädigung und damit verbundene finanzielle Einbußen

Kilger et al. [KiAS04] haben außerdem sechs mögliche Motivations- bzw. Beweggründe für nicht ethisch vertretbare Computeraktivitäten identifiziert. Diese lassen sich ebenfalls auf Social Engineering Angriffe übertragen (vgl. [KiAS04]):

Geld: Vor allem in der frühen Geschichte der Hackergemeinschaft war das illegale Beschaffen von großen Geldsummen oder anderen Finanzmitteln mit Hilfe von Hacking-Fähigkeiten äußerst verwerflich. Natürlich hinderte dies einzelne Mitglieder der Community trotzdem nicht daran, Waren oder Dienstleistungen in einem kleineren Ausmaß auf diesem Wege zu ergaunern, allerdings wurden diese Personen von der restlichen Gemeinschaft gemieden und ausgegrenzt. Im Laufe der Zeit hat sich diese Ansicht jedoch geändert und mittlerweile sind derartige Vorfälle keine Seltenheit mehr. Dabei versuchen Angreifer unter dem Vorwand, das jeweilige System kompromittiert zu haben und somit in Besitz von sensiblen Informationen zu sein, liquide Mittel vom abgezielten Unternehmen zu erpressen. Viele dieser Angriffe bleiben ungeklärt bzw. treten erst gar nicht an die Öffentlichkeit, da die betroffenen Unternehmen diese nicht melden und heimlich das „Schweigegeld“ zahlen. Anstatt Geld direkt von Unternehmen zu erpressen, bieten Hacker heutzutage sogar Denial-of-Service (DoS) Angriffe gegen Konkurrenten als „Dienstleistung“ an. Weiters ist die Anzahl an Kreditkarten-Diebstählen ebenfalls stark steigend. Durch das Kompromittieren von Sicherheitsfunktionen von E-Commerce Webseiten werden jährlich unzählige Kreditkartennummern und dazugehörige persönliche Informationen gestohlen und ermöglichen den Angreifern somit die erfolgreiche Durchführung von Online-Bestellungen unter einer falschen Identität.

Vergnügen: Das Vergnügen bzw. die Unterhaltung als Beweggrund eines Angriffes ist womöglich die Motivation mit den geringsten Konsequenzen für das anvisierte Unternehmen, weil hier das schlussendliche Ziel meist nicht destruktiver Natur ist. Darunter fällt beispielsweise das Hacken einer Webseite eines Unternehmens oder einer staatlichen Institution, um darauf unangebrachte Fotos oder Worte zu veröffentlichen, oder das Kompromittieren eines Mail-Servers, um prekäre E-Mails zu veröffentlichen oder weiterzuleiten. Die Anzahl an verschiedenen Vorhaben im Namen der Unterhaltung bzw. des Vergnügens ist nahezu grenzenlos.

Ego: Das eigene Ego spielt bis zu einem gewissen Grad bei jedem Angriff eine Rolle. Kernpunkt dieser Motivation ist die entstehende Befriedigung bei der Überwindung von technischen Hindernissen oder bei der Erarbeitung von innovativen Lösungen für ein Problem. Ein Gerät oder ein System dazu zu bringen, etwas genau so zu machen wie man es im Vorhinein beabsichtigt hat, ist ein derart starker Motivator, der bei solchen Angriffen nicht zu unterschätzen ist.

Anliegen (Ideologie): Von dieser Motivation werden meist sogenannte Hacktivisten getrieben – Hacker, die das Internet zur Förderung von bestimmten politischen, wissenschaftlichen oder sozialen Anlässen nutzen. Die häufigste Art von Hacktivismus ist die Verwendung von Computertechnologie, um einen bestimmten politischen Standpunkt zu verunglimpfen während die gegenüberstehende Ideologie befürwortet wird. Dadurch kommt ein „ziviler Cyberkrieg“ auf. Während Bürger früher nicht die Möglichkeit hatten, Einwände gegen

ideologische, politische oder militärische Maßnahmen zu erheben, können sie nun mit Hilfe von Hacking-Fähigkeiten Regierungs- und Militärsysteme kompromittieren und damit ihren Unmut preisgeben.

Beitritt in eine soziale Gruppe: Menschen sind von Natur aus soziale Wesen und haben daher den Drang sich zu sozialen Gruppen zusammenzuschließen. Hacker bzw. Kriminelle sind in dieser Hinsicht ganz gleich. Die soziale Struktur einer derartigen Gruppe ist meist eine stark ausgeprägte Meritokratie – sprich die Stellung eines Einzelnen innerhalb der Gruppe ist abhängig von dessen technischen Fähigkeiten und Leistungen. Da die technische Expertise jedoch nicht direkt „gemessen“ werden kann, wird diese oft bei der Aufnahme anhand von bereits implementierten (schadhaften) Beispielcodes oder Exploits bewertet. Daher kann die Motivation für einen Anwärter einer solchen Gruppe das Schreiben von elegantem Code zur Demonstration seines Könnens sein.

Status: Der Status bzw. die Stellung eines Einzelnen innerhalb der Hackergemeinschaft ist eine der am stärksten ausgeprägten Antriebskräfte. Das Hauptproblem in einer Meritokratie ist die Schwierigkeit seine eigene Position zu übermitteln, da diese Gruppen zum Großteil ausschließlich im Cyberspace existieren. Sämtliche Konversationen sind meist virtuell und somit können auffällige Markenzeichen einer Person (wie z.B. Eigenschaften der Aussprache, Blickrichtung während dem Sprechen oder Zuhören, usw.) nicht übermittelt werden. Daher müssen sie ihren Status in einer anderen Art und Weise kommunizieren – beispielsweise durch das Prahlen mit der Anzahl an Systemen, die sie gehackt haben bzw. die sich unter ihrer Kontrolle befinden. Hier von anderen abzuheben und somit der „Beste“ zu sein, ist dementsprechend eine hervorragende Motivation für das Kompromittieren von Systemen.

Typische Ziele eines Social Engineering Angriffes sind vor allem namhafte und renommierte Unternehmen, Finanzinstitute, Militär und Regierungsbehörden, Krankenhäuser sowie Infrastrukturanbieter (Hardware, Software und Kommunikation). Die meisten Angriffe zielen in der Regel auf größere Unternehmen mit hoch dotierten Vermögensgegenständen ab, kleine Unternehmen und Startups sind eher die Ausnahmen [Gran01] [Nohl08]. Innerhalb einer Organisation wiederum sind folgende Ziele die ersten Anlaufstellen für einen Social Engineer (vgl. [Nohl08]):

- Personen, welchen der Wert von Informationen nicht bewusst ist, wie z.B. Verwaltungspersonal, Empfangspersonal oder Sicherheitspersonal
- Personen mit besonderen Privilegien, wie z.B. Mitarbeiter des technischen Supports oder Systemadministratoren
- Spezielle Abteilungen mit wertvollen Informationen, wie Buchhaltung oder Personalabteilung
- Andere Personen, die entweder über ein mangelndes Sicherheitsbewusstsein verfügen, die im Zuge ihrer Arbeit anderen helfen, die hohe Zugriffsrechte, spezielles Wissen oder Zugang zu etwas anderem Wertvollen haben

Aufgrund der hohen Erfolgsaussichten ist Social Engineering mittlerweile Bestandteil von nahezu allen Angriffen und in vielen Fällen stellt diese Vorgehensweise den Weg des geringsten Widerstandes dar. Da sich weiters auch die Sicherheit von heutigen Softwareprodukten und Systemen ständig erhöht, steigt damit auch die Schwierigkeit für Angreifer diese zu kompromit-

tieren. Während ein Brute-Force Angriff auf ein Passwort mittlerweile schon Stunden, Tage oder sogar Wochen dauern kann, sind mit den richtigen Social Engineering Methoden und Ansätzen oft nur wenige Minuten dafür erforderlich. Deswegen werden heutzutage immer öfters Vorgehensweisen des Social Engineerings zusammen mit digitalen Hacking-Methoden kombiniert, um dadurch Angriffe noch effektiver und profitabler zu gestalten [SEF].

2.1.3 Warum Social Engineering Erfolg hat

Die äußerst vielversprechenden Erfolgsaussichten von Social Engineering sind großteils auf den Faktor Mensch selbst zurückzuführen. Ein Unternehmen kann trotz der Umsetzung der besten Sicherheitstechnologien, der ständigen Sensibilisierung und Weiterbildung der Mitarbeiter, der Befolgung sämtlicher Best Practices in Bezug auf Datensicherheit und der Bewachung aller Firmengebäude immer noch absolut gefährdet sein [MiSi06].

Der Faktor Mensch

Spricht man in der IT über Sicherheit, so fallen meistens Begriffe wie Verschlüsselung, Authentifizierung, digitale Signatur, digitales Zertifikat, Smart Card, Firewall, Virenschutz, usw. Natürlich spielen all diese Faktoren eine unverzichtbare Rolle in der IT-Sicherheit, jedoch wird leider oft einer Schwachstelle keine bzw. nur wenig Aufmerksamkeit geschenkt: dem Faktor Mensch. Treffend nach dem Sprichwort „Eine Kette ist nur so gut, wie ihr schwächstes Glied“ ist und bleibt der Mensch das schwächste Glied der Sicherheitskette [Lips09] [MiSi06] [Mann12].

Voraussetzung für einen erfolgreichen Angriff ist immer eine Komponente, die nicht ausreichend geschützt, falsch konfiguriert oder sogar fehlerhaft programmiert ist. Solange dieser „Fehler“ nicht behoben wurde, bleibt das jeweilige System gefährdet. Im Unterschied zu einem Programm oder System kann eine menschliche Schwäche jedoch nicht einfach „aktualisiert“ bzw. „gepatcht“ und damit der „Fehler“ beseitigt werden. Der Mensch ist ein dynamisches Wesen und kann nur auf Veränderungen reagieren und in Abhängigkeit seiner bereits vorhandenen Lebenserfahrung entsprechend handeln bzw. dazulernen. Da die Reaktionen eines Menschen allerdings situationsabhängig und im Vorfeld nicht ermittelbar sind, verlässt sich ein Social Engineer auf gewisse Gemeinsamkeiten, die jeder Mensch in seinem sozialen Umfeld entwickelt haben sollte. Darunter fällt beispielsweise das menschliche Bedürfnis hilfsbereit zu sein sowie unsere Nächsten zu lieben und einander vertrauensvoll zu begegnen [Lips09] [MiSi06]. Zusammen mit stark ausgeprägten Charakterzügen wie Höflichkeit, Sympathie und Charme nützt dies ein Social Engineer zum Aufbau eines Vertrauensverhältnisses zu seinen Zielpersonen aus [MiSi06]. Die Bildung von Vertrauen spielt dabei zwar eine wichtige, aber dennoch für einen Angriff nicht unbedingt notwendige Rolle [Lips09].

Als weiteren Erfolgsfaktor für Social Engineering nennen Mitnick und Simon [MiSi06] das falsche Sicherheitsgefühl von uns Menschen. Sicherheit ist ihrer Meinung nach demnach oft nur eine Illusion, welche in vielen Fällen durch Naivität, Ignoranz und Leichtgläubigkeit begünstigt wird. Vergleichbar mit einem fürsorglichen Hausbesitzer, welcher durch den Einbau eines einbruchsicheren Schlosses im Glauben der absoluten Sicherheit lebt, jedoch einen möglichen Einbruch durch das Zerschlagen eines Fensters oder durch das Knacken des Garagentores nicht berücksichtigt.

„Zwei Dinge sind unendlich, das Universum und die menschliche Dummheit, aber bei dem Universum bin ich mir noch nicht ganz sicher.“ (Albert Einstein)

Letztlich kann der Erfolg von Social Engineering Angriffen oft auf die menschliche Dummheit bzw. fehlende Ahnung von Sicherheitspraktiken zurückgeführt werden. Dementsprechend sind viele Sicherheitsexperten überzeugt, dass sie ihr Unternehmen durch die Installation von Standard-Sicherheitsprodukten (wie z.B. Firewalls, Authentifizierungsmechanismen oder Zutrittsbeschränkungen) weitgehend vor Angriffen geschützt haben. Jedoch ist Sicherheit kein Produkt oder rein technischer Aspekt, sondern viel mehr ein Prozess, welcher auch von zahlreichen menschlichen Faktoren abhängig ist [MiSi06].

Erfolgsfaktoren sozialer Manipulation

Zur Beeinflussung einer anderen Person in ihrem Tun und Handeln greift ein Social Engineer auf zahlreiche menschliche Eigenschaften und Tendenzen zurück. Die einfachste und oft auch erfolgreichste Art und Weise der sozialen Manipulation ist schlicht und einfach freundlich zu sein. In vielen Fällen kann mit reiner Freundlichkeit bereits schon einiges erreicht werden, da der Durchschnittsmensch stets bestrebt ist, anderen behilflich zu sein. Um die Erfolgschance zu erhöhen, wird oft bei Anfragen bzw. Wünschen zusätzlich ein Grund vorgetäuscht. Wenn man eine Person um einen Gefallen bittet, kann die Verwendung des Wortes „weil“ in der Frage gleich effektiv wie die Angabe eines tatsächlichen Grundes sein [Gupt08]. Des Weiteren kann die relativ einfache Manipulation von Personen und die damit verbundene Gefährdung von derartigen Angriffen auf die sechs folgenden Tendenzen zurückgeführt werden (vgl. [MiSi06] [Gupt08]):

Autorität: Menschen kommen einer Aufforderung viel eher nach, wenn diese von einer Autoritätsperson ausgesprochen wird. Uns wird üblicherweise von Kindheit auf schon der Respekt gegenüber autoritären Personen beigebracht – ob zu Hause, in der Schule, beim Militär oder am Arbeitsplatz. Ob wir eine Person als autoritär ansehen oder nicht, geht jedoch weit über verbale Aufforderungen hinweg und kann von zahlreichen weiteren Aspekten abhängen. Ein Beispiel dafür ist das Tragen einer Uniform. Das können besonders auffällige Uniformen sein, wie z.B. eine Polizeiuniform oder ein Arztmantel, doch im Bezug auf Social Engineering sind unscheinbare Uniformen, wie z.B. die einer Reinigungskraft oder eines Technikers, weitaus effektiver. Diese Personen haben in der Regel Zutritt zu sämtlichen Räumlichkeiten, werden nur selten von anderen befragt und arbeiten auch häufig zu Zeiten, in denen nur wenige oder keine anderen Mitarbeiter anwesend sind. Genau aus diesen Gründen stellen sie ein potentielles Risiko dar. Weiters kann gewöhnliche Kleidung ebenfalls als eine Art Uniform angesehen werden und eine gewisse Nachricht übermitteln (z.B. das Tragen eines „nerdigen“ T-Shirts im Gegensatz zu einem maßgeschneiderten Anzug). Doch auch der Titel einer Person ist eine spezielle Form einer Uniform. Aussagekräftige Titel, wie beispielsweise Doktor, Professor oder ähnliche, können das Ausmaß an wahrnehmender Autorität ebenso beeinflussen. Natürlich erfordert das Erlangen eines derartigen Titels oft jahrelange harte Arbeit, ein Social Engineer „erwirbt“ einen gefälschten Titel jedoch in nur wenigen Sekunden.

Zur Veranschaulichung der Ausnutzung dieser Tendenz nennen Mitnick und Simon [MiSi06] ein durchgeführtes Experiment in verschiedenen Krankenhäusern als Beispiel. Dabei kontaktierte ein Anrufer 22 Krankenstationen telefonisch, gab sich als Arzt aus und forderte das Pflegepersonal zur Verabreichung eines bestimmten Medikamentes für einen stationären Patienten auf. Obwohl eine Verschreibung per Telefon nicht erlaubt,

das Medikament auf der jeweiligen Station nicht zugelassen und die Dosis viel zu hoch war, wären 95 % der Krankenschwestern der Aufforderung nachgekommen – nur weil sich der Anrufer als Arzt des jeweiligen Hauses ausgab. Ein Social Engineer könnte sich diese Tendenz wiederum zunutze machen, in dem er sich als Mitarbeiter der IT-Abteilung oder der Geschäftsführung ausgibt und sich somit den Trug von Autorität verleiht.

Zuneigung: Menschen bevorzugen eher eine Zusammenarbeit mit Personen, mit welchen sie Gemeinsamkeiten teilen und die ihnen sympathisch und nett erscheinen. Diese Gemeinsamkeiten können dabei von beliebiger Natur sein: der gleiche Kleidungsstil, die gleichen Interessen, Ansichten oder Verhaltensweisen, um nur einige Beispiele zu nennen. Die Zuneigung zu einer anderen Person kann mit regelmäßigem Kontakt und der dadurch steigenden Vertrautheit erhöht werden. Besonders tückisch ist hier, dass das Opfer den Aufbau von Vertrautheit nicht bemerkt, denn dieser entsteht sogar sobald wir andere Personen regelmäßig sehen (z.B. bei der Arbeit) oder stets mit ihnen in Kontakt stehen. Ein gemeinsames Feindbild, gegenseitiges Helfen oder das Machen von Komplimenten kann ebenfalls positive Auswirkungen auf die Zuneigung zu einer anderen Person haben. Ein weiterer Faktor in diesem Zusammenhang ist auch das körperliche Erscheinungsbild des Gegenüber. Eine attraktive Person empfinden wir manchmal als klüger, freundlicher, stärker und generell als angenehmer.

Dieses Wissen kann ein Social Engineer wiederum nutzen, in dem er beispielsweise im Zuge eines Gespräches zunächst versucht, möglichst viele Informationen über das Opfer herauszufinden. Anschließend kann er diese Details zum Vortäuschen von Ähnlichkeiten verwenden – in dem er beispielsweise behauptet, die gleichen Interessen bzw. Hobbys zu besitzen, aus dem gleichen Staat zu kommen oder die gleiche Schule besucht zu haben.

Revanchieren: Diese Neigung ist in uns Menschen tief verwurzelt und bedeutet, dass wir stets bestrebt sind einen Gefallen zu erwidern, auch wenn wir nicht danach gefragt werden. Es ist mehr oder weniger eine automatische Reaktion, welche eine überaus starke Methode zur Beeinflussung von Personen darstellt. Supermärkte machen sich ebenfalls diese Tendenz zu Nutze. In dem sie gratis Kostproben von Lebensmitteln anbieten, bekommen Kunden das Gefühl ein Geschenk erhalten zu haben. Somit glauben sie (wenn auch nicht bewusst) in der Schuld der Verkaufsperson zu stehen und fühlen sich somit verpflichtet, das Produkt zur „Begleichung“ zu kaufen. Typischerweise besteht der Drang des Revanchierens auch wenn der Gegendienst verhältnismäßig viel größer als die erwiesene Gefälligkeit ist.

Auch der Social Engineer macht sich diese Tendenz zunutze. Er könnte z.B. einen Mitarbeiter anrufen, sich selbst als Kollege aus der IT-Abteilung ausgeben und behaupten, dass ein neuer Virus im Umlauf sei. Da der Virus von der aktuellen Schutzsoftware noch nicht erkannt wird, bietet er zur Vorbeugung von Problemen seine Hilfe an. Sobald der Mitarbeiter nun diese Hilfe annimmt, bittet er ihn nachfolgend eine neue (schadhafte) Software zu testen. Da der Mitarbeiter zuvor die Hilfe des Anrufers in Anspruch genommen hat, wird er typischerweise diese Bitte nicht abschlagen.

Konsequenz: Menschen versuchen in der Regel jeglichen Misserfolg zu meiden. Wenn wir etwas versprechen, so sind wir im Allgemeinen stets bestrebt es konsequent einzuhalten, damit wir nicht als unzuverlässig oder unwillig gegenüber anderen erscheinen. Gleichermaßen steigt die Tendenz zur Zusammenarbeit ebenfalls, wenn wir uns zuvor bereits öffentlich für

etwas eingesetzt oder gar verpflichtet haben. Wir kommen einem Engagement am ehesten nach, wenn dieses aktiv, öffentlich und mit einem gewissen Maß an Aufwand verbunden ist. Übernehmen wir in irgendeiner Form Verantwortung, begünstigt dies ebenfalls ein rasches und verantwortungsbewusstes Nachkommen. Daher sind beispielsweise Glücksspieler nach dem Setzen von ihren Gewinnchancen überzeugter als zuvor und Mitglieder von Wohltätigkeitsorganisationen stets bestrebt, konsequent Unterschriften für Charity-Programme zu sammeln.

Als Beispiel eines Angriffes, welcher auf dieser menschlichen Tendenz beruht, kann die Kontaktaufnahme eines Social Engineers zu einem neuen Mitarbeiter herangezogen werden. Im Zuge dessen klärt er ihn bezüglich der im Unternehmen vorherrschenden Sicherheitsrichtlinien und -prozeduren auf. Um anschließend „die Einhaltung der Passwortrichtlinien zu überprüfen“, fragt er den Mitarbeiter nach seinem derzeitigen Passwort. Sobald dieser ihm sein Passwort erst einmal offenbart hat, gibt der Angreifer daraufhin derartige Ratschläge bezüglich der Gestaltung von zukünftigen Passwörtern, so dass er mit geringem Aufwand in der Lage sein wird, diese zu erraten. Da der Mitarbeiter zuvor der Einhaltung sämtlicher Richtlinien zustimmte, ist er auch anschließend zu jeglicher Mitarbeit bereit.

Soziale Bestätigung: Wenn Menschen einer bestimmten Situation nicht gewachsen sind oder in Bezug auf eine Entscheidung Unsicherheit verspüren, richten sie sich oft an das Verhalten von Personen in ihrer näheren Umgebung. Durch die Tatsache, dass andere Personen gleich handeln, wird das in Frage gestellte Verhalten als richtig und angemessen angesehen. Dieses Phänomen ist als soziale Bestätigung bekannt und veranlasst Menschen in vielen Fällen nicht in ihrem besten Interesse zu agieren. Beispiele dazu sind unter anderem der Kauf eines Produktes, nur aufgrund dessen starker Nachfrage oder das Teilen von Passwörtern mit Mitarbeitern, weil es alle anderen in der Abteilung ebenfalls praktizieren. In ihrer schlimmsten Ausprägung kann diese Tendenz bis zu einem Phänomen namens „pluralistische Ignoranz“ führen. Hierbei spricht man von Notfallsituation, in welchen Personen sich aufgrund von Ratlosigkeit dem Verhalten der Mehrheit anpassen und schlussendlich niemand einschreitet bzw. hilft. Dazu zählen beispielsweise in Anwesenheit von zahlreichen Zeugen begangene Straftaten und niemand kommt dem Opfer zu Hilfe, oder wenn jemand auf offener Straße zusammenbricht und niemand anhält, um nach seinem Wohlbefinden zu fragen. Eilt in derartigen Situation jedoch auch nur eine einzige Person zur Hilfe, löst dies wiederum eine Kettenreaktion aus und weitere Freiwillige bleiben ebenfalls stehen, um zu helfen.

In Bezug auf ein Unternehmen kann dies jedoch gravierende Auswirkungen auf die Sicherheit haben, weil Mitarbeiter ihr Verhalten an das allgemeine Verhalten richten und nicht an Vorschriften aus Sicherheitsrichtlinien. Dadurch kann jegliches Maß an gewünschter Sicherheit durch die Mitarbeiter zunichte gemacht werden. Ein Beispiel dazu sind Unternehmen, in welchen Mitarbeiter ihre Passwörter weitergeben, obwohl dies ausdrücklich verboten ist. Würde dies hingegen ein Einzelner entgegen dem allgemeinen Verhalten und der sozialen Bestätigung nicht ebenso praktizieren, würde er als paranoid abgestempelt und von der Gruppe ausgegrenzt werden.

Ein Angreifer kann z.B. im Zuge der Durchführung einer Umfrage behaupten, dass andere Mitarbeiter aus der Abteilung ihn bereits ebenfalls unterstützt haben. Durch diesen

zusätzlichen Hinweis nimmt er der befragten Person jegliche Zweifel bezüglich der Authentizität der Anfrage und das Opfer wird der Teilnahme schließlich zustimmen. Anschließend versucht er durch geschicktes und gezieltes Fragen an Benutzernamen und Passwörter zu gelangen.

Mangel: Wenn Menschen wissen, dass ein gewünschtes Objekt knapp und nicht frei verfügbar ist, neigen sie eher zu einer Kooperation. Durch die Tatsache, dass andere ebenfalls das Objekt für sich in Anspruch nehmen wollen, kann dieser Umstand ein Gefühl des Konkurrenzkampfes in ihnen auslösen. Diese Tendenz kann bei Verkaufswerbungen mit beschränkten Angeboten (z.B. limitierte Anzahl oder begrenzte Verkaufszeit) beobachtet werden. Das bedeutet gleichzeitig auch, dass wir Objekten, welche schwierig zu erwerben sind, eine höhere Wertschätzung entgegenbringen als Objekten, dessen Erwerb sich als einfacher darstellt. Das gleiche Phänomen trifft auch auf Informationen zu: Wenn bestimmte Informationen verboten bzw. geheim sind, haben wir ein größeres Verlangen an dessen Erwerb. Wenn sie nicht frei verfügbar sind (weil sie z.B geheim sind), müssen sie interessant sein und weil sie geheim sind, sind sie limitiert und alles, das limitiert ist, weckt unser Interesse noch mehr.

Ein Social Engineer kann diese Neigung ausnützen, in dem er beispielsweise ein E-Mail mit der Bitte um Registrierung auf der „neuen“ Unternehmenswebseite an sämtliche Mitarbeiter sendet. In dem er zusätzlich angibt, dass die ersten 500 registrierten Personen eine kleine Anerkennung bekommen (z.B. ein Freiticket für einen Kinobesuch oder Ähnliches), versucht er das oben beschriebene Gefühl der beschränkten Verfügbarkeit auszulösen. Im Zuge der Registrierung wird der Benutzer zur Eingabe seiner dienstlichen E-Mail-Adresse sowie eines Passwortes aufgefordert. Da sich viele Menschen aus Gründen der Bequemlichkeit nicht viele verschiedene Passwörter merken wollen, wählen sie oft bei sämtlichen Registrierungen das gleiche oder zumindest ein ähnliches Passwort. Somit kann der Angreifer anschließend mit Hilfe dieser gewonnenen Informationen alle verwendeten Systeme der Zielperson zu kompromittieren versuchen.

2.1.4 Bedrohungen und Angriffspunkte

Viele Unternehmen sind stets bestrebt ein umfangreiches Sicherheitsprogramm umzusetzen und bauen dabei auf internationale Standards wie z.B. ISO/IEC 27001[1]. Diese decken zwar viele Bereiche bzw. Aspekte der Sicherheit ab, doch dem Thema Social Engineering wird hierbei keine oder nur wenig Aufmerksamkeit geschenkt. Es wird zwar oft kurz erwähnt und auf Benutzerbewusstsein und Schulungen hingewiesen, aber eine ausführliche Aufklärung der möglichen Bedrohungen und Gegenmaßnahmen ist nicht enthalten und erfordert somit die Berücksichtigung von weiteren Maßnahmen. Um einen Überblick über die möglichen Risiken zu bekommen, erläutert Mann [Mann12] die folgenden Bedrohungen und Angriffspunkte.

Verstecktes Informationsvermögen: Bei der Identifizierung der Risiken in Verbindung mit den in einem Unternehmen vorhandenen Informationen und deren Vermögenswert werden meist nur elektronische Daten und sämtliche gedruckte Dokumente in Betracht gezogen. Auf das Wissen mancher Mitarbeiter, das in vielen Fällen mindestens gleich oder wenn nicht sogar noch wichtiger und unverzichtbarer ist, wird dabei oft vergessen. Dazu zählt

[1] `http://www.iso.org/iso/iso27001`

z.B. das Wissen jenes Mitarbeiters, welcher ein kritisches IT-System verwaltet oder eine für das Unternehmen essentielle Software implementiert hat. Wenn zusätzlich die Dokumentation dazu nur rar oder im schlimmsten Fall gar nicht vorhanden ist, steigt die Wichtigkeit bzw. der Wert dieses Wissens dementsprechend. Die Absicherung dieser Art von Informationen ist äußerst schwierig, da das Maß an Kontrolle und die Möglichkeiten einer logischen Zugangs- und einer physischen Zutrittsüberwachung äußerst beschränkt sind. In Anbetracht dessen kann sich ein Unternehmen somit in einer überaus Besorgnis erregenden Lage befinden (wenn auch nicht immer bewusst), welche sich der Social Engineer natürlich zu seinen Gunsten zunutze zu machen versucht. Ihn trennt oft nur eine geschickt eingefädelte Täuschung der entscheidenden Know-How-Träger vom Erhalt dieses Wissens.

Risiko durch Dritte: Viele Unternehmen unterschätzen das Risiko, welches entsteht, wenn Dritte Zugriff auf ihre Informationen haben. Die Relevanz dieser Bedrohung kommt besonders dann zum Tragen, wenn Arbeitstätigkeiten outgesourct werden und sich in diesem Zuge fremde Arbeiter im Unternehmen befinden. In diesem Fall ist es für einen Social Engineer relativ einfach sich einerseits als fremder Mitarbeiter auszugeben, um so Zutritt zu Gebäuden oder Zugang zu Informationen zu erhalten, oder andererseits diesen Arbeitern direkt Informationen zu entlocken. Des Weiteren versuchen viele Unternehmen heutzutage durch die Umsetzung von international anerkannten Prozessen und Standards ihre Arbeitspraktiken nachweislich zu optimieren und hoffen somit auf einen nachhaltigen Wettbewerbsvorteil gegenüber anderen. Die Nebenerscheinungen bzw. Voraussetzungen dafür sind regelmäßige Audits, welche ebenfalls einen hervorragenden Angriffspunkt für einen Social Engineer darstellen. Da die externen Auditors in vielen Fällen Zugriff auf sämtliche Informationen des Unternehmens bekommen, stellen sie eine fabelhafte (falsche) Identität für einen Angreifer dar.

Personal: Von der Personalabteilung kann ebenfalls ein hohes Maß an Bedrohung ausgehen, weil diese meist im Zuge einer Stellenbesetzung für Personenüberprüfungen zuständig ist. In vielen Fällen wird eine umfangreiche Erhebung der Identität und des Hintergrundes eines Bewerbers meist aus Kostengründen nur bei höherrangigen Positionen (wie z.B. Manager) durchgeführt. Jedoch korreliert der Rang einer Position nicht unbedingt mit dem Ausmaß an Zugriff auf wichtige und kritische Unternehmensinformationen. Beispielsweise kann der Informationszugang eines Nachwuchstechnikers der IT-Abteilung weitaus umfangreicher als jener eines Senior Managers sein. Weiters wird auch oft dem beruflichen Hintergrund bzw. der beruflichen Laufbahn des Bewerbers im Falle eines brillierendem Wissens nur eine mangelhafte Aufmerksamkeit geschenkt. Genau diese „Mängel“ bzw. „Schwächen“ kann ein Angreifer ausnutzen, um so Zugang zu einem Unternehmen und dessen Informationen zu erlangen.

Physische Zutrittskontrolle: Die meisten Sicherheitsstandards (wie unter anderem auch der ISO/IEC 27001 Standard) legen ihren Fokus bei der physischen Zutrittskontrolle ausschließlich auf hardwaretechnische Maßnahmen, wie z.B. Türschlösser, Alarmanlagen oder Überwachungskamerasysteme. Doch die eigentliche Schwachstelle stellt nicht die technische Komponente einer derartigen Zutrittskontrolle dar, sondern die menschliche Komponente, welche diese bedient bzw. mit dieser interagiert. Dadurch wird auch die gängigste physische Sicherheitsbarriere durch einen Social Engineer angreif- und auch relativ einfach überwind-

bar. Entgegen der naheliegenden und weitverbreiteten Ansicht, erhöht die Präsenz einer menschlichen Komponente (wie z.B. eines Sicherheitsbeamten oder eines Wachmannes) die Erfolgschancen der Erlangung eines unberechtigten Zutritts. Dadurch kann der Social Engineer stets eine Situation geschickt einfädeln, so dass ihm trotz fehlender Berechtigung der Zutritt zu einem Gebäude gewährt wird. Ohne einer menschlichen Komponente sind seine Chancen hingegen stark begrenzt und er ist auf andere Sicherheitslücken bei seinem Gelingen angewiesen (wie z.B. die Möglichkeit, die Barriere zu überspringen oder das Vorhandensein von nicht abgesicherten Eingängen).

Logische Zugriffskontrolle: In vielen Unternehmen weist die logische Zugriffskontrolle äußerst Besorgnis erregende Schwächen und Mängel sowohl in Bezug auf das gesamte Modell dahinter als auch auf die laufende Verwaltung sämtlicher Zugriffsrechte auf. Die ständig steigenden Datenmengen und das damit verbundene rasche Systemwachstum führen in vielen Fällen zu unstrukturierten und nur schwer wartbaren Systemen. Weiters benötigen Mitarbeiter für ihre tägliche Arbeit Zugriff auf immer mehr Informationen, was wiederum in einem viel höheren Anteil an Anfragen für die Gewährung von Zugriff auf mehr Informationen verglichen zu Anfragen für das Entziehen von Zugriffsrechten resultiert. In Folge wird der Umgang mit Zugriffsberechtigungen mit der Zeit immer nachlässiger, was wiederum eine effektive Zugriffskontrolle nahezu unmöglich macht. In diesem Fall muss der Angreifer im Prinzip nur das schwächste menschliche Glied der Sicherheitskette finden, um so Zugang zu kritischen Unternehmensdaten zu erhalten.

Heimarbeiter: Heimarbeiter stellen in zweierlei Hinsicht hervorragende Ziele für einen Social Engineer dar. Einerseits kann sein Angriff direkt auf einen Heimarbeiter abzielen. Da die Sicherheitsvorkehrungen (wie z.B. Firewalls) zu Hause üblicherweise nicht derart ausgeprägt wie im Unternehmen sind, kann er beispielsweise einen Heimarbeiter zum Öffnen eines schadhaften E-Mail-Anhanges animieren, um so vertrauliche Informationen auszuspähen. Oder er kann die mangelnde Präsenz eines Langzeit-Heimarbeiters im Unternehmen ausnutzen, in dem er sich im Zuge eines Telefonates als neuer Mitarbeiter ausgibt und um dringende Informationsauskünfte bittet. Andererseits kann er sich jedoch im Zuge eines Angriffes dem Unternehmen gegenüber selbst als Heimarbeiter ausgeben. Hierbei stellen vor allem Mitarbeiter des Helpdesks einen guten Angriffspunkt dar, da sie in der Regel stets bestrebt sind, den zu Hause arbeitenden Personen zu helfen.

2.1.5 Modelle zur Beschreibung eines Social Engineering Angriffes

In der Literatur gibt es zahlreiche verschiedene Modelle und Taxonomien, welche einen Social Engineering Angriff beschreiben. In dieser Arbeit wird das Modell von Mitnick und Simon [MiSi06] sowie das Modell von Nohlberg und Kowalski [NoKo08] vorgestellt.

Der Angriffszyklus

Das wohl bekannteste Modell stammt von Mitnick und Simon [MiSi06] und unterteilt einen derartigen Angriff in die vier folgenden vorhersagbaren Schritte bzw. Phasen: Recherche, Entwicklung von Beziehung und Vertrauen, Ausbeutung von Vertrauen und Nutzung von Informationen [MMLV14]. Diese werden in der Literatur als Angriffszyklus beschrieben und sind in Abbildung 2.1 zur Veranschaulichung dargestellt.

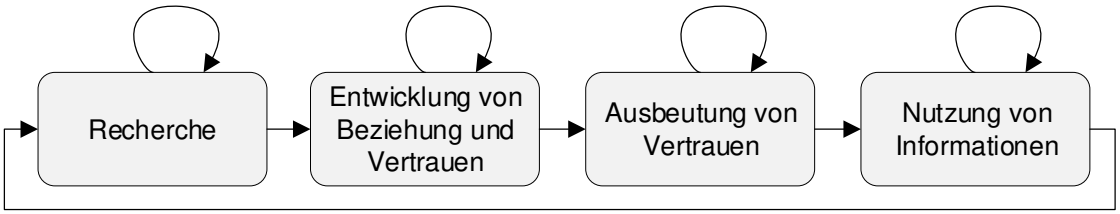

Abb. 2.1: Social Engineering Angriffszyklus (vgl. [MiSi06][Nohl08][MMLV14])

Abhängig vom jeweiligen Angriff und der anvisierten Zielperson kann dieser Zyklus oder auch einzelne Phasen davon mehrmals durchlaufen werden. Wenn der Angreifer entweder erwischt wird, aufgibt oder mit dem derzeitigen Ergebnis zufrieden ist, kann er in die nächste Phase übergehen oder den Angriff beenden [SEF]. Im Folgenden werden die einzelnen Phasen kurz beschrieben.

Recherche: Zu Beginn eines Angriffes erfolgt zunächst ein Prozess der Informationsbeschaffung. Hierbei versucht der Angreifer so viele Informationen wie möglich über seine Zielperson(en) zu sammeln [MMLV14]. Da der weitere Verlauf und schlussendlich auch der Erfolg des Angriffes im Wesentlichen von dieser Phase abhängig sind, wird ihr auch das Hauptaugenmerk und somit auch die meiste Zeit und der meiste Aufwand gewidmet [SEF].

Um an Informationen der Zielperson oder des angepeilten Unternehmens zu gelangen, stehen dem Angreifer hier eine Reihe von verschiedenen Methoden und Techniken zur Verfügung (siehe Kapitel 2.2). Dabei kann jede noch so scheinbar unwichtige Information dem Angreifer auf seinem Weg behilflich sein, unabhängig ob diese vom Hausmeister oder vom Geschäftsführer des jeweiligen Unternehmens stammt. Denn durch die Kombination von mehreren Informationen aus unterschiedlichen Quellen kann er trotzdem in der Lage sein, z.B. einen Zugriff auf sensible Unternehmensdaten oder - ressourcen zu erlangen oder sich einen umfangreichen Überblick über die Schwachstellen eines Systems zu verschaffen [SEF]. Die Informationsquellen können hierbei unterschiedlicher Natur sein. Angefangen von öffentlich zugänglichen Quellen, wie z.B. Telefonbücher, Zeitungen, Zeitschriften, Webseiten, Broschüren, usw. kann die Suche bis hin zu vergangenen Angriffen reichen [MiSi06][Nohl08].

Entwicklung von Beziehung und Vertrauen: Im nächsten Schritt versucht der Angreifer eine Beziehung und in Folge ein Vertrauensverhältnis zum Opfer aufzubauen. Dabei nutzt er die in Kapitel 2.1.3 beschriebenen menschlichen Schwächen und Tendenzen aus [Nohl08]. Des Weiteren kann hier die Nutzung von Insiderinformationen, die falsche Darstellung von anderen Identitäten, der Verweis auf dem Opfer bekannte Personen, die Vortäuschung eines Hilfebedarfs oder das Schlüpfen in eine autoritative Rolle dem Angreifer zu seinem Ziel verhelfen [MiSi06]. Auch diese Phase stellt einen äußerst kritischen Punkt in Bezug auf den Erfolg des Angriffes dar. Umso inniger und vertrauter die aufgebaute Beziehung zum Opfer ist, desto eher ist das Opfer auch bereit mit dem Angreifer zusammenzuarbeiten und ihn somit (wenn auch unbewusst) bei der Erreichung seines eigentlichen Zieles zu unterstützen. Der dazu notwendige Aufwand kann hier stark variieren und ist vom jeweils gewünschten Resultat abhängig. Oft ist schon eine höfliche Geste in Kombination mit einem freundlichen Lächeln ausreichend, damit die Zielperson dem Angreifer z.B. eine abgesicherte Tür offen

hält und ihn somit Zutritt zu einem Gebäude verschafft. Jedoch kann diese Phase dem Angreifer auch ein weitaus anspruchsvolleres Bemühen abverlangen – wenn er sich im Zuge dessen beispielsweise ein gefälschtes Profil in einem sozialen Netzwerk anlegen muss, um so online eine Beziehung zu seinem Opfer aufbauen zu können [SEF].

Ausbeutung von Vertrauen: Wenn es dem Angreifer gelang, eine Beziehung zum Opfer aufzubauen, kann er diese in der nächsten Phase mit Hilfe der im ersten Schritt gesammelten Informationen in irgendeiner Form missbrauchen oder ausnutzen. Hier versucht er dem Opfer Informationen (wie z.B. Kreditkartennummern, Passwörter oder andere geheime Daten) zu entlocken – möglichst ohne dabei einen Verdacht zu erregen [Nohl08][SEF]. Dies kann entweder in Form von einer einfachen Informationsanfrage, einer Bitte eine bestimmte Handlung auszuführen oder einer derartigen Manipulation, dass das Opfer den Angreifer um Hilfe bittet, erreicht werden [MiSi06][MMLV14]. Im Social Engineering Framework [SEF] werden unter anderem noch die folgenden Beispiele einer erfolgreichen Ausbeutung von Vertrauen genannt:

- Dem Angreifer eine Tür offen halten oder anderweitig Zutritt gewähren
- Benutzername und Passwort preisgeben
- Ein schadhaftes Programm mit Hilfe eines USB-Sticks auf einen Firmenrechner laden
- Einen schadhaften E-Mail-Anhang öffnen
- Geschäftsgeheimnisse in einer Diskussion mit einem vermeintlichen Kollegen offenbaren

Nutzung von Informationen: In der letzten Phase werden sämtliche Informationen zur Erreichung des „Endziels“ des Angriffes genutzt. Sollte dieses Ziel zu diesem Zeitpunkt noch nicht erreicht werden können (weil z.B. die gesammelten Informationen dazu noch nicht ausreichen), kann der Angreifer hier in eine vorherige Phase zurückkehren oder den gesamten Zyklus noch einmal durchlaufen [MiSi06][Nohl08][MMLV14]. Ansonsten wird hiermit der Angriff nach Möglichkeit derart beendet, dass das Opfer bezüglich der tatsächlichen Geschehnisse in Unwissenheit verweilt. Denn befindet sich das Opfer in dem Glauben, etwas Gutes für jemand anderen getan zu haben, bleibt für den Angreifer die Option einer zukünftigen Fortsetzung oder Erweiterung des Angriffes bestehen. Daher wird noch eine gut geplante und durchdachte Ausstiegsstrategie angestrebt, im Zuge derer alle digitalen Spuren beseitigt und das Hinterlassen von Informationen oder Gegenständen, welche auf einen Angriff rückschließen lassen, vermieden werden [SEF].

Der Zyklus der Täuschung

Nohlberg und Kowalski [NoKo08] erweitern und verfeinern mit ihrem konzeptuellen Modell den zuvor beschriebenen Angriffszyklus von Mitnick und Simon. Es besteht aus drei verschiedenen Zyklen: den Angriffs-, den Verteidigungs- und den Opferzyklus, welche in einem einzigen Zyklus vereinigt werden. Im Folgenden werden die Zyklen kurz beschrieben.

Der Angriffszyklus: Der Angriffszyklus befasst sich mit dem Verhalten und den Vorgehensweisen des Angreifers während eines derartigen Angriffes. Abbildung 2.2 illustriert die einzelnen Phasen dieses Zyklus.

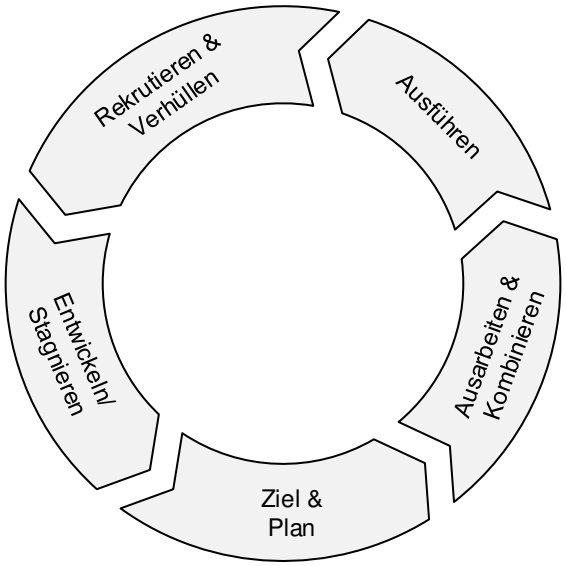

Abb. 2.2: Der Angriffszyklus (vgl. [NoKo08])

Ziel & Plan: Ein Angreifer will mit einem Angriff grundsätzlich einen bestimmten Zweck erreichen. Dementsprechend verfolgt er ein bestimmtes Ziel und einen Plan, wie er dieses Ziel erreichen kann. Damit der Angriff auch wirklich erfolgreich ist, muss der Angreifer die drei Eigenschaften Methode, Gelegenheit und Motiv besitzen (vgl. [PfPf15]). Unter Methode versteht man dabei die Fähigkeiten, das Wissen, die Werkzeuge und alle anderen Maßnahmen, welche für die Vorbereitung des Angriffes notwendig sind. Gelegenheit ist der optimale Zeitpunkt für die Durchführung des Angriffes und das Motiv ist die Motivation bzw. der Beweggrund des Angreifers hinter seinem Vorgehen. Jede dieser drei Eigenschaften ist für einen erfolgreichen Angriff unabdingbar – sollte auch nur eine fehlen, wird das Vorhaben scheitern. *Ausarbeiten & Kombinieren:* In dieser Phase versucht der Angreifer an sämtliche Informationen zu gelangen, welche für den Angriff benötigt werden. Dazu wendet er entweder gewöhnliche Social Engineering Methoden und Techniken (siehe Kapitel 2.2) an oder nutzt die menschlichen Tendenzen zur Manipulation von Personen (siehe Kapitel 2.1.3) aus. *Ausführen:* In diesem Schritt führt der Angreifer eine illegale oder nicht erlaubte Handlung aus, wie z.B. nach Anmeldedaten zu fragen oder schadhafte E-Mails zu versenden. *Rekrutieren & Verhüllen:* Zu dieser Phase zählen sämtliche Aktivitäten, welche der Angreifer zur Verbergung der illegalen Handlungen des vorherigen Schrittes vornimmt. Darunter fällt z.B. das normale Weiterführen der „Freundschaft“ zum Opfer oder das Opfer als nicht vertrauenswürdig erscheinen zu lassen. *Entwickeln/Stagnieren:* In dieser Phase lernt der Angreifer vom bisherigen Prozess und rechtfertigt seine Taten innerlich. Wenn der bisherige Verlauf erfolgreich war, kann er hier den Angriff weiterführen bzw. weiterentwickeln und einen neuen Zyklus beginnen. Sollte dies nicht der Fall sein, kann er den Angriff entweder beenden oder in eine vorherige Phase zurückkehren und sein Glück erneut versuchen.

Der Verteidigungszyklus: Der Verteidigungszyklus beschreibt die grundsätzlich möglichen Optionen für den Verteidigenden. In vielen Fällen ist der Verteidiger gleichzeitig auch das

Opfer, es kann sich dabei jedoch beispielsweise auch um einen Sicherheitsexperten eines Unternehmens handeln.

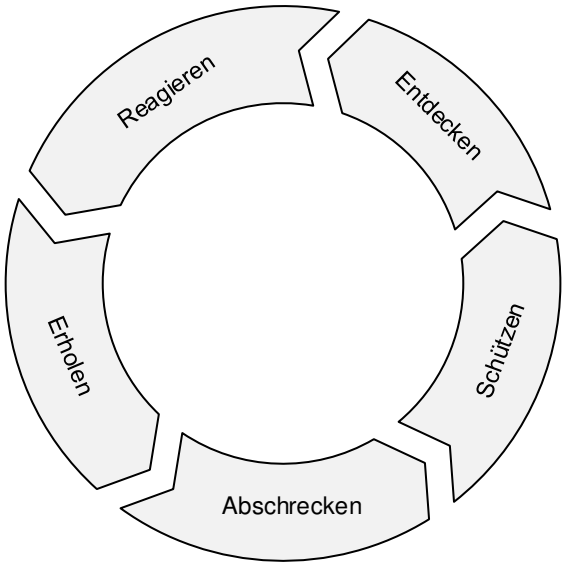

Abb. 2.3: Der Verteidigungszyklus (vgl. [NoKo08])

In Abbildung 2.3 sind einige Optionen, welche der Verteidigende zur Gegenwirkung eines Angriffes unternehmen kann, grafisch in Form eines Zyklus veranschaulicht. Zunächst kann ein Angriff durch das Vorhandensein einer bewährten und öffentlich bekannten Verteidigungspolitik oder eines allgemein bekannten Rufes, sämtliche Vorfälle bei den entsprechenden Sicherheitsbehörden zu melden, *abgeschreckt* werden. Weiters kann man das Unternehmen *schützen*, indem man möglichst keine sensiblen Daten öffentlich zugänglich macht, die Mitarbeiter ständig bezüglich der potentiellen Risiken und Methoden der Angreifer schult und auch klare Richtlinien definiert, wie bei einem Vorfall reagiert bzw. vorgegangen werden muss. Ebenfalls kann einerseits durch die Überwachung des Netzwerkverkehrs und andererseits durch angemessen ausgebildete Mitarbeiter, welche unangebrachte und auffällige Fragen auch als solche erkennen, ein Angriff *entdeckt* werden. In dem im Unternehmen den Mitarbeitern die Möglichkeit der Meldung von Social Engineering Vorfällen angeboten und ihnen die manipulierenden Vorgehensweisen eines Angreifers bewusst gemacht werden, kann auf einen erkannten Angriff entsprechend *reagiert* werden. Zuletzt kann sich ein Unternehmen von einem Angriff wieder *erholen* bzw. sich von den dadurch gesammelten Erfahrungen bereichern, in dem sämtliche Angriffe dokumentiert, adäquate Richtlinien und Vorgehensweisen definiert sind sowie der Wert von jeglichen Informationen bekannt ist.

Der Opferzyklus: Der Fokus von diesem Zyklus ist auf das Verhalten des Opfers eines Angriffes gerichtet. Häufig wird bei der Aufarbeitung bzw. Analyse eines derartigen Verbrechens das Augenmerk ausschließlich auf den Angreifer gerichtet. Würde man jedoch dem Opfer mehr Aufmerksamkeit schenken, könnten einige Vorfälle leicht vermieden werden. Abbildung 2.4 stellt den Ablauf dieses Zyklus dar.

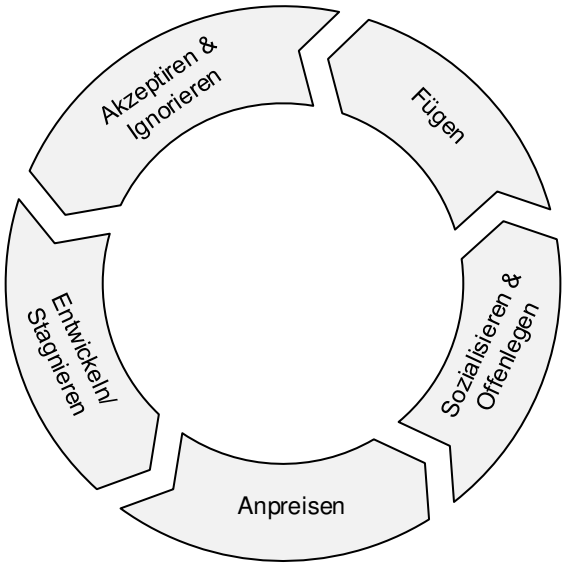

Abb. 2.4: Der Opferzyklus (vgl. [NoKo08])

Viele spätere Opfer prahlen mit dem Besitz von etwas Wertvollen und machen auch kein Geheimnis daraus. Dadurch *preisen* sie förmlich ihre Eignung als mögliches Opfer *an*. In dem sich das Opfer mit dem Angreifer *sozialisiert*, macht es sich selbst anfällig gegen Manipulationen und Täuschungen und durch das *Offenlegen* von Wertgegenständen werden diese für den Angreifer leicht zugänglich. Wenn dann anschließend die eigentliche Straftat vollzogen wird, *fügt* sich das Opfer, in dem es beispielsweise geheime Informationen preisgibt. Danach kann die Zielperson die vollzogene Straftat entweder *akzeptieren* (z.B. in dem sie glaubt, dass sie nicht so schwerwiegend bzw. ernstzunehmend war) oder (bewusst oder unbewusst) *ignorieren*. Schlussendlich kann sich das Opfer nach einem Angriff *entwickeln* und sich somit in Zukunft nicht mehr so leicht ausbeuten lassen, oder es kann auch *stagnieren* und die Rolle des Opfers auch zukünftig einfach hinnehmen.

Der Zyklus der Täuschung: Vereinigt man diese drei Zyklen zu einem einzigen und fügt in der Mitte das eigentliche Ziel hinzu, ergibt sich eine ganzheitliche Darstellung aller Voraussetzungen für einen Social Engineering Angriff. Diesen Zyklus beschreiben Nohlberg und Kowalski [NoKo08] als den Zyklus der Täuschung (engl. The Cycle of Deception, siehe Abbildung 2.5).

Für das Gelingen des Angriffes müssen dabei sämtliche Schritte bzw. Phasen aller Zyklen übereinstimmen und nahtlos ineinander übergehen. Der Angreifer muss zumindest die ersten drei Schritte des Angriffes erfolgreich abwickeln. Denn ohne Plan und Methode, ohne Informationen über das potentielle Opfer oder ohne der Möglichkeit den Angriff auszuführen, wird dieser mit hoher Wahrscheinlichkeit fehlschlagen. Will er sich zusätzlich die Option, den Angriff auch in Zukunft weiterführen zu können, offen halten, so ist er ebenfalls auf das Gelingen in den Schritten vier und fünf angewiesen. Denn wenn er nicht in der Lage ist den Angriff zu verbergen, wird er höchstwahrscheinlich erwischt werden, und wenn er selbst den Angriff nicht als eine positive Erfahrung empfindet, wird er ihn ebenfalls nicht weiterführen. Gleichermaßen trifft dies auch auf den Verteidigungszyklus

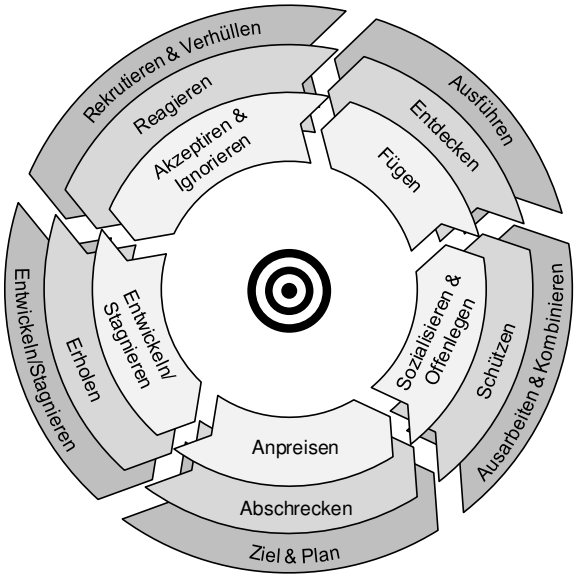

Abb. 2.5: Der Zyklus der Täuschung (vgl. [NoKo08])

zu. Sollte auch nur eine einzige Option darin gut genug sein den Angriff zu stoppen, wird er offensichtlich fehlschlagen oder der Angreifer zumindest erwischt werden. Wenn andererseits wiederum die Handlungen des Verteidigenden nicht die gewünschten Resultate erzielen, kann der Angriff nicht unterbunden werden. Betrachtet man zuletzt den Opferzyklus, so muss sich die Zielperson auch hier in der Regel in allen Schritten und Belangen dem Angreifer fügen, damit der Angriff schlussendlich erfolgreich ist.

2.2 Angriffe und Methoden

Um eine Einzelperson oder ein gesamtes Unternehmen anzugreifen, kann ein Social Engineer auf zahlreiche verschiedene Methoden und Ansätze zurückgreifen. Dabei können die erforderlichen Voraussetzungen und Fähigkeiten durchaus unterschiedlich sein: Während einige davon technische Fähigkeiten voraussetzen, erfordern andere wiederum die Soft Skills zur Manipulation von Personen. Einige können aus der Ferne via Internet durchgeführt werden, andere wiederum nur persönlich an einem bestimmten Ort. Für manche wird kein spezielles Equipment benötigt, während andere hingegen eine anspruchsvolle Ausstattung voraussetzen [SEF].

In der Literatur lassen sich verschiedene Kategorien und Taxonomien der vorhandenen Methoden und Techniken (oft auch als Angriffsvektoren bezeichnet) finden. Während Krombholz et al. [KHHW15] und Fan et al. [FaLR17] jeweils auf vier Kategorien setzen (Erstere physisch, technisch, sozial und sozio-technisch bzw. Letztere physisch, technisch, sozial und hybrid), beschreibt Nohlberg [Nohl08] hingegen nur drei Kategorien (physisch, sozial und eine Kombination aus technisch und sozial). Da die Grenzen hier jedoch fließend sind und eine eindeutige Unterteilung somit nur schwer definierbar ist, wird in dieser Arbeit eine Kombination aus [KHHW15] und [FaLR17] mit Fokus auf Letztere beschrieben.

2.2.1 Physische Ansätze

In diese Kategorie fallen sämtliche Ansätze, welche zur Erlangung der gewünschten Informationen eine körperliche Aktivität des Social Engineers erfordern [KHHW15]. Die bekanntesten und gängigsten physischen Angriffe bzw. Methoden werden nachfolgend kurz erläutert.

Dumpster Diving

Unter Dumpster Diving versteht man das Durchwühlen des Mülls einer Person oder eines Unternehmens, um so an sensible Informationen zu gelangen [LoMi11][SEF]. Das Ausmaß der darin gefundenen Daten ist in vielen Fällen äußerst Besorgnis erregend und kennt keine Grenzen. Angefangen von persönlichen Fotos, E-Mails, Lebensläufen, Bankauszügen, medizinischen Befunden bis hin zu Informationen über diverse Benutzerkonten, verwendete Softwares, ausgedruckte Logdateien und vieles mehr – nahezu alles kann im Abfall gefunden werden. Wie auch bei allen weiteren Formen von Social Engineering versucht ein Angreifer durch eine intelligente Vorgehensweise den notwendigen Aufwand möglichst gering zu halten. Ein Passwort mittels eines Brute-Force-Angriffes stundenlang herauszufinden versuchen ist ziemlich unklug, wenn es einfach von einem weggeworfenen Post-it aus einem Mülleimer bezogen werden kann [SEF].

Doch heutzutage ist sogar das Durchwühlen nicht einmal mehr zwingend notwendig. Bei vielen Abfalllagerorten hängen wichtige Dokumente beim Mülleimer heraus oder liegen gar nebenan am Boden, oder es werden durchsichtige Müllsäcke verwendet, wodurch dem Angreifer das Ausspähen von Daten ebenfalls erheblich erleichtert wird. Vielen Privatpersonen als auch Mitarbeitern in einem Unternehmen sind dabei die möglichen Auswirkungen von einfach weggeworfenen sensiblen Informationen nicht bewusst. Demzufolge werden auch häufig wichtige Dokumente trotz entsprechender Kennzeichnungen, wie z.B. „nur für den internen Gebrauch“ oder „streng vertraulich“, ohne jegliche Vorkehrungen (wie z.B. zuvor schreddern) einfach in den Mülleimer geworfen und stellen schlussendlich eine leichte Beute für einen Social Engineer dar [LoMi11].

Shoulder Surfing

Shoulder Surfing bezeichnet die direkte Beobachtung von Personen, um so an sicherheitsrelevante Informationen zu gelangen. Dazu zählt z.B. das Spähen über die Schulter, auf den Bildschirm oder auf die Tastatur einer anderen Person [KHHW15]. In vielen Fällen ist dazu erst gar nicht das Handwerk eines Social Engineers und dessen speziellen Wissen und Techniken erforderlich. Denn zahlreiche Informationen lassen sich bereits durch einen genaueren Blick auf das verwendete Gerät der Zielperson selbst erkennen. So kann z.B. eine fachlich versierte Person anhand der Bauform und der sichtbaren Schnittstellen bereits das Alter und Modell des Gerätes feststellen. Ein Hacker würde zusätzlich in einem günstigen Moment einen Blick auf den Bildschirm des Gerätes werfen, um so möglicherweise mehr über das verwendete Betriebssystem, die installierten Anwendungen und Ähnliches zu erfahren. Ein Social Engineer würde hingegen vor allem Ausschau nach aufgeklebten Visitenkarten oder Stickers bzw. Post-its halten, deren Informationsgehalt oft weitaus höher ist. Damit können unter anderem Daten wie Name, Adresse, Arbeitgeber, Berufsbezeichnung, Telefonnummer oder sogar Anmeldedaten enthüllt und damit ein Profil des Opfers erstellt werden. Durch diese Vorgehensweise kann der Angreifer erste Informationen bereits ohne großen Aufwand und ohne jegliches rechtswidriges Verhalten sammeln [LoMi11].

Es gibt zahlreiche verschiedene Schauplätze eines derartigen Angriffes, doch manche sind besser geeignet und sollten daher besonders berücksichtigt werden. Dazu zählen sämtliche Orte, die

über ein Terminal verfügen und somit Benutzereingaben erfordern. Beispiele dafür sind Kassen in Geschäften, wo der Mitarbeit seine Zugangsdaten eingeben muss oder Bankomaten, wo die Eingabe eines PINs notwendig ist. Generell ist die potentielle Gefahr eines Shoulder Surfing Angriffes an öffentlichen Orten, wie Flughäfen, Coffeeshops, Lounges oder Restaurants/Bars besonders erhöht [LoMi11].

2.2.2 Technische Ansätze

Technische Ansätze stehen in Verbindung mit bestimmten technischen Maßnahmen, welche meist über das Internet durchgeführt werden. Das Internet entpuppt sich dabei als eine hervorragende Informationsquelle für einen Social Engineer, unter anderem weil vielen Menschen die möglichen Folgen der über dieses Medium preisgegebenen Daten nicht bewusst sind. Neben Webseiten zählen heutzutage vor allem soziale Netzwerke zu den beliebtesten Anlaufquellen für einen Angreifer, da diese über massenweise persönliche Informationen verfügen. Zur Sammlung und Aggregation dieser Daten aus unterschiedlichen im Web verfügbaren Quellen kann zusätzlich auf eine Reihe von Hilfsmitteln und Tools, wie z.B. Suchmaschinen oder Datensammlungstools, zurückgegriffen werden [KHHW15]. Angriffe und Methoden, welche dieser Kategorie zugehören, werden nun näher betrachtet.

Phishing

Beim Phishing und seinen verschiedenen Ausprägungen (siehe Kapitel 3) versucht der Angreifer an sensible Daten zu gelangen oder das Opfer zu einer gewünschten Handlung zu animieren, in dem er sich in einem elektronischen Kommunikationsverkehr als vermeintlich vertrauenswürdige Instanz ausgibt. Typischerweise kann ein derartiger Angriff über die verschiedensten Kanäle (wie z.B. körperliche Anwesenheit, E-Mail, Webseiten, soziale Netzwerke, Cloud, usw.) durchgeführt werden und kann für nur ein Einzelziel (eine Person oder ein Unternehmen) bis hin zur breiten Masse als Ziel konzipiert sein [KHHW15][FaLR17]. Nähere Details bezüglich Phishing, den verschiedenen Formen davon und den möglichen Gegenmaßnahmen dazu werden in Kapitel 3 beschrieben.

Waterholing

Das Konzept hinter einem Waterholing-Angriff ist vergleichbar mit einem Raubtier, das in der Wüste bei einem Wasserloch wartet. Das Raubtier weiß, dass eine potentielle Beute auf der Suche nach Wasser möglicherweise das Wasserloch aufsuchen wird. Anstatt jedoch selbst auf die Jagd zu gehen, wartet es stattdessen am Wasserloch und somit bis die Beute zu ihm kommt. Gleichermaßen kompromittiert der Angreifer bei diesem Angriff zunächst eine für die Zielperson(en) relevante bzw. regelmäßig besuchte Webseite. Dazu infiziert er diese mit einer beliebigen Malware, wartet anschließend bis seine Opfer die manipulierte Webseite besuchen und hofft, dass in diesem Fall auch das jeweils verwendete Gerät erfolgreich infiziert wird [O'Mc12][KHHW15].

Voraussetzung für diesen Angriff ist immer eine entdeckte Schwachstelle der zu manipulierenden Webseite. Erst dadurch ist der Angreifer in der Lage JavaScript- oder HTML-Code in den vorhandenen Quellcode der Webseite einzuschleusen. Dieser Code beinhaltet üblicherweise einen Verweis (in Form eines Links oder iFrames) auf jene Webseite, welche den tatsächlichen Exploitcode zur Ausnutzung der entdeckten Schwachstelle hostet. Sobald ein Benutzer anschließend auf die manipulierte Webseite zugreift, wird er automatisch auf die schadhafte Seite weitergeleitet.

Dort wird dann der Schadcode ausgeführt, welcher dem Angreifer schließlich die Installation von Malware am Gerät des Opfers ermöglicht [O'Mc12].

Solche Injection-Angriffe auf Webseiten sind keine Seltenheit und sind allgemein in der Computerkriminalität gang und gäbe. Der Unterschied bei Waterholing-Angriffen liegt jedoch in der Auswahl der zu kompromittierenden Webseiten. Bei gewöhnlichen Injection-Angriffen wird meist versucht, so viele willkürliche Seiten wie nur möglich zu infizieren, während Waterholing-Angriffe stattdessen weitaus gezielter ausgerichtet sind. Hier werden meist nur Webseiten eines speziellen Bereichs oder eines bestimmten Unternehmens anvisiert, damit dementsprechend auch nur Personen aus diesem Bereich bzw. diesem Unternehmen davon betroffen sind. Das Abzielen auf eine bestimmte Webseite ist dabei weitaus schwieriger als das Finden von Seiten, welche eine bestimmte Schwachstelle aufweisen und erfordert daher eine gründliche Untersuchung bzw. Analyse. Häufig werden die Webseiten dabei bereits Monate vor dem eigentlichen Angriff kompromittiert und anschließend regelmäßig vom Angreifer besucht, um den Zugriff bzw. die Kontrolle sicherzustellen. Somit können mehrere Webseiten zugleich infiziert und dadurch ein Zero-Day-Exploit voll ausgenutzt werden [O'Mc12].

Waterholing-Angriffe werden immer häufiger zu wichtigen Vektoren bei Angriffen auf multinationale Unternehmen. Anstatt mit Hilfe von individuellen Phishing-Nachrichten Mitarbeiter eines Unternehmens direkt anzupeilen, legen die Angreifer den Fokus auf häufig besuchte Webseiten ihrer Zielpersonen. Auch diese Webseiten werden hierzu zuerst mit Malware infiziert und anschließend der Besuch eines Mitarbeiters abgewartet. So wurden z.B. bereits namhafte Unternehmen wie Apple und Facebook Opfer eines derartigen Angriffes. Dabei wurde ein beliebtes iOS Entwicklungsforum durch die Ausnutzung eines Java Zero-Day-Exploits infiziert. Als anschließend die Mitarbeiter der genannten Unternehmen dieses Forum besuchten, wurden ihre Geräte kompromittiert und somit bekamen die Angreifer Zugriff auf die internen Netzwerke der beiden Unternehmen [KHHW15].

2.2.3 Soziale Ansätze

Unter soziale Ansätze fallen sämtliche Methoden und Techniken, welche auf die menschlichen Schwächen abzielen und somit auf den im Kapitel 2.1.3 beschriebenen psychologischen Aspekten zur Manipulation von Personen beruhen. Weiters verlässt sich hierbei der Social Engineer auf die Gemeinsamkeiten, welche alle Menschen in ihrer sozialen Umgebung üblicherweise entwickelt haben sollten und versucht somit häufig eine Beziehung zu seinem Opfer aufzubauen. Die menschliche Neugier kann bei nahezu all diesen Angriffen als ein entscheidender sozialer Einflussfaktor hervorgehoben werden. Als Übertragungsmedium bzw. Kanal dieser Ansätze dient zu einem erheblichen Teil das Telefon [KHHW15]. Nachfolgend wird ein Überblick über die am weitesten verbreiteten sozialen Ansätze gegeben.

Reverse Social Engineering

Reverse Social Engineering ist eine Methode, bei welcher der Social Engineer eine Situation derart einrichtet, dass das Opfer mit einem bestimmten Problem konfrontiert wird und zu dessen Lösung die Hilfe des Angreifers in Anspruch nimmt [MiSi06]. Dabei bietet er der Zielperson zuerst seine Hilfe bei einem bestimmten Problem (wie z.B. bei einem Netzwerkverbindungsproblem) an und gibt sich im Zuge dessen beispielsweise als Mitarbeiter des Supports oder als fachlich kompetente Person in Bezug auf IT-Angelegenheiten aus (z.B. mittels Versenden von

E-Mails oder Verteilen von Visitenkarten). Anschließend täuscht er genau dieses Problem durch Manipulation vor und wartet daraufhin bis er vom Opfer kontaktiert wird [Kee08][Nohl08]. Da das Opfer in diesem Fall den Kontakt zum Social Engineer von sich aus aufnimmt, schenkt es ihm jegliche Glaubwürdigkeit und hat somit keine Zweifel bezüglich der von ihm vorgetäuschten Identität. Daher wird es in der Regel vom Angreifer auch keine personenbezogenen Angaben verlangen und ist ihm somit praktisch ausgeliefert [MiSi06][Kee08]. Ziele solcher Angriffe sind häufig fachlich nicht vertraute Personen, die sich aufgrund ihrer mangelnden Kenntnisse und ihrer Ahnungslosigkeit viel eher anleiten lassen. Sie sind einerseits anfälliger den gewünschten Forderungen des Angreifers nachzukommen (wie z.B. Installation einer Anwendung) und andererseits wissen sie auch häufig den Wert von bestimmten Daten oder Informationen nicht zu schätzen [MiSi06].

Bei einem erfolgreichen Angriff gibt das Opfer dem Angreifer bestimmte Informationen preis (wie z.B. Anmeldedaten oder Netzwerkinformationen), das absichtlich hervorgerufene Problem wird vom Social Engineer wieder behoben und der Zielperson ist weder die Bekanntgabe von vertraulichen Informationen noch die Gegenwärtigkeit eines Social Engineering Angriffes bewusst [Kee08].

Pretexting

Als Pretexting bezeichnet man den Akt der Erstellung und Ausnutzung eines erfundenen Szenarios („Pretext“ genannt), um die Zielperson zur Preisgabe von sensiblen Informationen oder zur Durchführung einer gewünschten Handlung zu animieren [HaWi10][FaLR17]. Doch dieser Ansatz geht weit über das Schlüpfen in eine andere Rolle oder das Erfinden von Lügen hinaus. In manchen Fällen wird dadurch sogar eine gesamte Identität komplett neu geschaffen und anschließend zur Ausbeutung des Opfers ausgenutzt. Oder der Social Engineer kann diese Methode ebenfalls zur Imitation einer anderen Person in einer bestimmten beruflichen Position oder anderweitigen Rolle verwenden. Die für den Angreifer möglichen Anwendungsszenarien sind hierbei nahezu grenzenlos, jedoch haben sie alle eines gemeinsam – sie erfordern eine Menge an Recherche. Denn ein perfekter und glaubwürdiger Pretext kann nur mit ausreichenden Informationen und einer ausgiebigen Planung geschaffen werden. Das Ausmaß des Erfolges des Angriffes korreliert hierbei mit der Qualität und Quantität der vorhandenen Informationen [HaWi10].

Wie generell beim Social Engineering zählt auch hier das Vertrauen zu den wichtigsten Aspekten. Gelingt es dem Angreifer nicht ein Vertrauensverhältnis zum Opfer aufzubauen, wird er in der Regel mit seinem Vorhaben scheitern. Wenn sein erfundenes Szenario oder seine vorgetäuschte Identität mangelhaft oder als nicht glaubwürdig erscheint, wird ihm das Opfer höchstwahrscheinlich auf die Schliche kommen. Vergleichbar mit dem Einfügen des richtigen Schlüssels in ein Schloss, kann der richtige Pretext sämtliche Verdächtigungen und Zweifel beseitigen und somit die Tür zum Erfolg öffnen [SEF].

2.2.4 Hybride Ansätze

Hybride Ansätze zählen zu den aussichtsreichsten Methoden und können sowohl physische, technische als auch soziale Aspekte beinhalten [FaLR17]. Die zwei bekanntesten Angriffe dieser Kategorie werden nachfolgend erläutert.

Baiting

Bei einem Baiting-Angriff lässt der Angreifer ein mit Malware infiziertes Speichermedium (wie z.B. CD, DVD oder USB-Stick) an einer Stelle zurück, an welcher das Opfer es auch höchstwahrscheinlich finden wird [KHHW15][RaNa14]. In der Hoffnung, dass die Zielperson das Speichermedium anschließend auch verwendet, strebt der Angreifer zunächst eine Infizierung des verwendeten Gerätes und in Folge auch eine Infizierung des gesamten Netzwerkes an. Ziel hierbei ist die Erlangung eines Zugriffs auf sämtliche Informationen des jeweiligen Gerätes als auch des gesamten Netzwerkes [RaNa14].

Üblicherweise fertigt der Angreifer vorerst einige Kopien der mit Malware infizierten Speichermedien an. Um das Interesse zu wecken, beschriftet er diese zusätzlich mit interessanten Namen und Bezeichnungen (wie z.B. „streng vertraulich“ oder „Mitarbeiter-Gehaltsliste-2017“) und platziert diese anschließend an besonders frequentierten Orten, wie z.B. Toiletten, Gängen, Rezeptionsbereichen, Aufzügen oder Parkplätzen. Dem menschlichen Trieb der Neugier folgend sind die Zielpersonen (entweder eine Einzelperson oder Mitarbeiter eines Unternehmens) in der Regel am Inhalt des Speichermediums interessiert und sobald sie es (ohne dass sie sich den Auswirkungen bewusst sind) in ihr Gerät einbringen, wird dieses von der Schadsoftware befallen und diese erledigt die restliche Arbeit für den Angreifer (wie z.B. das Weiterleiten der gewünschten Informationen an den Angreifer). Auch wenn das infizierte Speichermedium von einem sicherheitsbewussten Mitarbeiter gefunden und an die IT-Abteilung überreicht wird, kann der Angriff in vielen Fällen nicht unterbunden werden. Denn in der Regel werden sie auch dort den Datenträger analysieren und sobald er wiederum in ein Gerät eingebracht wird, wird auch dieses mit hoher Wahrscheinlichkeit infiziert. Die dazu verwendeten Schadprogramme werden üblicherweise derart durchdacht und listig implementiert, so dass ein gängiges Antivirenprogramm diese nicht erkennen kann [KHHW15][RaNa14].

Tailgaiting

Tailgaiting, auch bekannt als „Piggybacking“, bezeichnet einen Angriff, bei welchem der Angreifer ohne entsprechender Berechtigung erfolgreich einen Zutritt zu einem abgesicherten Gebäude erlangt [FaLR17]. In den meisten Fällen erreicht er dies, in dem er einfach einer autorisierten Person dicht folgt und den Anschein erweckt, als sei er ebenfalls ein Mitarbeiter des entsprechenden Unternehmens (beispielsweise durch das Tragen eines offensichtlichen, jedoch gefälschten Firmenausweises). Dabei hofft der Angreifer entweder, dass der jeweilige Mitarbeiter ihm aus Höflichkeit die Tür offen hält oder verwendet einen Pretext und täuscht somit einen falschen Vorwand vor (z.B. dass er seinen Ausweis vergessen oder verloren hat). Sobald der Angreifer einmal die physische Zutrittsbarriere überwunden hat und sich somit im Unternehmen befindet, stehen ihm jegliche Türen offen. Daraufhin ist er relativ einfach in der Lage weitere Angriffe, wie z.B. Baiting, Shoulder Surfing oder Dumpster Diving, durchzuführen und somit an essentielle Informationen zu gelangen. Auch hier nutzt der Social Engineer wiederum die menschliche Tendenz der Höflichkeit und Hilfsbereitschaft aus – wenn auch zwar im guten Glauben, jedoch in diesem Fall fahrlässig [RaNa14].

3 Phishing

Dieses Kapitel liefert detaillierte Informationen über das Thema Phishing und stellt zur Vermeidung einer zu tiefen Gliederung des Grundlagenkapitels 2 ein eigenständiges Kapitel dar. Zu Beginn wird ein Überblick über ein paar aktuelle Daten und Fakten gegeben und anschließend die typische Vorgehensweise bei einem Phishing-Angriff näher betrachtet. Daraufhin werden die bekanntesten Angriffskanäle und die verwendeten Methoden und Techniken dargestellt sowie auf die wichtigsten Angriffsvektoren eingegangen. Abschließend werden zur Vermeidung, Vorbeugung und Erkennung von derartigen Angriffen noch die möglichen Gegenmaßnahmen dazu vorgestellt.

3.1 Aktuelle Daten und Fakten

Die Anti-Phishing Working Group (APWG) veröffentlicht mit ihrem „Phishing Activity Trends Report“ quartalsmäßig die aktuellsten Zahlen und Fakten bezüglich der derzeitigen Phishing-Lage. Die Basis hierfür bilden sämtliche Phishing-Angriffe, die entweder von ihren Mitgliedsunternehmen, von globalen Forschungspartnern oder über ihre Webseite bzw. über E-Mail gemeldet wurden. Die Daten aus dem letzten Report über das vierte Quartal 2016 [AaMa17] sind dabei äußerst Besorgnis erregend. Mit einer Gesamtanzahl von 1.220.523 Phishing Angriffen im Jahr 2016 geht dieses Jahr als das katastrophalste seit Beginn ihrer Aufzeichnungen (2004) in die Geschichte ein. Dies entspricht einer Erhöhung von 65 % zum Vorjahr, wobei die Tendenz weiterhin steigt. Während es im vierten Quartal 2004 noch 1.609 Angriffe pro Monat waren, so stieg diese Zahl ebenfalls enorm und landete im vierten Quartal 2016 bei bereits 92.564, was einer Erhöhung von 5.753 % über 12 Jahre hinweg entspricht.

Sieht man sich die Statistik der meist anvisierten Industriesektoren aus dem vierten Quartal 2016 (siehe Abbildung 3.1) an, so sind von Phishing-Angriffen am häufigsten Unternehmen aus dem Einzelhandels- bzw. Dienstleistungssektor, gefolgt von Unternehmen aus dem Finanzsektor und Internetdienstanbieter (ISP) betroffen.

Die am häufigsten dazu verwendete Top-Level-Domain (TLD) war im vierten Quartal 2016 (Stand Dezember) .com mit 44,5 %, gefolgt von .net mit 3,54 %, .tk mit 3,27 % und .org mit 2,92 % (siehe Abbildung 3.2). Dabei wurden einige dieser Domains kompromittiert, in dem die Angreifer die jeweiligen Web-Server knackten und im Zuge dessen Phishing-URLs in die bestehenden Webseiten einschleusten. Die restlichen wurden hingegen einfach von den Angreifern neu registriert und in Folge als Phishing-Webseiten verwendet. Besonders auffällig hierbei ist, dass nur ein sehr geringer Anteil aller Phishing-Webseiten den Domainnamen möglichst gleich wie die Originalseite vorzutäuschen versuchen (wie z.B. *pay5al.com* oder *pay.pal.com* anstatt der legitimen Domain *paypal.com*). Dies wiederum belegt, dass ein erfolgreicher Angriff nicht zwin-

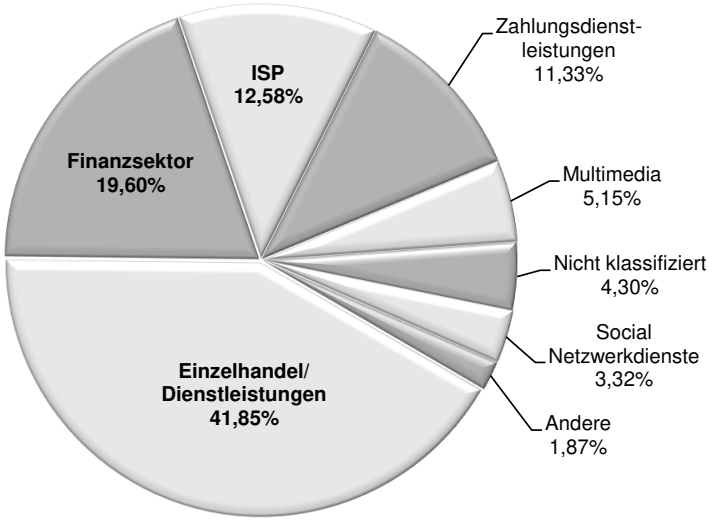

Abb. 3.1: Meist anvisierte Industriesektoren im vierten Quartal 2016 (vgl. [AaMa17])

gend auf trügerische Domainnamen angewiesen ist, sondern die Opfer mittels Hyperlinks (die die eigentliche Zieldomain verbergen), Kurz-URL-Diensten (welche die Zieldomain verschleiern) oder des Einfügens des legitimen Namens bzw. der legitimen Marke an einer beliebigen Stelle in der URL in die Irre geführt werden.

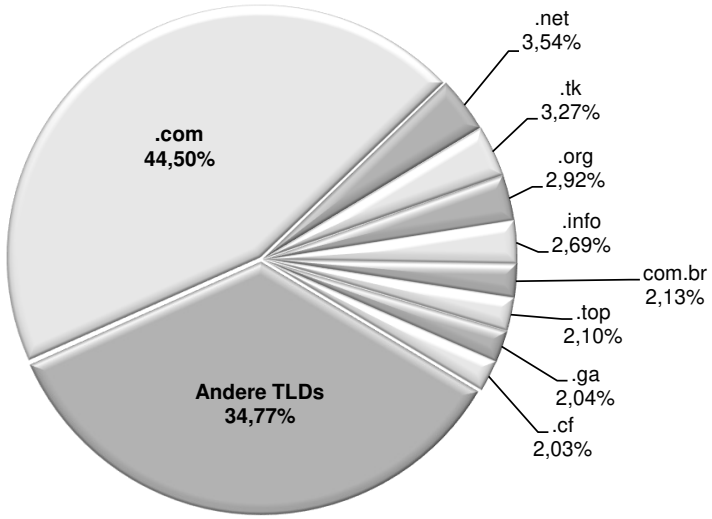

Abb. 3.2: Phishing-Domains nach TLD im Dezember 2016 (vgl. [AaMa17])

Laut dem Internet Security Threat Report [CCCL+17] war im Jahr 2016 unter 2.596 E-Mails jeweils eine Phishing-E-Mail enthalten. Darunter gewannen vor allem sogenannte Business E-Mail Compromise (BEC) Betrügereien, auch CEO-Betrug genannt, an Beliebtheit. Hierbei handelt es sich um Finanzbetrüge, bei welchen gefälschte E-Mails an Mitarbeiter eines Unternehmens gesendet werden. Unter dem Anschein, dass diese vom CEO oder Senior Management stammen, werden die Zielpersonen im Zuge dessen zu einem Geldtransfer aufgefordert. Diese Betrügereien können zu äußerst kritischen Auswirkungen für ein Unternehmen führen, da sie in der Regel nur wenig bis gar keine technische Expertise voraussetzen. Im ersten Halbjahr 2016 wurden insge-

samt mehr als 400 BEC Betrügereien pro Tag registriert, wobei vor allem klein- und mittelgroße Unternehmen davon betroffen waren. Schätzungen zu Folge betrug der Schaden von derartigen Straftaten in den letzten drei Jahren mehr als drei Milliarden Dollar, darunter mehr als 22.000 Opfer weltweit. Diese Masche wird oft auch im Zuge von Spear-Phishing und Whaling-Angriffen (siehe Kapitel 3.5) verwendet.

3.2 Vorgehensweise

Gupta et al. [GTJA16] identifizieren insgesamt fünf speziell für einen Phishing-Angriff typische Phasen: Planung und Setup, Phishing, Einbruch/Infiltration, Datensammlung und Ausbruch/Exfiltration. Wie diese einzelnen Phasen bei der klassischen Vorgehensweise durchlaufen werden und zusammenhängen, wird in Abbildung 3.3 veranschaulicht.

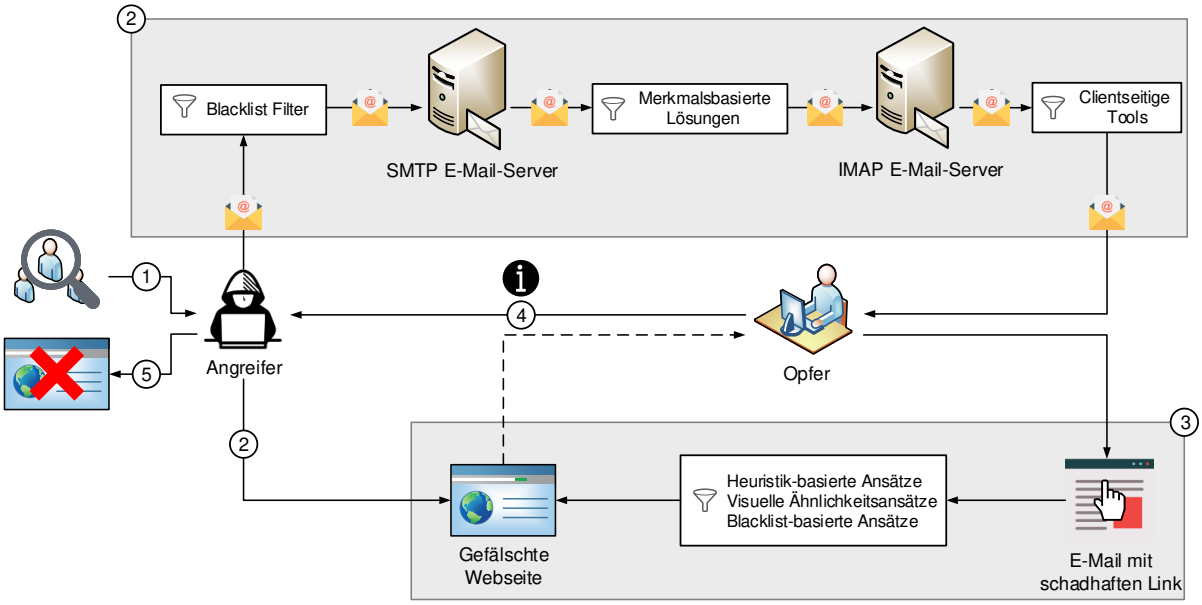

Abb. 3.3: Vorgehensweise bei einem klassischen Phishing-Angriff (vgl. [GTJA16][GuAP17])

1. **Planung und Setup:** Im ersten Schritt bestimmt der Angreifer zunächst sein Ziel. Dies kann entweder ein Unternehmen, eine Einzelperson, eine bestimmte Personengruppe, eine gesamte Nation oder die breite Masse sein. Anschließend versucht er möglichst viele Informationen (wie z.B. E-Mail-Adressen) über sein Opfer und die zugrundeliegende Infrastruktur herauszufinden. Dazu kann er beispielsweise den Netzwerkverkehr überwachen, persönlich das betroffene Unternehmen besuchen oder mit Spammer[1] zusammenarbeiten. Mit Hilfe dieser gewonnenen Details setzt er anschließend einen passenden Angriff auf. In den meisten Fällen umfasst dieser eine gefälschte E-Mail, welche entweder einen Link auf eine manipulierte bzw. gefälschte Webseite oder einen schadhaften Anhang beinhaltet [Ali15][KHHW15][GTJA16].

2. **Phishing:** In der nächsten Phase wird die gefälschte Webseite online verfügbar gemacht sowie die zuvor aufgesetzte E-Mail im Namen einer vermeintlich vertrauenswürdigen In-

[1] Unter Spammer versteht man organisierte Cyberkriminelle, deren Expertise in der Sammlung von aktiven E-Mail-Adressen liegt. Sie verfügen oft über Datenbanken mit Millionen von E-Mail-Adressen [Ali15].

stanz (wie z.B. einer Bank) an die Zielperson(en) gesendet. Im Zuge dessen wird unter der Angabe eines falschen Vorwandes zum Klicken auf einen schadhaften Link oder Anhang aufgefordert. Dabei passiert die E-Mail zunächst die DNS-basierte Blacklist des jeweiligen E-Mail-Systems. Sollte der Domainnamen des Senders dabei in der Blacklist erscheinen, wird die Mail noch bevor sie den SMTP-Server erreicht blockiert. Anderenfalls können sich auf dem Weg bis in das elektronische Postfach des Empfängers noch zahlreiche auf strukturelle Merkmale basierte Filter befinden. Die letzte Hürde bilden schlussendlich die clientseitigen Filter, welche jede E-Mail ebenfalls auf unterschiedliche Merkmale untersuchen und möglicherweise blockieren [Ali15] [KHHW15][GTJA16][GuAP17].

3. **Einbruch/Infiltration:** Sobald das Opfer auf den schadhaften Link klickt, kann auch die Anfrage für die entsprechende Webseite wiederum clientseitig durch diverse Applikationssicherheitslösungen blockiert werden. Hierzu zählen unter anderem Blacklist-basierte Ansätze, welche die Anfrage blockieren, falls die URL in einer entsprechenden Blacklist erscheint. Ebenso Ansätze, welche auf Heuristiken bzw. visuelle Ähnlichkeiten basieren und eine Anfrage blockieren, wenn die jeweilige Webseite typische Phishing-Merkmale aufweist. Wurde die Anfrage jedoch nicht blockiert, so wird letztlich entweder eine beliebige Art von Malware installiert, welche dem Angreifer das Eindringen in das System ermöglicht, oder eine gefälschte Webseite erscheint, welche eine Eingabe von sensiblen Daten (wie z.B. Anmeldedaten) fordert [GTJA16][GuAP17].

4. **Datensammlung:** Wenn der Angreifer erst einmal Zugang zum Zielsystem erlangt hat, besteht der nächste Schritt in der Extraktion aller erforderlichen Daten und Informationen. Im Falle, dass der Angreifer beliebige Anmeldedaten erfolgreich „gephisht“ hat, kann er nun auf das entsprechende Konto zugreifen und dem Opfer droht somit ein finanzieller Verlust. Sollte es sich hingegen um einen Malware-Angriff handeln, so kann der Angreifer das kompromittierte System in Folge auch für DDoS-Angriffe (Distributed-Denial-of-Service-Angriffe) in Betracht ziehen [GTJA16].

5. **Ausbruch/Exfiltration:** Der letzte Schritt besteht in der Beseitigung sämtlicher Spuren und dem Entfernen aller Hinweise bzw. Anhaltspunkte, welche auf einen Angriff oder den Angreifer selbst rückschließen lassen. Dazu zählt z.B. das Entfernen der gefälschten Webseite oder das Löschen des trügerischen E-Mail-Accounts. Abschließend reflektiert der Angreifer in dieser Phase noch den gesamten Verlauf des Angriffes, um so zukünftige Angriffe noch raffinierter gestalten zu können [GTJA16].

3.3 Angriffskanäle

Zur Durchführung eines Phishing-Angriffs stehen einem Social Engineer eine Reihe von verschiedenen Angriffskanälen zur Verfügung. Einige davon werden im Folgenden näher betrachtet.

3.3.1 E-Mail

E-Mail ist das mit Abstand am häufigsten bei einem Phishing-Angriff verwendete Übertragungsmedium. Mit Hilfe von zahlreichen Techniken und Tools kann der Angreifer hierbei Millionen von legitimen und aktiven E-Mail-Adressen mit seinen speziell für die jeweiligen Opfer angepassten E-Mails zugleich erreichen. Wie bereits erwähnt, arbeitet er im Zuge dessen häufig mit Spammern zusammen, um so an die Adresslisten für seine Phishing-Nachrichten zu gelangen.

Dabei machen die bereits bekannten Fehler bzw. Schwächen in den üblichen Mail-Server Kommunikationsprotokollen (wie SMTP) derartige Angriffe erst möglich. Dadurch sind Phisher relativ einfach in der Lage, durch die Manipulation des Nachrichtenheaders E-Mails mit einem gefälschten Absender an beliebige Empfängeradressen zu versenden. Aber nicht nur technische Aspekte, sondern vor allem auch die im Zuge dessen angewendeten Ansätze des Social Engineerings tragen einen erheblichen Teil zu deren Erfolg bei. Die heute bereits schon weit verbreitete Bekanntheit von Phishing-Angriffen hat zwar zu einer erhöhten Vorsicht der Benutzer beim Versenden von vertraulichen Informationen via E-Mail geführt, jedoch sind die Erfolgsaussichten generell immer noch relativ gut [Ollm07][Alse14].

Verwendete Techniken in Phishing-E-Mails

Damit die Zielperson(en) auch wirklich auf den Trug der Phishing-E-Mails hereinfallen, stehen dem Angreifer eine Reihe von verschiedenen Techniken zur Verfügung. Die heutzutage am weitest verbreiteten und gewöhnlichsten Ansätze werden nachfolgend näher betrachtet.

Generelle Design-Merkmale: Die Akzeptanz einer Phishing-E-Mail kann vor allem durch ein sorgfältiges Design und Erscheinungsbild erhöht werden. Der Titel wird nach Möglichkeit attraktiv gewählt, damit die Motivation des Benutzers zum Öffnen der Nachricht dementsprechend hoch ist. Der Inhalt wird ebenfalls mit starken und qualitativ hochwertigen Argumenten bzw. vermeintlichen Vorwänden versehen [Alse14]. Einige Beispiele dazu sind unter anderem (vgl. [Hong12][RaNa14][Alse14]):

- Zurücksetzen des Passwortes aufgrund von Datenkorruption, eines Serverabsturzes oder ähnlichen Sicherheitsproblemen
- Bestätigen der Anmeldedaten aufgrund von ungewöhnlichen Transaktionen, mehrfach fehlgeschlagenen Anmeldeversuchen oder anderen verdächtigen Vorgängen
- Installation eines Sicherheitsupdates aufgrund eines neu entdeckten Angriffes
- Befolgung von bestimmten Sicherheitsvorkehrungen aufgrund von Systemproblemen
- Teilnahme an einer Bankumfrage in Verbindung mit einem kleinen Geldbetrag als Gegenleistung bzw. Erkenntlichkeit, wofür die Angabe von Kontoinformationen erforderlich ist

Zusätzlich beinhalten diese Nachrichten oft noch Phrasen wie „um die Sicherheit ihres Benutzerkontos gewährleisten zu können“ oder „anderenfalls müssen wir Ihr Benutzerkonto sperren“, wodurch die Besorgnis des Opfers erregt und auf die Dringlichkeit des Anliegens hingewiesen werden soll. Um die Glaubwürdigkeit zu erhöhen, wird das Erscheinungsbild der Nachricht ebenfalls derart angepasst, dass es von einer legitimen Nachricht der jeweiligen Instanz nur sehr schwer oder gar nicht unterscheidbar ist. Hier wird vor allem auf eine korrekte Syntax und Semantik (richtige Satzstellung und Vermeidung von Tippfehlern) sowie eine angemessene Struktur (Stil des Inhaltes, authentische Firmenlogos und Copyright-Informationen) geachtet. Schlussendlich werden zur Erhöhung des Vertrauens und der Reduzierung aller wahrgenommenen potentiellen Risiken (wie z.B. der Verlust von Geld oder privaten Daten) noch Zeichen bzw. Statements hinzugefügt, welche die Privatsphäre oder die Sicherheit der Informationen sicherstellen sollen. Dazu zählen z.B. Icons von Drittanbietern zur Gewährleistung der Datensicherheit oder die Angabe von Datenschutzerklärungen [Ollm07][Alse14].

Ausnutzung der Schwächen des SMTP Protokolls: Durch die Ausnutzung der Schwächen im Simple Mail Transfer Protocol (SMTP) können ganz einfach illegitime E-Mails an beliebige Adressaten versendet werden. Diese weit verbreitete Methode, auch bekannt als „Mail-Spoofing“, ist aufgrund eines mangelnden Authentifizierungsmechanismus im Protokoll selbst erst möglich. Dazu muss sich der Angreifer nur über eine Telnet-Verbindung zum SMTP-Server verbinden und kann anschließend die verschiedenen Informationen des Nachrichtenheaders beliebig manipulieren bzw. fälschen. Üblicherweise wird im Zuge einer Phishing-E-Mail als Absenderadresse („Mail From:“) eine vermeintlich vertrauenswürdige Instanz vorgetäuscht [Ollm07][PaJK10]. Um so eine manipulierte Nachricht mittels SMTP versenden zu können, sind die in Tabelle 3.1 ersichtlichen Schritte auf einer Kommandozeile notwendig (vgl. [PaJK10][Klen08]):

Tab. 3.1: Beispiel des Versendens einer gefälschten E-Mail mittels SMTP

Kommando	Beschreibung
`telnet smtp.example.com 25`	Aufbau einer Verbindung zum SMTP-Server auf Port 25
`EHLO test.example.com`	Identifizierung des Clients für den Server
`MAIL FROM:<info@raiffeisen.at>`	Absenderadresse der E-Mail
`RCPT TO:<person@targetdomain.com>`	Empfängeradresse der E-Mail
`DATA`	Kennzeichnet den Beginn des Nachrichteninhaltes

`From: <info@raiffeisen.at>`
`To: person@targetdomain.com`
`Date: Thu, 4 May 2017 05:33:29 -0700`
`Subject: Bestätigung der Anmeldedaten`
 	
`Text der Nachricht...`
`.`
	Nachrichteninhalt, wobei zwischen dem Nachrichtenkopf und der eigentlichen Nachricht eine Leerzeile vorhanden sein muss. Das Ende der Nachricht wird mittels einer Zeile, welche nur einen Punkt enthält, gekennzeichnet.

`QUIT`	Verbindungsabbau

Verwendung von HTML-basierten E-Mails zur Verschleierung der Ziel-URL: Da HTML zu den interpretierten Sprachen zählt, kann eine Verschleierung der Ziel-URL auf verschiedene Arten erreicht werden (vgl. [Ollm07]):

- In dem die Textfarbe gleich gewählt wird wie die Hintergrundfarbe, können verdächtige bzw. Misstrauen erweckende Teile der URL verschleiert werden.
- Links bzw. Hyperlinks können in HTML mittels des `<a href="URL">` Tags definiert werden, wobei das `href`-Attribut der Angabe der Ziel-URL dient. Darauffolgend kann ein beliebiger String angegeben werden, welcher an Stelle der eigentlichen URL als Text für den Benutzer am Bildschirm erscheint und mit dem End-Tag `</a>` abschließt. Dadurch kann eine legitime URL als sichtbarer Text angegeben werden, hinter welcher sich tatsächlich jedoch ein Verweis auf eine schadhafte Phishing-URL verbirgt:

  ```
  <a href="http://www.myphishingsite.com">http://www.raiffeisen.at</a>
  ```

- Gleichermaßen kann eine tatsächliche URL auch durch das Hinzufügen einer Grafik, welche den Anschein einer Textnachricht bzw. legitimen URL erweckt, verborgen werden.

- Außerdem können diese E-Mails durch die zahlreichen verfügbaren Optionen und Möglichkeiten von HTML derart gestaltet werden, dass sie den Eindruck einer im Nur-Text-Format formatierten Nachricht hinterlassen und somit gar nicht als solche erkannt werden.

E-Mail-Anhänge: Häufig werden Phishing-E-Mails auch schadhafte Anhänge in Form von ausführbaren Dateien beigefügt. Typischerweise wird der Benutzer im Verlauf der Nachricht zum Öffnen der „vertrauenswürdigen“ Datei unter der Angabe eines falschen Vorwandes (wie z.B. zur Verifikation oder Überprüfung von Transaktionsdetails) aufgefordert. Sollte der Anhang geöffnet werden, wird wiederum eine beliebige Art von Malware am System installiert [Ollm07].

Umgehen der Spam-Erkennung: Viele Phishing-E-Mails werden an eine Vielzahl von verschiedenen Personen zugleich gesendet, wobei die dazu verwendeten Zieladressen oft aus mehreren unterschiedlichen Quellen stammen. Aufgrund dessen werden sie auch meist von den gängigen Spam-Filtern erkannt und der Benutzer erhält somit im Vorhinein bereits eine Warnung. Um dies zu verhindern, fügen Angreifer zusätzliche Phrasen, SMTP-Headers oder andere Referenzen zur Umgehung dieser Spamfilter den E-Mails hinzu. Eine Möglichkeit dazu ist das Miteinbeziehen von bedeutungslosen Sätzen am Ende der Nachricht, welche oft auch für das Opfer durch eine geschickte Farbwahl nicht erkennbar sind. Dadurch soll vor allem den auf diversen Heuristiken basierten Spamfiltern das Erkennen von derartigen Nachrichten erschwert werden. Eine weitere Methode, um Ähnliches zu erreichen, ist das Hinzufügen von absichtlichen Tippfehlern oder Leerzeichen in bestimmten Schlüsselwörtern, wie z.B. „Bitte Best-ätigen Sie Ihre An-melded-aten hier“. Oder es können aber auch diverse Header-Flags der Übertragungsprotokolle, welche möglicherweise auf eine derartige Nachricht hinweisen, manipuliert werden [Ollm07].

Ausnutzung von Schriftartunterschieden: Die im Zuge einer Phishing-E-Mail verwendete Schriftart ist meist ebenfalls nicht willkürlich gewählt, sondern kommt aufgrund von verschiedenen Merkmalen gezielt zum Einsatz. Dazu werden häufig Schriftarten gewählt, bei welchen das Erscheinungsbild von manchen Kleinbuchstaben gleich wie jenes von bestimmten Großbuchstaben (oder umgekehrt) ist. Dadurch erscheint ein Wort zwar als korrekt, jedoch beinhaltet es einen falschen Buchstaben und kann somit beispielsweise von einem Spam-Filter nicht als Schlüsselwort erkannt werden. Beispiele dazu sind unter anderem das Ersetzen des Großbuchstabens von „i“ mit dem Kleinbuchstaben von „L“ oder das Ersetzen der Zahl Null mit dem Großbuchstaben von „O“. Weiters wird häufig auch nur für ein bestimmtes Zeichen oder einen bestimmten Buchstaben eine andere Schriftart verwendet, damit die tatsächlich verwendete Sprache als eine andere Sprache erscheint (wie z.B. die Verwendung des kryllischen „o“ anstatt des lateinischen „o“, vgl. [Ollm07]).

Verwendung der jeweiligen Landessprache: In den Anfängen von Phishing-Angriffen waren diese meist nur auf englischsprachige Benutzer ausgelegt [MoTM15]. Um die Erfolgschancen jedoch zu erhöhen, wird mittlerweile auch besonders auf die Verwendung der jeweiligen lokalen Sprache der Zielgruppe geachtet. Sollte eine Nachricht nicht in der Landessprache der Zielperson sein, kann dies bereits zu ersten Zweifeln führen [Ollm07]. Auf der anderen Seite kann die verwendete Sprache ebenfalls die Fähigkeit der Zielpersonen, Täuschungshinweise (wie z.B. Tippfehler oder grammatikalische Fehler) zu erkennen,

stark beeinflussen. Ein durchgeführtes Experiment von Alseadoon [Alse14] zeigt dabei unter anderem, dass Personen, dessen Muttersprache nicht ident mit der im Phishing-E-Mail verwendeten Sprache ist, eine deutlich erhöhte Anfälligkeit aufweisen.

Verwendung von Kredikartennummern: Eine weitere beliebte Methode, um den in der E-Mail angegebenen Vorwand attraktiver und glaubwürdiger zu gestalten, ist die Angabe von Teilen der Kreditkartennummer des Opfers. Vielen Personen genügt diese Information als eine Art Bestätigung der Vorkenntnisse der ausgegebenen Instanz bezüglich ihrer eigenen Identität und ihrer Kreditkarteninformationen. Somit glauben sie schließlich, dass es sich um eine vertrauenswürdige Nachricht handeln muss, da ja nur ihr Finanzinstitut über diese Information verfügen kann. Meistens werden dazu jedoch nur die ersten vier Zahlen der Kreditkartennummer angezeigt und die restlichen mit Hilfe von Sternchen (*) verdeckt (z.B. 4985 **** **** ****). Viele Personen wissen wiederum jedoch nicht, dass die ersten vier Ziffern nicht einzigartig bzw. eindeutig sind, sondern nur die Kennung der jeweiligen Kreditkartengesellschaft darstellen und somit für einen Angreifer leicht ermittelbar sind [Ollm07].

Gefälschte Beiträge in Internetforen: Durch das Schreiben einer gefälschten E-Mail in ein beliebiges Internetforum oder eine beliebige E-Mail-Verteilerliste kann ein Angreifer anonym eine Vielzahl von Personen zugleich erreichen. Dadurch wird ihm ein mühseliges Schreiben von individuellen Nachrichten erspart und gleichzeitig kann er nur eine bestimmte Zielgruppe seiner Wahl ansprechen. Auch hier nutzt der Angreifer wiederum die zuvor beschriebenen Methoden und Techniken, um die eigentliche Intention der Nachricht bestmöglich zu verschleiern [Ollm07].

3.3.2 Webseiten

Neben E-Mails zählen auch Webseiten zu den am häufigsten verwendeten Methoden, um einen Phishing-Angriff durchzuführen [Ollm07]. Dazu versucht der Angreifer meist seine Opfer zum Besuch einer gefälschten Webseite zu animieren (beispielsweise mittels Phishing-E-Mails), um im Zuge dessen persönliche Informationen sammeln zu können [Hong12]. Die dazu verwendeten Seiten werden meist derart gefälscht bzw. gestaltet, dass sie einer legitimen Seite möglichst ähneln und das Opfer somit in den Glauben versetzt wird, dass es sich tatsächlich um die originale Webseite handelt (auch bekannt als „Web-Spoofing“) [BaBa13]. Diese Seiten werden entweder mit Hilfe von kostenlosen Webspaces, der Kompromittierung eines vorhandenen Web-Servers oder der Registrierung einer neuen Domäne gehostet [Hong12]. Die dazu verwendeten Methoden können dabei grob in zwei Bereiche unterteilt werden: einerseits Methoden, um die tatsächliche URL der gefälschten Webseite zu verschleiern und andererseits Methoden, welche bei der Manipulation bzw. Fälschung der Webseite selbst angewendet werden.

URL-Verschleierung

Der Schlüssel zu einem erfolgreichen Phishing-Angriff liegt oft in der Überzeugung einer anderen Person, einen Link (URL) zu folgen, damit sie auf die gewünschte (gefälschte) Webseite des Phishers gelangt. Um dies zu erreichen, stehen dem Angreifer eine Vielzahl von verschiedenen Methoden und Tricks bezüglich der Verschleierung der tatsächlichen URL zur Verfügung (auch „URL-Spoofing“ genannt). Die Bedeutendsten und Bekanntesten werden nachfolgend erläutert (vgl. [Ollm07][ZhLK09][Hong12][MoTM15][AaRa16]).

Verwendung von bösgläubigen Domainnamen: Wenn im Zuge eines Angriffes eine Domäne neu registriert wird, entspricht die Verwendung eines bösgläubigen Domainnamens der einfachsten Verschleierungsmethode. Hierbei wird der Name der nachgeahmten Domain möglichst ähnlich dem Namen der legitimen Domain gewählt. Beispielsweise kann die legitime Domain *raiffeisen.at* mit der dazugehörigen Transaktionswebseite *https://banking.raiffeisen.at* auf vielfältige Weise imitiert werden:

- *http://banking.raiffeisen.at**.ch***
- *http://raiffeisen.**banking**.at*
- *http://**login-banking**.raiffeisen.at*

Angreifer nutzen auch hier in vielen Fällen wiederum visuelle Ähnlichkeiten von verschiedenen Ziffern aus – auch bekannt als homographischer Angriff. Dabei wird entweder auf die Mehrdeutigkeit von einzelnen ASCII-Ziffern gesetzt oder die Möglichkeit der Verwendung von verschiedenen Zeichensätzen bzw. Alphabeten missbraucht (wie bereits beim Angriffskanal 3.3.1 E-Mail beschrieben).

Verwendung von gekürzten URLs: Aufgrund der Komplexität und zunehmenden Länge von URLs kommen heutzutage immer häufiger Kurz-URL-Dienste (engl. URL Shortener) zum Einsatz. Mit ihrer Hilfe wird für eine bereits existierende URL eine andere, alternative URL, die meistens weitaus kürzer ist, generiert und zur Verfügung gestellt. Angreifer lassen sich dazu oft für ihre Phishing-Webseite zunächst eine gekürzte URL kreieren. Anschließend verweisen sie in der Phishing-E-Mail absichtlich auf eine komplexe und inkorrekte, aber vermeintlich richtige URL und geben zusätzlich die gekürzte URL als Alternative an, falls es zu etwaigen Problemen mit der (inkorrekten und nicht funktionierenden) anderen URL kommt. Sehr beliebte Domains für gekürzte URLs sind unter anderem *bit.ly* und *bitly.com*.

Verwendung der IP-Adresse in der URL: Die meisten Personen sind nur mit eindeutigen Domainnamen in der URL vertraut und wissen gar nicht, dass auch die IP-Adresse an dieser Stelle verwendet werden kann. Dadurch kann ebenfalls die Zieldomain verschleiert oder ein Contentfilter umgangen werden. Beispielsweise kann anstatt der URL *http://www.raiffeisen.at/oesterreich/NA-NA-NA-30-NA.html* auch die URL *http://217.13.188.3/oesterreich/NA-NA-NA-30-NA.html* verwendet werden.

Verwendung von verschiedenen Kodierungen: Damit auch lokale Sprachen in Internetanwendungen (wie z.B. einem Webbrowser oder einem E-Mail-Client) unterstützt werden können, bieten die meisten Anwendungen unterschiedliche Datenkodierungen an. Im Zuge dessen kann ein Angreifer ebenfalls Gebrauch von verschiedenen Kodierungen machen, um so die tatsächliche Gestalt einer URL zu verschleiern. Z.B. wird ein Leerzeichen bei der Prozentkodierung als *%20* dargestellt, bei der Unicode-Kodierung jedoch als *%u0020*.

Verwendung von Fast-Flux-Netzwerken: Unter Fast-Flux (FF) versteht man grundsätzlich ein dezentralisiertes Botnetz (d.h. ein Netzwerk bestehend aus kompromittierten Computersystemen), welches ebenfalls zur Verschleierung des tatsächlichen Webhosts einer Webseite genutzt werden kann. Die einzelnen Hosts dieses Netzwerkes werden dabei als Proxys verwendet und ändern ständig dynamisch die IP-Adressen ihrer DNS-Einträge. Diese Technik basiert auf dem Prinzip des Round-Robin DNS, welches ursprünglich ei-

gentlich zur Lastverteilung für Netzwerkdienste eingeführt wurde. Dabei existieren für einen bestimmten Domainnamen mehrere DNS-Einträge. Sobald eine DNS-Anfrage für diesen Namen eintrifft, wird diese mit dem am längsten nicht verwendeten Eintrag beantwortet. Gleichermaßen werden die in einem FF-Netzwerk kompromittierten Hosts als Frontend-Proxys verwendet, so dass sich der Angreifer hinter einem Schleier aus sich permanent ändernden IP-Adressen verbirgt und dadurch eine Rückverfolgung erheblich erschwert wird.

Verwendung von Cloud-Diensten: Um eine gefälschte Webseite zu hosten, greifen Phisher auch immer häufiger auf kostenlose Cloud-Dienste, wie z.B. Google Drive oder Google Spreadsheet, zurück. Dadurch entspricht die URL der Seite in diesem Fall einer legitimen Google-Adresse, wobei die URL *google.com* vermutlich kein Benutzer so schnell blockieren wird.

Webseitenmanipulation

Zu Beginn des Aufkommens von Phishing-Angriffen wurden die dazu verwendeten Webseiten noch per Hand selbst von den Betrügern auf Basis des jeweiligen Originals nachgebaut. Dies führte in vielen Fällen zu einer äußerst schlechten Qualität, gekennzeichnet von zahlreichen Rechtschreibfehlern und Links zu Bildern der legitimen Webseite. Heutzutage wird jedoch ein Großteil aller Phishing-Webseiten automatisiert mit Hilfe von sogenannten Toolkits erstellt. Dazu muss der Angreifer lediglich die originale, zu kopierende Webseite sowie Informationen, wohin die gestohlenen Daten weitergeleitet werden sollen, angeben. In Folge generieren diese Toolkits automatisch eine idente Fälschung und befüllen diese mit dem Inhalt des Originals [Hong12]. Bei der Fälschung bzw. Manipulation einer vorhandenen Webseite kommen unter anderem die folgenden Methoden und Techniken zum Einsatz (vgl. [Ollm07][MoTM15][Shak12]):

Verwendung von Frames: Frames weisen eine hohe Beliebtheit bei Phishing-Angriffen auf, da durch ihre Verwendung bösartige Webseiteninhalte eingebettet werden können. Dazu können entweder `<frameset>` oder `<iframe>` Elemente verwendet werden. Bei der Verwendung der ersten Variante wird ein `<frameset>` Element bestehend aus zwei verschiedenen `<frame>` Elementen definiert. Das erste Element beinhaltet hierbei die legitime URL und nimmt 100 % des verfügbaren Anzeigebereiches in Anspruch, wobei das zweite hingegen die schadhafte URL beinhaltet und 0 % des verfügbaren Bereiches in Anspruch nimmt und somit verborgen bleibt (siehe Beispiel 3.1). Mit Hilfe der Webseite, welche im verborgenen Frame verlinkt wird, kann der Angreifer zusätzlichen Inhalt in die bestehende Seite einbinden, vertrauliche Informationen ergaunern oder einen beliebigen Schadcode ausführen [Ollm07]):

```
<frameset rows="100%,*" framespacing="0">
  <frame name="real" src="http://www.raiffeisen.at" scrolling="auto">
  <frame name="hiddenContent" src="http://www.myphishingsite.com" scrolling="auto">
</frameset>
```

Src. 3.1: Beispiel eines versteckten HTML-Frames

Bei der Verwendung der zweiten Variante wird ein `<iframe>` Element definiert, welches ebenfalls zur Einbindung von Inhalten einer anderen Webseite genutzt werden kann. Unter anderem setzen auch viele beliebte soziale Netzwerke (wie z.B. Facebook) auf iFrames, damit Benutzer dynamisch eigene Inhalte auf ihren Profilen einbinden können. Dies

wiederum ermöglicht es Phishern, beispielsweise mittels der Ausnutzung von Cross-Site-Scripting-Schwachstellen (siehe Kapitel 3.4.2) einen schadhaften Inhalt in die bestehende Webseite einzuschleusen. In Beispiel 3.2 wird zur Demonstration mittels iFrames zuerst der Inhalt einer legitimen Webseite im oberen Bereich eingebunden, gefolgt von einer Login-Möglichkeit im unteren Bereich, welche jedoch von einer Phishing-Webseite stammt.

```
<iframe src="http://www.testcompany/index.html" width="1450" height="300" frameborder="0"></
    iframe>
<iframe src="http://myphishingsite.com/wp-login" width="1450" height="250" frameborder="0"></
    iframe>
```

Src. 3.2: Beispiel der Verwendung von iFrames

Überschreiben von bestehenden Webseiteninhalten: Auch zur Überschreibung des angezeigten Inhaltes einer legitimen Webseite stehen dem Angreifer zahlreiche Tricks zur Verfügung. Eine sehr beliebte Methode darunter ist unter anderem die Erstellung eines virtuellen Containers mittels des HTML-Tags `<div>`. Diesen Container kann der Angreifer mit beliebigen Inhalten befüllen und durch die Angabe einer absoluten Positionierung (mittels des Attributes `style`) über dem legitimen Inhalt platzieren. Mit dieser Vorgehensweise kann sogar eine gesamte Webseite überschrieben werden. Beispiel 3.3 illustriert wie dies mit Hilfe von JavaScript und HTML erreicht werden kann.

```
var d = document;
d.write('<div id="fake" style="position:absolute; left:200; top:200; z-index:100">');
d.write('<img src="raiffeisen-logo.png" alt="Raiffeisenbank"></div>');
```

Src. 3.3: Beispiel der Überschreibung eines bestehenden Inhaltes

Weitere Methoden: Neben Frames und virtuellen Containern kann ein Angreifer zusätzlich noch auf die folgenden Raffinessen zurückgreifen:

- Die Einbettung von verschleierten HTML-Links (wie in Kapitel 3.3.1 bereits beschrieben) in bekannten Webseiten oder Internetforen.
- Die Verwendung von gefälschten Werbebannern, welche das Opfer auf die Phishing-Webseite weiterleiten.
- Die Verwendung von Pop-ups oder rahmenlosen Fenstern, um die tatsächliche Quelle einer Phishing-Nachricht zu verbergen.
- Die Einbettung eines schadhaften Inhaltes in die Webseite, welcher eine bekannte Schwachstelle des Browsers ausnutzt, um so schadhafte Anwendungen auf dem Zielsystem zu installieren (wie z.B. Key-Loggers oder Trojaner).
- Das Missbrauchen von konfigurierten Vertrauenseinstellungen im Browser, um im Zuge dessen auf zugelassene Skripts oder Speicherbereiche zuzugreifen.
- Die Ausnutzung von Cross-Site-Scripting Fehlern, um so die tatsächliche Quelle der Webseite zu verbergen (Details siehe Kapitel 3.4.2).
- Die Verwendung eines gefälschten Schloss-Symbols in der Adressleiste, um so eine vorhandene SSL/TLS-Verbindung vorzutäuschen.

3.3.3 Weitere Angriffskanäle

Zwar zählen E-Mail und Webseiten mit Abstand zu den am häufigsten verwendeten Kanälen eines Phishing-Angriffes, jedoch gewinnen die nachfolgenden Übertragungsmedien ebenfalls an Beliebtheit.

Telefon/VoIP: Die Durchführung eines Phishing-Angriffes mittels Telefon oder Voice over Internet Protocol (VoIP) wird auch als Vishing bezeichnet. Dabei handelt es sich um eine Abkürzung, welche sich aus den beiden Wörtern „Voice“ und „Phishing“ zusammensetzt. Wie auch bei anderen Phishing-Angriffen, versucht der Social Engineer auch hier seine Opfer zur Preisgabe von vertraulichen Informationen oder zur Ausführung einer bestimmten Handlung zu animieren [Ollm07][RaNa14]. Dazu ruft er entweder mit einer gefälschten oder unterdrückten Nummer die Zielperson an und gibt sich als Techniker, Mitarbeiter oder als eine andere autoritäre Person aus und versucht somit möglichst viele Informationen zu erlangen. Häufig kommen dabei auch Stimmenverzerrer zum Einsatz, welche dem Angreifer zusätzlich das Verbergen seiner eigenen Identität erleichtern [SEF]. Einige Vishing-Angriffe werden hingegen auch automatisiert mittels Tonbänder und automatischen Dialogsystemen durchgeführt [RaNa14].

Vishing weist eine viel höhere Erfolgsrate als andere Phishing-Methoden auf und verursacht jährlich Schäden im Milliardenbereich. Dies kann unter anderem auf mehrere Gründe zurückgeführt werden. Zunächst kann mit Hilfe eines Telefonanrufes im Gegensatz zu E-Mail ein viel größerer Teil der Bevölkerung erreicht werden (vor allem auch ältere Personen). Weiters ist der Bevölkerung heutzutage aufgrund der weiten Verbreitung von Callcentern der Anruf eines Unbekannten, welcher nach verschiedenen Informationen fragt, bereits geläufig. Zudem sind automatische Dialogsysteme ebenso keine Seltenheit mehr und finden in der Bevölkerung mittlerweile ebenfalls eine hohe Akzeptanz. Außerdem kann eine Social Engineering Nachricht mittels eines Telefonanrufes viel persönlicher und authentischer vermittelt werden und der Zeitpunkt der Übermittlung kann ebenfalls zu Gunsten des Angreifers optimal gewählt werden ([Ollm07][SEF]):

SMS: Verwendet ein Phishing-Angriff als Übertragungsmedium eine SMS-Nachricht, so ist dies auch unter dem Kunstwort Smishing (Zusammensetzung aus „SMS“ und „Phishing“) bekannt. Darunter versteht man den Versuch, eine Person mit Hilfe einer SMS-Nachricht zum Öffnen einer schadhaften URL anzuregen. Auch hier gibt sich der Absender als vermeintlich vertrauenswürdige Instanz aus und sobald das Opfer auf den jeweiligen Link klickt, wird in der Regel eine schadhafte Anwendung installiert und der Angreifer erlangt somit Kontrolle über das Smartphone. Die im Zuge dessen versendeten Nachrichten werden immer raffinierter gestaltet und belaufen sich beispielsweise auf Rabattgutscheine, Hochzeitseinladungen oder Informationen zu diversen Veranstaltungen [JMSP17]. Smishing-Angriffe treten heutzutage immer häufiger auf, da unter anderem die Verwendung von Smartphones weit verbreitet ist und Benutzer mittlerweile mobilen Anwendungen vermehrt ein umfassendes Vertrauen in Bezug auf die Zahlung von Rechnungen oder anderen Geschäftstransaktionen schenken [SEF].

Instant Messengers und diverse Chats: Aufgrund des steigenden Trends von Instant Messengers und diversen anderen Chats (wie z.B. Whatsapp, Facebook Messenger, Skype, usw.), bilden diese auch immer häufiger die Basis von Phishing-Angriffen. Die Funktio-

nalität in diesen Anwendungen wird zunehmend umfangreicher und das Versenden von dynamischen Inhalten, wie z.B. Bilder, Videos, URLs, usw., wird mittlerweile von allen gängigen Messengers unterstützt. Dadurch lassen sich viele Methoden und Techniken des klassischen web-basierten Phishings auch hier anwenden [Ollm07].

3.4 Angriffsvektoren

Damit ein Phishing-Angriff auch erfolgreich ist, erfordert er die Anwendung von mehreren verschiedenen Methoden und Techniken. Dies wiederum resultiert in einer Menge an unterschiedlichen Wegen bzw. Angriffsvektoren, welche einem Social Engineer zur Verfügung stehen [Ollm07]. Die wichtigsten und am häufigsten vorkommenden Angriffsvektoren werden nachfolgend näher betrachtet.

3.4.1 Man-in-the-Middle-Angriff

Ein Man-in-the-Middle-Angriff zählt zu den erfolgreichsten Vektoren, um Zugriff auf Benutzerinformationen und -ressourcen zu erlangen [Ollm07]. Dabei schleust sich der Angreifer in die Verbindung zwischen Benutzer und der legitimen Webseite oder des legitimen Systems ein und agiert wie ein Proxy. Dadurch ist er in der Lage, unbemerkt den gesamten Datenverkehr abzufangen, aufzuzeichnen und gegebenenfalls auch manipuliert weiter zu senden [Ollm07][GaJa12][BaBa13]. Unabhängig ob eine HTTP- oder HTTPS-Verbindung zugrunde liegt, verbindet sich der Benutzer hierbei dem Anschein nach, als wäre er mit dem legitimen Server verbunden, zum Angreifer. Der Angreifer baut wiederum gleichzeitig eine Verbindung zum legitimen Server auf und vermittelt somit die gesamte Kommunikation zwischen den beiden Parteien weiter – typischerweise in Echtzeit. Im Falle einer HTTPS-Verbindung baut der Benutzer eine SSL/TLS-Verbindung zum Angreifer auf und dieser wiederum eine eigene SSL/TLS-Verbindung zum legitimen Server. Somit ist er auch in diesem Szenario in der Lage, den gesamten Datenverkehr auf beiden Seiten zu entschlüsseln, einzusehen und verschlüsselt wieder weiterzusenden (siehe Abbildung 3.4) [Ollm07].

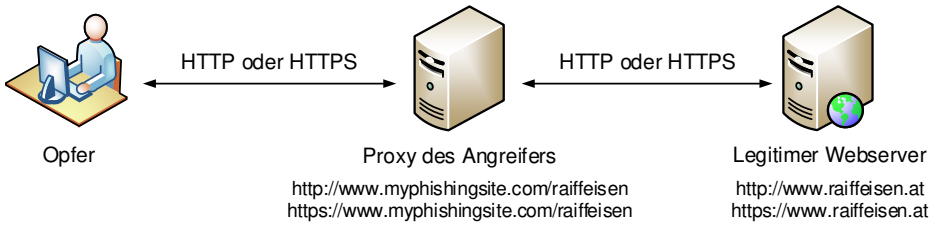

Abb. 3.4: Man-in-the-Middle-Angriff (vgl. [Ollm07])

Um den Benutzer, anstatt zum legitimen Webserver, zum jeweiligen Proxy-Server umzuleiten, kann der Angreifer auf mehrere Optionen zurückgreifen. Besteht für ihn die Möglichkeit, sich im selben Netzwerk oder direkt auf dem Weg zum legitimen Webserver einzuschleusen, kann er einen transparenten Proxy (Proxy, über welchen sämtlicher HTTP- und HTTPS-Verkehr völlig transparent läuft) einrichten. Weiters kann er mittels DNS Cache Poisoning den Netzwerkverkehr durch die Manipulation der DNS-Caches von verschiedenen Netzwerkkomponenten ebenfalls umleiten. Zuletzt kann die Verwendung von URL-Verschleierungsmethoden (siehe Ka-

pitel 3.3.2) oder die Manipulation der Proxy-Einstellungen im Browser des Benutzers ebenfalls zum gewünschten Resultat führen [Ollm07].

3.4.2 Cross-Site-Scripting-Angriff

Cross-Site-Scripting-Angriffe (XSS-Angriffe) zielen auf Schwachstellen in Webanwendungen ab, welche auf fehlende bzw. mangelhaft implementierte Sicherheitspraktiken während des Entwicklungsprozesses zurückgeführt werden können [Ollm07][ShKh14]. Sie sind bei dynamischen Webseiten ohne einer angemessenen Validierung von Benutzereingaben möglich und erlauben dem Angreifer im Zuge dessen schadhaften Code (JavaScript, VBScript, PHP, etc.) in die generierte Webseite einzuschleusen [ShKh14][GuAP17]. Die vom Angreifer verfolgten Ziele können hierbei angefangen von der Erlangung von Zugangsberechtigungen zu sensiblen Seiteninhalten, Webseiten-Cookies und zahlreichen anderen im Browser gespeicherten Informationen, über das Mitlesen von sämtlichen Benutzereingaben, bis hin zur Weiterleitung des Benutzers auf eine andere (schadhafte) Webseite reichen [Mill05][ShKh14][GuAP17]. Abbildung 3.5 stellt den typischen Verlauf eines XSS-Angriffes exemplarisch dar.

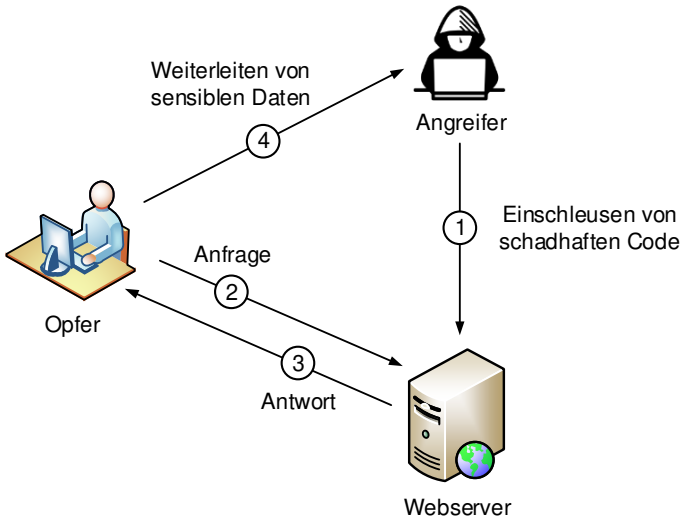

Abb. 3.5: Cross-Site-Scripting-Angriff (vgl. [ShKh14])

Angenommen bei der in Abbildung 3.5 angeforderten Webseite handelt es sich um ein beliebtes Internetforum, in welches Benutzer Fragen zu bestimmten Themengebieten posten können. Dabei wird jeder Eintrag in einer Datenbank gespeichert und anschließend allen Besuchern angezeigt. Sollte die Webseite hierbei über keine oder nur eine mangelhafte Eingabevalidierung der Daten (in diesem Fall der Frage des Benutzers) verfügen, ist sie anfällig für einen XSS-Angriff. Dazu muss der Angreifer lediglich einen eigenen Eintrag erstellen, wobei er JavaScript-Code umgeben von `<script>` Tags in die Eingabedaten miteinbezieht. Wird die Webseite mit diesem Eintrag in Folge von einem Benutzer angefragt, wird jedes Mal komplett transparent der injizierte JavaScript-Code ausgeführt und je nach seiner Intention der vom Angreifer angepeilte Schaden (wie z.B. das Weiterleiten von sensiblen Daten) verrichtet (vgl. [ShKh14]).

3.4.3 Session Hijacking

Da sowohl HTTP als auch HTTPS zustandslose Protokolle sind und somit mehrere zusammenhängende Anfragen von einem Benutzer nicht als solche erkennen und zuordnen können, müssen dazu dementsprechende Methoden bei der Entwicklung der Webanwendung berücksichtigt werden. Für gewöhnlich wird zu diesem Zwecke nach der Anmeldung eine Sitzung (engl. Session) für jeden Benutzer erstellt [Ollm07]. Diese ist notwendig, um den zu einer Anfrage dazugehörigen Benutzer nachverfolgen, dessen Berechtigungen für den Zugriff auf verschiedene Bereiche der Webseite überprüfen sowie den aktuellen Zustand der Session feststellen zu können. Dazu bekommt der Client vom Server im Zuge der Erstellung der Session eine Session-ID zugeteilt [BuVE13]. Diese muss fortan Teil jeder nachfolgenden Anfrage dieses Clients sein, entweder in dem sie direkt in der URL, mit Hilfe von Cookies oder in versteckten Formularfeldern an den Server übermittelt wird [Ollm07].

Diese Session-ID kann jedoch von einer anderen Person gestohlen und verwendet werden, was unter dem Begriff Session Hijacking bekannt ist. Ein Angreifer kann dies auf unterschiedliche Art und Weise erreichen. Eine Möglichkeit dazu besteht in der Durchführung eines XSS-Angriffes, bei welchem der schadhaft injizierte Code das Cookie mit der darin enthaltenen Session-ID des Benutzers zum Angreifer weiterleitet. Mit dieser gestohlenen Session-ID hat er in Folge ohne jegliche Authentifizierung die gleichen Berechtigungen, wie der legitime Benutzer auch und kann sich somit als dieser ausgeben [BuVE13]. Bei einer weiteren Variante verwendet der Angreifer eine vordefinierte Session-ID und wartet anschließend bis sich ein legitimer Benutzer unter dieser authentifiziert [Ollm07]. Der Ablauf dieses Angriffes ist in Abbildung 3.6 illustriert.

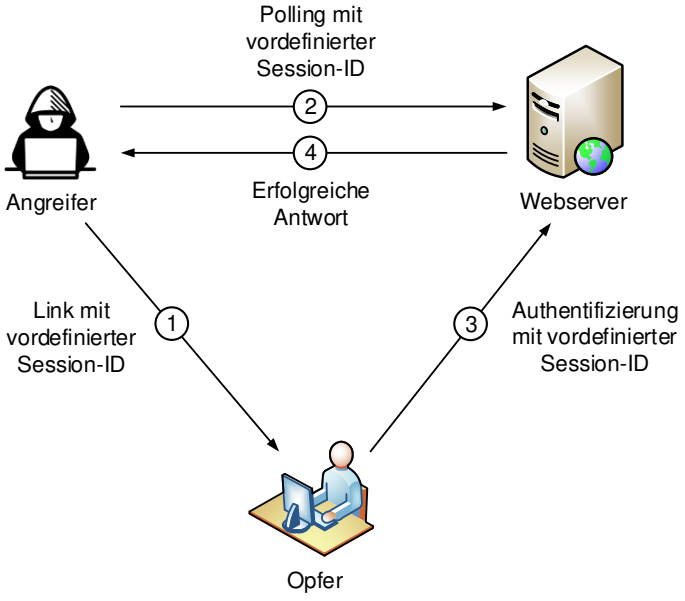

Abb. 3.6: Session-Hijacking-Angriff (vgl. [Ollm07][GTJA16])

Hierbei sendet der Angreifer eine Phishing-Nachricht mit einem Link zu einer legitimen Webseite, welcher jedoch eine vordefinierte Session-ID enthält. Daraufhin pollt er unter der Verwendung dieser vordefinierten Session-ID ständig den jeweiligen Server mit einer Anfrage für eine eingeschränkte Seite (wie z.B. eine E-Banking-Seite einer Bank). Solange sich kein legitimer Benutzer

unter dieser Session-ID anmeldet, bekommt der Angreifer Fehlermeldungen als Antwort zurück. Sobald ein Benutzer jedoch dem Link in der Phishing-Nachricht folgt und sich anschließend authentifiziert, lässt der Server alle Verbindungen unter der Verwendung dieser Session-ID zu und der Angreifer bekommt ebenfalls die gleichen Zugriffsberechtigungen wie der legitime Benutzer [Ollm07].

3.4.4 Spy-Phishing

Spyware ist eine Art von Malware und fasst alle Anwendungen zusammen, welche heimlich Informationen über eine Person oder ein Unternehmen ohne dessen Wissen und Zustimmung sammeln. Der Einsatz solcher Softwares ist vielfältig und reicht angefangen von Regierungsüberwachung über Unternehmensüberwachung bis hin zu kriminellen Verwendungszwecken. Unter letzterem Anwendungsbereich fällt unter anderem auch die Verwendung dieser Programme im Zuge eines Phishing-Angriffes, in der Literatur auch als Spy-Phishing bekannt [LiVi05][Yane06]. Bei einem derartigen Angriff setzt der Angreifer eine Phishing-E-Mail mit entweder einem Trojaner im Anhang oder einem Link zum Download des Trojaners auf. Sobald dieser heruntergeladen und ausgeführt wird, wird im Hintergrund die Spyware installiert. Fortan zeichnet diese alle gewünschten Informationen auf und leitet sie an den Angreifer weiter [Yane06]. Dabei können unterschiedliche Arten von Spyware verwendet werden, am häufigsten kommen jedoch Key- und Screenloggers zum Einsatz.

Keyloggers

Keyloggers sind eine Art von Spyware, welche die Tastatureingabe des Benutzers im Hintergrund aufzeichnen. Dadurch können sie eine Vielzahl an vertraulichen Informationen, wie z.B. Passwörter, Kreditkartennummern, private E-Mail-Korrespondenzen, Adressen, etc., erfassen. Diese Informationen werden entweder lokal auf dem Benutzergerät in Log-Files gespeichert und zu einem späteren Zeitpunkt vom Angreifer ausgelesen oder werden unverzüglich an ihn weitergeleitet [LiVi05]. Sie können einerseits als vorkompilierte Objekte lokal auf dem Gerät laufen und somit anwendungs- und kontextunabhängig alle Tastenanschläge aufzeichnen oder sie sind andererseits in clientseitigem Scripting-Code geschrieben und erfassen nur sämtliche Tastenanschläge im Kontext des Webbrowsers [Ollm07]. Der Funktionsumfang derartiger Programme geht jedoch meist weit über die reine Aufzeichnung von Tastenanschlägen hinaus und kann unter anderem auch die Protokollierung von Webseitenzugriffen, Programmausführungen, Aktivitäten der Zwischenablage und vieles mehr umfassen [LiVi05].

Screenloggers

Ähnlich wie Keyloggers zeichnen Screenloggers nicht die Tastatureingabe auf, sondern sämtliche Bildschirm- und Mausbewegungen. Sie werden vor allem bei sicheren Webanwendungen, welche über Schutzmechanismen gegen Keylogger-Angriffe verfügen, eingesetzt und sind im Gegensatz zu Keyloggern auch in der Lage, Eingabedaten über virtuelle Tastaturen zu erfassen. Da sie ausnahmslos alle Bildschirmaktivitäten aufzeichnen, können sie dem Angreifer zur Erlangung von wichtigen Eingabedaten des Benutzers auf verschiedenen Webseiten verhelfen [ASKV+12]. Zur Gewährleistung einer schnellen Übertragung der in diesem Zuge erfassten Daten mit möglichst geringem Volumen, zeichnen viele Screenloggers nur den relevanten Bereich der Webseite (z.B. nur den Bereich des Anmeldeformulars) und nicht den gesamten Bildschirm in Form von Screenhots auf [Ollm07].

3.5 Weitere Ausprägungen

Neben den bereits erwähnten Formen von Phishing (klassisches Phishing, Vishing, Smishing, usw.) gibt es unter anderem auch noch drei weitere Ausprägungen, welche nachfolgend kurz beschrieben werden.

Spear-Phishing: Spear-Phishing unterscheidet sich nicht großartig vom klassischen Phishing und verfolgt die gleiche Intention [RaNa14]. Anstatt jedoch eine große Anzahl an Nachrichten mit zufälligen Adressaten zugleich zu versenden, weist eine Spear-Phishing-Nachricht eine weitaus genauere Zielausrichtung auf und richtet sich lediglich an bestimmte Personen oder spezifisch ausgewählte Zielgruppen (wie z.B. sämtliche Personen eines bestimmten Unternehmens) [SEF][GTJA16]. Um die Nachricht möglichst glaubwürdig und personalisiert gestalten zu können, sammelt der Angreifer zuvor möglichst viele Informationen über seine Zielperson(en) [KHHW15][GuAP17]. Anschließend fertigt er eine passende Nachricht unter der Angabe eines falschen Vorwandes (wie z.B. Aktualisierung von kritischen Personendaten oder Zurücksetzen von kritischen Benutzerkennungen und Passwörtern) sowie der Aufforderung zum Klicken auf einen bestimmten Link oder zum Herunterladen eines schadhaften Anhanges an [RaNa14][Ali15]. Als Absender wird meist eine unternehmensinterne, höherrangige und zuverlässige Respektsperson angegeben [RaNa14]. Dadurch wird die Erfolgswahrscheinlichkeit des Angriffes enorm erhöht, da Mitarbeiter eine Nachricht eher öffnen und der darin gestellten Aufforderung nachkommen, wenn diese von einer scheinbar bekannten bzw. legitimen Person stammt [RaNa14][GTJA16]. Aufgrund dessen sind Spear-Phishing-Angriffe nur sehr schwer erkennbar und weisen eine viel höhere Trefferquote als klassische Phishing-Angriffe auf [Ali15][GTJA16].

Whaling: Ein Whaling-Angriff ist ähnlich wie ein Spear-Phishing-Angriff ebenfalls sehr gezielt und fokussiert ausgerichtet. Die Zielpersonen sind hierbei jedoch Führungskräfte innerhalb eines Unternehmens oder andere hochkarätige Individuen (wie z.B. hochrangige Beamte oder Regierungsmitglieder) mit Zugang zu kritischen Informationen und Ressourcen. Sollten ihre Anmeldeinformationen oder Zugangsberechtigungen kompromittiert werden, kann dies zu einer Gefährdung und Schädigung ihres gesamten Geschäftes führen. Bei einem derartigen Angriff ist die Vorbereitung und Recherche des Angreifers besonders wichtig. Er muss möglichst viel über das Opfer und dessen Umfeld wissen, um so eine zielgenaue und fein abgestimmte Phishing-Nachricht gestalten zu können [Ollm07][SEF]. Dabei sind die entsprechenden Nachrichten nicht unbedingt auf E-Mails beschränkt, sondern können zusammen mit infizierten Speichermedien auch auf dem normalen Postweg versendet werden [Ollm07].

Clone-Phishing: Die Grundlage eines Clone-Phishing-Angriffes bildet eine bereits versendete, legitime E-Mail, welche entweder einen Link oder einen Anhang beinhaltet. Der Angreifer kopiert den gesamten Inhalt samt Empfängeradressen und tauscht die darin enthaltenen Links und Anhänge gegen manipulierte bzw. gefälschte Versionen davon aus [Ma13][ChCR16]. Damit die Nachricht auch den Anschein erweckt, als käme sie vom ursprünglichen Absender, fälscht er zusätzlich noch dessen Adresse und sendet sie anschließend unter der Behauptung, es handle sich um eine aktualisierte Version des Originales, erneut an den/die jeweiligen Adressaten [ShSa12]. Bei den meisten populären Webmail-Diensten (wie z.B. Gmail, Hotmail oder Yahoo) kann der Benutzer ebenfalls eine Reply-To-

Adresse angeben. Dadurch kann sich die Absenderadresse einer E-Mail von jener Adresse, an welche die Antworten darauf gesendet werden, unterscheiden. Diese Funktionalität kann sich auch ein Angreifer im Zuge eines Clone-Phishing-Angriffes zunutze machen, in dem er seine E-Mail-Adresse im Reply-To-Feld angibt. Infolgedessen wird eine Antwort nicht an den legitimen Benutzer, sondern an den Phisher gesendet und eine mögliche Aufdeckung des Angriffes wird zusätzlich erschwert [Ma13].

3.6 Gegenmaßnahmen

Es gibt zahlreiche verschiedene Gegenmaßnahmen, Lösungen und Empfehlungen, um derartigen Angriffen entgegenzuwirken und vorzubeugen. Während einige davon versuchen Phishing direkt zu bekämpfen, fokussieren sich andere wiederum auf allgemeine Bedrohungen, welche die Basis dafür bilden können (wie z.B. Malware) [Mill05]. Da ein Phishing-Angriff Aspekte des Social Engineerings mit Aspekten der Technik kombiniert, können die vorhandenen Gegenmaßnahmen dementsprechend in technische und nichttechnische Maßnahmen unterteilt werden [Ali15].

3.6.1 Technische Gegenmaßnahmen

Wie in Kapitel 3.3 sowie 3.4 ersichtlich, ist das Arsenal an verschiedenen Methoden und Techniken eines Phishers relativ groß. Demzufolge gibt es zu deren Bekämpfung aus technischer Sicht keine All-in-One-Lösung, sondern es ist die Umsetzung einer Kombination von mehreren verschiedenen Informationssicherheitstechnologien und -techniken erforderlich. Zur Gewährleistung eines umfassenden Schutzes müssen diese Maßnahmen auf drei verschiedenen Ebenen implementiert werden: auf der Benutzerebene, Serverebene und Enterprise-Ebene (vgl. [LiVi05][Ollm07][KaSe14][Ali15]).

Benutzerebene

Sämtliche Maßnahmen der Benutzerebene streben einen Schutz des Benutzergerätes an. Aufgrund der stark variierenden Benutzerkenntnisse und des damit verbundenen Sicherheitsbewusstseins, ist die Sicherheit auf Heimrechnern im Gegensatz zu jener auf Firmenrechnern typischerweise äußerst dürftig. Dennoch kann durch die Umsetzung der nachfolgenden Gegenmaßnahmen in beiden Fällen ein wirksamer Schutz gegen Phishing-Angriffe erreicht werden [Ollm07].

Desktop-Schutzsoftware: Zwar sind die meisten Benutzergeräte mittels einer gängigen Anti-Viren-Schutzsoftware ausgestattet, jedoch sollten idealerweise mehrere unterschiedliche Schutzagenten gleichzeitig zum Einsatz kommen. Unabhängig ob dadurch etwaige unternehmenseigene Schutzdienstleistungen dupliziert werden, sollten zumindest lokale Services zur Gewährleistung des Vorhandenseins eines Anti-Viren-Schutzes, eines Anti-Spam-Schutzes, einer Desktop-Firewall, eines Angriffserkennungssystems (engl. Intrusion Detection System – IDS) sowie einer Spyware-Erkennung verwendet werden. In Anbetracht eines Phishing-Angriffes sollten diese zusammen die folgenden Funktionalitäten umfassen (vgl. [Ollm07]):

- Einen auf Basis eines E-Mail-Anhanges, Downloads oder dynamischen HTML- oder Skript-Inhaltes möglichen Versuch der Installation von schadhafter Software zur Laufzeit erkennen und blockieren

- Gängige Spam-Übertragungstechniken identifizieren und entsprechende Nachrichten unter Quarantäne stellen
- Die neuesten Viren- und Spamsignaturen periodisch herunterladen und die Schutzsoftware dementsprechend aktualisieren
- Nicht autorisierte Verbindungen ausgehend von installierten Programmen oder aktiven Prozessen erkennen und blockieren
- Anomalien im Netzwerkverkehr erkennen und entsprechende Gegenmaßnahmen einleiten
- Eingehende Verbindungen auf gesperrten Netzwerkports blockieren
- Eine Übermittlung von sensiblen Daten an eine misstrauische Partei unterbinden

E-Mail-Einstellungen: Die heutzutage gängigen E-Mail-Anwendungen verfügen über einen umfassenden Funktionalitätsumfang und ermöglichen somit einen technisch ausgefeilten elektronischen Schriftverkehr. Während einige dieser Features für unternehmensinterne Anwendungen und Systeme notwendig sind, kann jedoch auf die meisten im täglichen Gebrauch auch verzichtet werden. Viele davon bilden nämlich die Basis für Phishing-Angriffe und sollten daher standardmäßig deaktiviert werden [Ollm07].

Die in diesem Zusammenhang wohl gefährlichste Funktionalität stellt die Erlaubnis von HTML-basierten E-Mails dar. Zwar können dadurch im Gegensatz zu Klartext-E-Mails Nachrichten mit Hilfe von eingebetteten Grafiken, Links, Animationen und Tonklängen bzw. Musik visuell anspruchsvoller gestaltet werden, jedoch stellen sie ein nicht zu unterschätzendes Sicherheitsrisiko dar (siehe Kapitel 3.3.1). Aus diesem Grund sollte diese Funktionalität in sämtlichen E-Mail-Korrespondenzen entweder eingeschränkt oder komplett darauf verzichtet und stattdessen die Option von Klartext-E-Mails bevorzugt werden. Dadurch werden sämtliche Inhalte in Klartext angezeigt und somit die gängigsten Angriffsvektoren vermieden [LiVi05][Ollm07].

Des Weiteren bieten viele populäre E-Mail-Anwendungen auch die Blockierung von gefährlichen Anhängen an. Hierbei wird eine interne Liste mit sämtlichen als schädlich eingestuften Formaten verwaltet und der Benutzer am Öffnen dieser gehindert. Idealerweise sollten E-Mail-Anhänge (unabhängig vom Format) nie direkt in der Anwendung geöffnet werden können, sondern vom Benutzer zunächst immer lokal gespeichert werden müssen. Dadurch kann einerseits das Rendering der jeweiligen Anwendung nicht beeinflusst und andererseits die Inspektion von lokalen Anti-Viren-Programmen forciert werden. Wenn diese Einstellungsmöglichkeiten in der Anwendung vorhanden sind, sollten sie zur Vermeidung des Öffnen eines schadhaften Anhanges auch unbedingt genutzt werden [Ollm07].

Möglichkeiten des Browsers: Ähnlich wie E-Mail-Anwendungen, bieten auch Browser zahlreiche erweiterte Funktionalitäten an, welche in Folge bei einem Angriff ausgenutzt werden können. Die Palette hierzu wächst laufend enorm, jedoch kommt das Thema Sicherheit in vielen dieser Erweiterungen zu kurz und infolgedessen beinhalten sie oftmals unbeabsichtigte Sicherheitslücken. Unternehmen und auch Privatnutzer sollten einen für den jeweiligen Gebrauch entsprechenden Browser verwenden. Wenn beispielsweise dieser nur für das Surfen im Internet benötigt wird, ist ein vielseitiger und ausgefeilter Browser nicht

notwendig. Um generell Phishing-Angriffen vorzubeugen, sollten außerdem die folgenden Empfehlungen berücksichtigt werden (vgl. [Ollm07]):

- Deaktivierung von Pop-up Fenstern
- Deaktivierung der Java Runtime Unterstützung
- Deaktivierung der ActiveX Unterstützung
- Deaktivierung von sämtlichen Multimedia- und Autoplay-Erweiterungen
- Blockierung von unsicheren Cookies
- Deaktivierung von automatischen Downloads durch den Browser
- Speicherung von Downloads in einem Verzeichnis, welches von einem Anti-Viren-Programm gescannt wird

Des Weiteren gibt es mittlerweile bereits eine Vielzahl von verschiedenen Browser-Plugins, welche den Benutzer bei der Identifizierung einer Phishing-Webseite unterstützen. Die meisten davon werden in Form von Toolbars dem Browser hinzugefügt und überwachen bzw. untersuchen zur Laufzeit alle besuchten Webseiten [Mill05][Ollm07]. Basierend auf verschiedenen Charakteristiken klassifizieren Gupta et al. [GuAP17] die vorhandenen Lösungen bzw. Ansätze in vier verschiedene Kategorien: Blacklist- und Whitelist-basierte, Heuristik-basierte, visuelle Ähnlichkeiten-basierte und sonstige Lösungen.

Blacklist- und Whitelist-basierte Lösungen versuchen mit Hilfe von vordefinierten Listen eine Phishing-Webseite zu erkennen [MoTM15]. Eine Blacklist beinhaltet hierzu eine Reihe von bereits bekannten oder verdächtigen Phishing-URLs und -IP-Adressen, welche ständig aktualisiert und erweitert wird, wohingegen eine Whitelist alle zugelassenen bzw. als vertrauenswürdig eingestuften Domänen enthält [GTJA16][GuAP17]. Eine Blacklist kann entweder lokal oder auf einem Server abgelegt und mit Hilfe von verschiedenen Methoden befüllt werden. Beliebte Hilfsmittel hierzu sind unter anderem Webcrawlers, manuelles Voting oder Honeypots. Wann auch immer ein Benutzer eine Webseite besucht, wird die dazugehörige URL mit dieser Blacklist abgeglichen und der Benutzer bei Bedarf gewarnt. Hierbei erweisen sich vor allem die Quantität, die Qualität und das Timing als die wichtigsten Aspekte. Quantität bezieht sich auf die Anzahl der vorhandenen Phishing-URLs in der Liste, Qualität auf die Anzahl der falsch klassifizierten (legitimen) Webseiten (Falsch-Positiv-Rate) und Timing auf die zeitliche Lücke zwischen der Inbetriebnahme einer neuen Phishing-Webseite und ihrer Aufnahme in eine Blacklist [MoTM15]. Vor allem aufgrund Letzterem können mit diesem Ansatz jedoch bei Weitem nicht alle Angriffe erkannt werden: Laut durchgeführten Studien landen in etwa 47-83 % aller Phishing-URLs erst nach 12 Stunden in einer Blacklist. Angesichts der Tatsache, dass 63 % aller Angriffe bereits nach zwei Stunden wieder beendet sind, entspricht dies einer äußerst signifikanten Verzögerung [MoTM15][GTJA16].

Heuristik-basierte Lösungen bewerten eine Webseite anhand der Extraktion von bestimmen Features und Merkmalen (z.B. des HTML-Codes oder der URL) [MoTM15]. Basis hierfür sind gesammelte Daten und Erfahrungen von vergangenen (und aufgedeckten) Phishing-Angriffen [GuAP17]. Im Gegensatz zu Blacklist-basierten Methoden kann eine Lösung dieser Art eine neue, noch unbekannte Phishing-Webseite auch in Echtzeit erken-

nen. Die Effektivität hängt von der Auswahl der Menge an charakteristischen Merkmalen ab, welche die Grundlage für die Unterscheidung zwischen einer legitimen und einer schadhaften Webseite bilden [MoTM15]. Dieser Ansatz weist zwar eine höhere Falsch-Positiv-Rate auf, liefert aber dennoch bessere Resultate als ein Blacklist-basierter Ansatz [GuAP17].

Visuelle Ähnlichkeiten-basierte Lösungen vergleichen das Erscheinungsbild einer verdächtigen Webseite mit jenen des dazugehörigen Originals [JaGu17]. Die Ähnlichkeit zwischen einer Phishing-Webseite und der entsprechenden legitimen Seite stellt einen äußerst wichtigen Aspekt in Bezug auf den Erfolg eines Phishing-Angriffes dar. Wenn die Erscheinungsbilder hierbei nicht nahezu ident sind, steigt folglich die Wahrscheinlichkeit der Aufdeckung des Betruges [GuAP17]. Die hierzu vorhandenen Ansätze vergleichen unter anderem die HTML Document Object Model (DOM) Baumstruktur, verschiedene visuelle Merkmale (Textinhalte als auch Textmerkmale, wie z.B. Schriftart, Schriftfarbe, usw.), den Cascading Style Sheet (CSS) Code oder Bilder bzw. Screenshots, oder basieren auf der menschlichen Wahrnehmung von visuellen Komponenten und machen sich die sechs Gestaltprinzipien zunutze [JaGu17].

Sonstige Lösungen können keinem der drei bisher genannten Ansätzen zugeordnet werden. Sie sind meist nicht jüngst entstanden, sondern kamen im Zuge der Evolution der Internettechnologien auf und haben dennoch eine historische Auswirkung auf Phishing-Angriffe. Beispiele dazu sind unter anderem Lösungen, welche den Versuch des Austausches von Sicherheitsanzeigen (meist in Form von Logos und Icons) im Browser erkennen und vermeiden, oder Lösungen, welche im Zuge der Authentifizierung das Browserfenster des Benutzers individuell gestalten (z.B. mittels eines persönlich zugewiesenen Hintergrundbildes), um so gefälschte Webseiten relativ einfach visuell erkennen zu können (in der Literatur bekannt unter „Dynamic Security Skins“) [GuAP17].

Beispiele von bereits umgesetzten Lösungen zu den unterschiedlichen Ansätzen werden in Kapitel 4.1 betrachtet.

Digital signierte E-Mails: Zur Sicherstellung der Integrität einer Nachricht und somit zur Erkennung von jeglichen Manipulationsversuchen, können mit Hilfe von Public-Key-Verschlüsselungsverfahren sämtliche E-Mails digital signiert und verifiziert werden. Diese Funktionalität wird mittlerweile von nahezu allen gängigen E-Mail-Anwendungen unterstützt und erfordert die Generierung eines öffentlichen/privaten Schlüsselpaares für jeden Benutzer. In Folge kann unter der Verwendung des privaten Schlüssels eine E-Mail digital signiert und unter der Verwendung des dazugehörigen öffentlichen Schlüssels diese Signatur wiederum verifiziert werden. Wenn der öffentliche Schlüssel des Absenders dem Empfänger in authentischer Form vorliegt und die Verifikation erfolgreich war, kann eine Veränderung der Nachricht auf dem Transportweg ausgeschlossen werden [Ollm07].

Serverebene

Die Serverebene zielt auf eine Absicherung aller über das Internet zugänglichen Systemen sowie kundenspezifischen Anwendungen ab. Durch die Implementierung von Anti-Phishing-Techniken in sämtlichen unternehmerischen Webanwendungen können Benutzer von Seiten des Unternehmens vor der Gefahr eines Phishing-Angriffes bewahrt werden [Ollm07]. Nachfolgend werden

die wichtigsten Gegenmaßnahmen, welche auf dieser Ebene zur Gewährleistung eines Schutzes berücksichtigt werden sollen, näher betrachtet.

Verbesserung des Kundenbewusstseins: Das Kundenbewusstsein stellt einen äußerst kritischen Aspekt in Bezug auf die Verhinderung eines Phishing-Angriffes dar. Ein Unternehmen sollte ständig seine Kunden und Anwendungsbenutzer über die Gefahren eines derartigen Angriffes und den möglichen Präventivmaßnahmen informieren [Ollm07]. Auch wenn das jeweilige Unternehmen beispielsweise kein Finanzinstitut oder Internet Service Provider ist, sollte dennoch eine E-Mail-Richtlinie bezüglich der Kommunikation mit Kunden vorhanden sein. Diese sollte den grundlegenden Inhalt bzw. Aufbau einer Nachricht vom Unternehmen zum Kunden festlegen und sollte auch von jedem Mitarbeiter strikt durchgesetzt werden. Typische darin enthaltene Punkte sind in etwa die Untersagung des Versendens von HTML-E-Mails, Anhängen, Links oder Fragen bezüglich sensiblen Informationen [LiVi05]. Außerdem sollten die Benutzer auch auf kritischen Anmeldeseiten auf die Art und Weise, wie das Unternehmen sicher mit seinen Kunden kommuniziert, hingewiesen werden. Des Weiteren sollten auch Möglichkeiten zur Meldung von Phishing-Angriffen und verdächtigen E-Mails im Namen des jeweiligen Unternehmens zur Verfügung gestellt werden [Ollm07].

Validierungsmöglichkeiten für Kommunikationen: Eng verbunden mit dem zuvor beschriebenen Kundenbewusstsein, kann ein Unternehmen mehrere Maßnahmen ergreifen, um Kunden die Validierung einer offiziellen Kommunikation zu erleichtern. Es sollten zunächst sämtliche E-Mails an Kunden personalisiert gestaltet werden. Dazu kann in der Nachricht entweder auf den vollständigen Namen des Kunden oder auf andere persönliche Informationen, die das Unternehmen mit ihm teilt, verwiesen werden (z.B. die Verwendung der Anrede „Sehr geehrter Herr Mustermann“ anstatt „Sehr geehrter Kunde“). Sich auf eine vorherige Nachricht zu beziehen (wenn möglich), kann ebenfalls für den Aufbau einer Vertrauensbasis hilfreich sein. Wie bereits auf der Benutzerebene angesprochen, können auch auf dieser Ebene von Seiten des Unternehmens digital signierte E-Mails verwendet werden. Dazu muss jedoch sichergestellt werden, dass die Kunden bzw. Benutzer auch den richtigen Umgang mit diesen Nachrichten verstehen und die korrekte Durchführung der Validierung beherrschen. Schlussendlich können noch Validierungsportale auf der Webseite des Unternehmens für die Kunden zur Verfügung gestellt werden. Hierbei kann der Benutzer den Inhalt einer empfangenen E-Mail in ein Textfeld hineinkopieren und die Anwendung liefert ihm die Authentizität der Nachricht [Ollm07].

Validierung von Eingabedaten: Eine mangelhaft implementierte oder gar nicht vorhandene Eingabevalidierung von Benutzerdaten ist eine der häufigsten Sicherheitslücken in webbasierten Anwendungen. Der Grundsatz lautet hier, niemals Eingabedaten von Benutzern oder anderen Anwendungskomponenten zu vertrauen. Diese Daten sollten stets vor der Verarbeitung escaped bzw. maskiert werden und niemals in ihrer ursprünglichen Form direkt an den Benutzer zurückgegeben werden [Ollm07].

Sicheres Session-Handling: Wie bereits zuvor erwähnt, sind HTTP und HTTPS zustandslose Protokolle und erfordern deshalb die Implementierung einer Session-Handling-Routine, welche wiederum anfällig für Session Hijacking Angriffe sein kann. Um möglichen Gefahren vorzubeugen, sollten unter anderem Sitzungsinformationen niemals in der URL übertragen

und akzeptiert werden, Session-IDs immer mit einer Ablaufzeit versehen werden, eine aktive ID jederzeit widerrufen werden können, widerrufene IDs nicht wiederverwendet werden sowie sämtliche Versuche der Übermittlung einer ungültigen ID (z.B. einer abgelaufenen oder widerrufenen) zu einer Weiterleitung auf die Anmeldeseite führen und im Zuge dessen eine neue Session-ID ausgestellt werden [Ollm07].

Starker Authentifizierungsprozess: Ein sicherer Authentifizierungsprozess ist enorm wichtig und kann einen erheblichen Beitrag zur Verhinderung eines erfolgreichen Phishing-Angriffes beisteuern. Generell kann eine Authentifizierung auf drei verschiedenen Faktoren basieren (vgl. [LiVi05]):

- Etwas, das der Benutzer weiß, wie z.B. ein Passwort
- Etwas, das der Benutzer besitzt, wie z.B. eine Smart Card
- Etwas, das der Benutzer ist, wie z.B. ein Fingerprint

Wenn der Authentifizierungsprozess nicht nur einen, sondern zwei dieser drei Faktoren erfordert, handelt es sich um eine Zwei-Faktor-Authentifizierung [LiVi05]. Diese kann auf unterschiedliche Arten umgesetzt bzw. implementiert werden: Bei vielen E-Commerce Webanwendungen bekommt der Benutzer beispielsweise einen Hardwaretoken zur Erzeugung einer sich ständig ändernden Komponente, welche zusätzlich in die Anmeldedaten einfließt und dadurch ein One-Time-Passwort erzeugt wird. Weil dieses Passwort nur einmal verwendet werden kann, wird der Benutzer und sein Konto auch im Falle der Aufzeichnung seiner Anmeldeinformationen durch einen Angreifer geschützt [Mill05][Ollm07]. Banken setzen bei Online-Banking Webanwendungen wiederum auf die Verwendung von Transaktionsnummern (TANs) [Mill05]. Hierbei sind die Anmeldedaten der erste Faktor sowie die meist per SMS erhaltene Transaktionsnummer der zweite.

Enterprise-Ebene

Die Enterprise-Ebene umfasst Schutzmaßnahmen für verteilte Technologien und Managementdienstleistungen von Drittanbietern. Diese können von Unternehmen und Internet Service Providern vorgenommen werden, um Kunden als auch interne Benutzer vor Phishing-Angriffen zu bewahren. Die auf dieser Ebene vorhandenen Sicherheitslösungen stehen meist im Zusammenhang mit benutzer- und serverseitigen Lösungen und bieten zusammen einen soliden Schutz gegen Phishing und ähnliche Bedrohungen. In Folge werden die wichtigsten Sicherheitsmaßnahmen dieser Ebene zusammengefasst (vgl. [Ollm07]).

Mail-Server Authentifizierung: Grundsätzlich sollte sich der Mail-Server des Absenders immer gegenüber jenen des Empfängers authentifizieren. Dadurch kann Letzterer überprüfen, ob die IP-Adresse des Ersteren für die angegebene E-Mail-Domain autorisiert ist oder nicht. Sollte dies nicht der Fall sein, wird die empfangene E-Mail verworfen und gelangt somit erst gar nicht in die Mailbox des Empfängers.

Digital signierte E-Mails: Wie auch auf den beiden anderen Ebenen, können auch auf dieser Ebene Einstellungen bezüglich digital signierten E-Mails vorgenommen werden. Hierzu können sämtliche E-Mail-Signaturen bereits vom Empfänger-Mail-Server automatisch validiert werden noch bevor sie den eigentlichen Empfänger erreichen. Dadurch kann dieser im Vorhinein vor ungültigen und unsignierten Nachrichten gewarnt werden.

Domainüberwachung: Ein Unternehmen sollte auch ständig die Registrierung von Domänen, welche im Zusammenhang mit ihrem Unternehmensnamen oder -marken stehen, überwachen. Einerseits muss die Gültigkeitsdauer von eigenen Domänen berücksichtigt und eine Verlängerung bei Bedarf ehest möglich beantragt werden. Ansonsten könnte die Domäne von einer anderen Partei (möglicherweise von einem Angreifer) erworben werden. Andererseits muss auch auf die Registrierung von ähnlich benannten Domänen geachtet werden, welche in Folge ebenfalls für (be)trügerische Zwecke verwendet werden könnten.

Gateway-Schutzdienste: An den Grenzen des eigenen Netzwerkes können unterschiedliche Gateway-Schutzdienste zur Überwachung und Kontrolle des eingehenden und ausgehenden Netzwerkverkehrs eingerichtet werden. Dazu zählen unter anderem Gateway Anti-Virus/Anti-Spam Scanning zur Erkennung von Malware und Spams, Gateway Content Filtering zur Erkennung von verdächtigen Inhalten oder Proxy Services zur Kontrolle von Verbindungen und zum Schutz des internen Netzwerkes.

3.6.2 Nichttechnische Gegenmaßnahmen

Technische Gegenmaßnahmen können zwar eine grundlegende Basis für die Bekämpfung von Phishing bilden, jedoch bei weitem nicht alle Angriffe blocken. Aufgrund der Tatsache, dass menschliches Fehlverhalten die Hauptursache für Datenschutzverletzungen in der Systemsicherheit ist, sollte auf die nichttechnischen Gegenmaßnahmen und vor allem auf den Faktor Mensch ein besonderes Augenmerk gelegt werden [Ali15]. Weil viele Benutzer erst gar nicht wissen, wie Phishing funktioniert und wie komplex solche Angriffe sein können, liegt der Schlüssel zu deren erfolgreichen Bekämpfung in der Benutzerbildung und -aufklärung [GuAP17][MoTM15]. Mit gezielten Schulungen und Ausbildungen muss auf die davon ausgehende Gefahr hingewiesen werden, um so das Sicherheitsbewusstsein jedes Einzelnen zu steigern [Ali15].

Aber auch die Benutzer selbst können eine Reihe von unterschiedlichen Maßnahmen berücksichtigen, um einem Phishing-Angriff nicht zum Opfer zu fallen (vgl. [Ollm07][Kapo15]):

- Wenn man sich bezüglich des Ursprunges einer E-Mail nicht sicher ist, sollte man niemals die darin enthaltenen Links öffnen. Stattdessen kann Kontakt zum in der Nachricht angegebenen Unternehmen aufgenommen werden (entweder per Telefon oder mit Hilfe der Webseitenadresse), um die Echtheit der Nachricht zu überprüfen.
- Man sollte niemals Dateien von einer nicht vertrauenswürdigen Webseite herunterladen. Diese könnten eine beliebige Art von Malware mit Schadenspotential enthalten.
- Man sollte auch niemals auf HTML-E-Mails mit eingebetteten Formularen antworten. Auch wenn diese legitim sind, könnten die im diesen Zuge angegebenen Daten unverschlüsselt übermittelt werden.
- Das Versenden von Personen- oder Finanzdaten per E-Mail sollte vermieden werden.
- Wann auch immer die Eingabe von Personen- oder Finanzdaten auf Webseiten erforderlich ist (z.B. beim Online-Banking), sollte stets auf das Schloss-Symbol in der Statusleiste des Browsers und die Verwendung von HTTPS geachtet werden. Anderenfalls ist eine sichere Datenübertragung nicht gewährleistet.
- Wenn eine Webseite über ein Zertifikat verfügt, sollte sichergestellt werden, dass dieses auch von einer vertrauenswürdigen Zertifizierungsstelle ausgestellt wurde.

4 Vorgeschlagenes Anti-Phishing-Konzept

In diesem Kapitel wird ein umfassendes Anti-Phishing-Konzept zur Erkennung von Phishing-Webseiten vorgeschlagen. Um aufzuzeigen, dass ein derart holistischer Ansatz in der Literatur nicht wiederzufinden ist, werden zunächst ein paar bereits vorhandene Anti-Phishing-Lösungen betrachtet und daraufhin die Architektur sowie der Analyseprozess des in dieser Arbeit vorgeschlagenen Konzepts präsentiert.

4.1 Related Work

Die Palette an verschiedenen bereits vorhandenen Anti-Phishing-Lösungen zur Erkennung von Phishing-Webseiten ist relativ breit gefächert.

Google Safe Browsing APIs: Google Safe Browsing APIs [Goo] ermöglichen Client-Anwendungen die Abgleichung einer URL mit der Blacklist von Google, welche alle bekannten unsicheren Web-Ressourcen enthält und laufend aktualisiert wird. Mit Hilfe der Lookup API kann eine URL zur Überprüfung ihres Status zum Google Safe Browsing Server gesendet werden und der Benutzer erhält eine Antwort in Form von sicher oder unsicher. Die Update API kann hingegen genutzt werden, um eine verschlüsselte Version der Blacklist herunterzuladen und somit URLs lokal in der Client-Anwendung überprüfen zu können.

PhishNet: PhishNet [PKKG10] ist ein Blacklist-basierter Ansatz bestehend aus zwei Hauptkomponenten. Die URL-Vorhersagekomponente (engl. URL prediction component) generiert mit Hilfe von fünf verschiedenen Heuristiken neue schadhafte URLs auf Basis von existierenden Blacklist-Einträgen. Diese durchlaufen anschließend einen Validierungsprozess und werden bei bestehendem Verdacht auf eine Phishing-Webseite zur Blacklist hinzugefügt. Die approximative URL-Abgleichungskomponente (engl. approximate URL matching component) vergleicht eine URL mit den Einträgen in der Blacklist mit Hilfe von regulären Ausdrücken und Hashmaps. Dadurch kann eine Phishing-Webseite auch erkannt werden, wenn die tatsächliche URL nicht exakt mit einem Eintrag in der Blacklist übereinstimmt.

Jain und Gupta: Jain und Gupta [JaGu16] schlugen ein Whitelist-basiertes Browser-Plugin vor, welches aus zwei Modulen besteht. Die Einträge der sich automatisch aktualisierenden Whitelist bestehen jeweils aus einem Domänennamen und der dazugehörigen IP-Adresse. Sobald eine Webseite besucht wird, vergleicht das URL- und DNS-Vergleichsmodul, ob sich die jeweilige Domäne bereits in der Whitelist befindet. Sollte dies der Fall sein, wird die dazugehörige IP-Adresse noch mit jener in der Whitelist abgeglichen. Bei einer Übereinstimmung handelt es sich um eine legitime und anderenfalls um eine Phishing-Webseite.

Sollte die Domäne jedoch noch nicht in der Whitelist aufscheinen, wird das Phishing-Identifizierungsmodul aktiv. Dieses entscheidet anhand von bestimmten Eigenschaften der Hyperlinks, ob es sich um eine Phishing-Webseite handelt oder nicht und erweitert im letzteren Fall die Whitelist um einen weiteren Eintrag.

SpoofGuard: Das Browser-Plugin SpoofGuard [CLTM+04] führt für jede angesurfte Webseite eine Reihe von verschiedenen Tests durch und berechnet aus den einzelnen Testergebnissen ein Gesamtergebnis (Total Spoof Score - TSS). Dieses wird basierend auf vom Benutzer konfigurierten Schwellwerten visuell in Form eines grünen, gelben oder roten Ampelsymbols im Browser angezeigt. Im Zuge der Tests wird einerseits sowohl eine zustandslose als auch eine zustandsbehaftete Seitenevaluierung und andererseits auch eine Überprüfung der zum Server gesendeten Daten durchgeführt. Bei der zustandslosen Evaluierung werden die URL, enthaltene Bilder und Links sowie die Zertifikate der Webseite bewertet. Die zustandsbehaftete Evaluierung beurteilt die aktuelle Webseite auf Basis eines Vergleiches mit gespeicherten Verlaufsdaten (z.B. ob die derzeitige Domäne einer bereits besuchten ähnelt). Bei der Evaluierung der zum Server übermittelten Daten wird der Benutzer gewarnt, sobald er versucht sensible Informationen in ein Formular einer Webseite mit zu hohem TSS einzugeben.

CANTINA: Zhang et al. [ZhHC07] implementierten eine Heuristik-basierte Browser-Erweiterung namens CANTINA. Diese bewertet eine Webseite anhand von insgesamt acht verschiedenen Heuristiken, welche Aspekte der Domäne bzw. der URL, des HTML-Codes und des Textinhaltes bewerten. Die Implementierung liefert für jede dieser Heuristiken entweder den Wert -1, wenn der Verdacht auf eine Phishing-Webseite besteht oder +1 anderenfalls. Zur Berechnung des Gesamtergebnisses kommt ein einfacher linearer Klassifikator zum Einsatz. Hierbei wird die gewichtete Summe der Einzelergebnisse zusammengezählt und eine Webseite als legitim klassifiziert, falls der Wert größer als Null ist und als Phishing-Webseite anderenfalls.

CANTINA+: Aufbauend auf CANTINA stellten Xiang et al. [XHRC11] in einer späteren Arbeit CANTINA+ vor. Diese Lösung verwendet vier Heuristiken von CANTINA, erweitert eine weitere davon und fügt insgesamt zehn andere hinzu. Sechs dieser Features beziehen sich auf die URL der Webseite, vier auf den HTML-Inhalt und die restlichen fünf durchsuchen das Internet nach Informationen über die zugrundeliegende Webseite. Anstelle eines linearen Klassifikators verwendet CANTINA+ hingegen einen mittels eines maschinellen Lernverfahrens trainierten Klassifikator. Um die Falsch-Positiv-Rate zusätzlich zu reduzieren, werden noch vor der Extraktion der Features und der anschließenden Klassifikation zwei Filter angewendet. Der Erste überprüft auf Basis des Vergleichs von Hashwerten, ob es sich bei der aktuellen Webseite um ein Duplikat einer bereits bekannten Phishing-Webseite handelt. Der Zweite filtert hingegen alle Webseiten, welche über keine (zumindest erkennbaren) Anmeldeformulare verfügen.

PhishShield: Rao und Ali [RaAl15] entwickelten eine Desktop-Anwendung namens PhishShield, welche ebenfalls einen Heuristik-basierten Ansatz verfolgt und zusätzlich über eine Whitelist verfügt. Im ersten Schritt wird die Domäne der URL mit dieser Whitelist abgeglichen und bei einer Übereinstimmung als legitim klassifiziert. Anderenfalls wird im nächsten Schritt die Webseite auf Basis von vier verschiedenen Heuristiken bewertet. Die-

se überprüfen, ob der Body der Webseite Links beinhaltet, ob im Footer Links auf NULL (#) verweisen, ob die Copyright-Informationen und der Titel der Seite die Domäne beinhalten und ob die Domäne mit einer auf Basis der enthaltenen Hyperlinks berechneten Webseitenidentität übereinstimmt. Sollte auch nur eine dieser Heuristiken als positiv bewertet werden, wird die Webseite als Phishing-Webseite eingestuft.

Netcraft Toolbar: Die Netcraft Toolbar [Net04] zeigt dem Benutzer unterschiedliche Informationen zur derzeit besuchten Webseite an. Dazu zählen unter anderem der Registrierungszeitpunkt der Domäne, in welchem Land diese gehostet ist und die Popularität der Seite unter allen Toolbar-Benutzern. Weiters besteht auch die Möglichkeit, sich durch einen Klick auf einen Link einen Gesamtreport der Webseite anzeigen zu lassen. Dieser umfasst detailliertere Informationen bezüglich der Domäne, der SSL/TLS-Konfiguration, der Zertifikatskette, der Hosting-Historie, usw. Das Kernstück dieser Browser-Erweiterung ist jedoch eine Risikobewertung (engl. Risk Rating), welche dem Benutzer in der Toolbar in Form eines farbigen Balkens angezeigt wird. Diese Bewertung basiert ausschließlich auf Eigenschaften der URL bzw. der Domäne der Webseite und bezieht unterschiedliche Faktoren in die Berechnung mit ein (wie z.B. ob ein Hostname, eine IP-Adresse oder eine Portnummer in der URL vorkommt). Als dominantester Faktor wird hierbei jedoch das Alter der Domäne herangezogen, da Phishing- im Gegensatz zu legitimen Webseiten typischerweise sehr kurzlebig sind.

AntiPhish: Kirda und Kruegel [KiKr05] entwickelten ein Browser-Plugin namens AntiPhish, welches den Datenfluss von sensiblen Informationen (wie z.B. Passwörter) überwacht. Dazu muss der Benutzer dem Plugin zunächst mitteilen, dass die eingegebenen Informationen auf der aktuellen Webseite vertraulich sind und somit vor Phishing-Angriffen geschützt werden sollen. Sobald anschließend Daten in ein Formular dieser Seite eingegeben werden, durchsucht das Plugin sämtliche HTML-Textfelder vom Typ Passwort und speichert dessen Inhalt zusammen mit einem Verweis auf die Domäne der aktuellen Webseite. Wann auch immer der Benutzer fortan auf einer beliebigen Seite in ein Textfeld vom Typ Text oder vom Typ Passwort Daten eingibt, wird im Hintergrund überprüft, ob diese Daten mit bereits gespeicherten Passwörtern übereinstimmen. Wenn dies der Fall ist und die aktuelle Webseite nicht der zum Passwort gespeicherten Domäne angehört, wird ein Phishing-Versuch vermutet und der Benutzer entsprechend gewarnt.

PwdHash: PwdHash ist eine weitere Browser-Erweiterung, welche von Ross et al. [RJMB+05] publiziert wurde. Im Zuge einer Authentifizierung wird mit Hilfe einer kryptografischen Hashfunktion für jede Webseite ein unterschiedliches Passwort generiert. Dazu wird der Inhalt eines Passwortfeldes erfasst und aus diesem und der Domäne der aktuellen Webseite ein Hashwert abgeleitet, welcher anstelle des Klartext-Passwortes an den Server gesendet wird. Sollte der Benutzer sein Passwort auf einer Phishing-Webseite eingeben, ist das vom Angreifer empfangene Passwort somit für alle anderen Domänen nutzlos. Wie jedoch auch die Autoren angeben, schützt PwdHash nicht vor lokal installierten Schadprogrammen (wie z.B. Spyware und Keyloggers) sowie DNS- und bestimmten JavaScript-Angriffen.

BogusBiter: Yue und Wang [YuWa10] entwickelten die Browser-Erweiterung BogusBiter, welche eine vermutete Phishing-Webseite mit einer relativ großen Anzahl an gefälschten Anmeldedaten befüllt. Zu deren Erkennung ist BogusBiter jedoch auf eine andere clientseiti-

ge Lösung angewiesen und zählt somit nicht zu den präventionsbasierten Ansätzen. Erst wenn eine Anmeldeseite von einem anderen Tool als Phishing-Webseite eingestuft wurde, wird die Erweiterung aktiv und warnt den Benutzer zunächst. Entscheidet dieser sich, die Warnung zu ignorieren, werden im Zuge der Übermittlung der tatsächlich eingegebenen Anmeldedaten zusätzlich auch noch $S - 1$ gefälschte Anmeldedaten an die Webseite gesendet. Wenn der Angreifer die gesammelten Anmeldeinformationen anschließend auf der legitimen Webseite verifiziert, können dadurch gestohlene Zugangsdaten zeitgerecht erkannt werden. Ignoriert der Benutzer die Warnung jedoch nicht, werden die tatsächlichen Anmeldedaten abgefangen und nur S gefälschte Zugangsdaten der Reihe nach an die Webseite gesendet.

PhishWHO: Tan et al. [TCWS16] präsentierten mit PhishWHO eine Lösung, welche eine Phishing-Webseite hinsichtlich der Differenz zwischen dem Domainnamen der aktuellen und der abgezielten (legitimen) Webseite erkennt. Dazu wird zuerst überprüft, ob die zugrundeliegende Webseite beliebige Eingabefelder besitzt oder nicht. Sollte dies nicht der Fall sein, wird sie als harmlos und somit legitim eingestuft. Anderenfalls werden mit Hilfe eines N-Gramm-Modells aus dem Klartext der Webseite Identitätsschlüsselwörter (engl. website identity keywords) extrahiert. Diese werden anschließend einer Suchmaschine übergeben und auf Basis der zurückgelieferten Suchergebnisse mit Hilfe von drei verschiedenen Features der Domainnamen der abgezielten Webseite bestimmt. Zuletzt wird noch mittels eines Drei-Schichten-Identitäts-Abgleichsystems (engl. 3-tier identity matching system) die aktuelle Domäne mit der abgezielten Domäne verglichen und aufgrund dessen über die Legitimität der zugrundeliegenden Webseite entschieden.

Medvet et al.: Medvet et al. [MeKK08] publizierten einen auf visuelle Ähnlichkeiten-basierten Ansatz, welcher unter anderem von AntiPhish inspiriert wurde. Anhand von bestimmten Features, welche alle sichtbaren Textabschnitte, alle sichtbaren Bilder und das visuelle Gesamterscheinungsbild einer Webseite berücksichtigen, wird eine Signatur extrahiert. Diese Signatur wird mit jener der legitimen Seite verglichen und ein Ähnlichkeitswert berechnet, welcher schließlich aussagt, ob es sich bei der zugrundeliegenden Webseite um eine legitime oder eine Phishing-Webseite handelt.

FeedPhish: Srinivasa Rao und Pais [SrPa17] schlugen mit der Anwendung FeedPhish einen Ansatz vor, welcher auf Basis der Automatisierung des menschlichen Verhaltens bei der Eingabe von sensiblen Daten Phishing-Webseiten erkennt. Dazu wird zunächst überprüft, ob die Webseite der zugrundeliegenden URL eine Anmeldemöglichkeit für den Benutzer zur Verfügung stellt. Sollte dies der Fall sein, werden die Eingabefelder für Benutzernamen und Passwort identifiziert und gefälschte Anmeldedaten an den Webserver gesendet. Sollte die Nachfolgeseite über kein Passwortfeld verfügen, wurden die Anmeldedaten höchstwahrscheinlich nicht validiert und die Webseite wird als Phishing-Webseite eingestuft. Sollte ein Passwortfeld vorhanden sein, wird die Domäne der Anmeldeseite mit jener der Nachfolgeseite verglichen und im Falle von Abweichungen ebenfalls die Webseite als Phishing-Webseite kategorisiert. Bei einer Übereinstimmung besitzt die Nachfolgeseite einerseits ein Passwortfeld und stammt andererseits auch von der selben Domäne. In diesem Fall wird die Nachfolgeseite noch auf das Vorhandensein von zwei Features hin untersucht und abhängig davon schlussendlich ebenfalls als legitim oder Phishing-Webseite eingestuft.

4.2 Architektur

Die Architektur besteht aus einem Webbrowser-Plugin (Client) und einem REST-Server, welche über eine sichere SSL/TLS-Verbindung miteinander kommunizieren (siehe Abbildung 4.1). Durch die Verwendung einer Client-Server-Architektur wird die Analyselogik komplett vom Client entkoppelt, wodurch einerseits die Komplexität bzw. der Funktionsumfang des Webbrowser-Plugins möglichst gering gehalten werden kann und andererseits eine zukünftige Erweiterbarkeit auf beliebige andere Webbrowser ebenfalls mit geringem Aufwand ermöglicht wird.

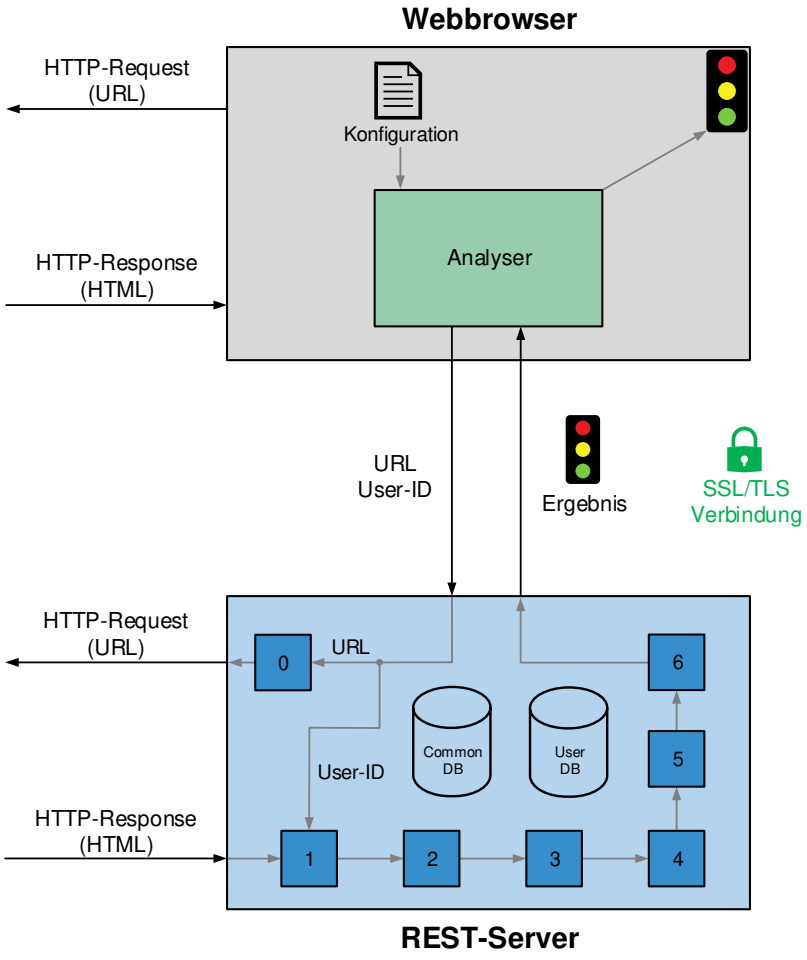

Abb. 4.1: Architektur des Anti-Phishing-Konzepts

Sobald der Benutzer im Webbrowser eine URL aufruft, wird diese automatisch vom Plugin (in Abbildung 4.1 „Analyser“ genannt) übernommen und zusammen mit den anonymisierten User-Informationen an den REST-Server geschickt. In Abhängigkeit der jeweiligen Benutzerkonfiguration werden einzelne Analysetasks am REST-Server getriggert. Bevor die Analyse gestartet werden kann, wird zunächst die zur übermittelten URL dazugehörige HTML-Seite per HTTP-Request vom jeweiligen Webserver angefordert (siehe Abbildung 4.1 Schritt 0). Anschließend startet der eigentliche Analyseprozess, dessen einzelne Analysetasks grob in sechs übergeordnete Schritte eingeteilt werden können (siehe Kapitel 4.3). In diesen wird die zugrundeliegende

Webseite einerseits anhand von verschiedenen Heuristiken und andererseits anhand eines Klassifikators sowie des individuellen Verhaltensmusters des jeweiligen Benutzers bewertet.

Anhand der einzelnen Ergebnisse der unterschiedlichen Prozessschritte wird am Server das Gesamtergebnis berechnet, welches die Webseite wie bei einer Ampel in eine der drei folgenden Kategorien einstuft:

- Grün: Keine bzw. geringe Gefahr – Wahrscheinlichkeit einer Phishing-Webseite gering
- Gelb: Erhöhte Gefahr – Verdacht einer möglichen Phishing-Webseite besteht
- Rot: Hohe Gefahr – Wahrscheinlichkeit einer Phishing-Webseite sehr hoch

Dieses Endergebnis wird schließlich vom Server an das Plugin zurückgesendet und dem Benutzer im Browser in Form eines Ampelsymbols visuell dargestellt. Damit das Browser-Plugin während der Analysedauer des Servers nicht blockiert, ist dieses parallel auszuführen. Des Weiteren soll für den Benutzer auch die Möglichkeit bestehen, die im gesamten Prozess verwendeten Blacklists/Whitelists und Keyword-/Phrasen-Listen beliebig zu erweitern. Dazu kann er mit Hilfe des Plugins für ihn bekannte Phishing-Webseiten als solche kennzeichnen oder einzelne Wörter bzw. Textpassagen visuell markieren. Diese Daten werden anschließend an den Server gesendet und die entsprechende Liste in der Datenbank dementsprechend erweitert.

4.3 Analyseprozess

Sobald die zur übermittelten URL dazugehörige Webseite geladen wurde, startet der Analyseprozess am REST-Server. Wie bereits erwähnt, können die verschiedenen Analysetasks in insgesamt sechs übergeordnete Prozessschritte eingeteilt werden, welche abhängig von der Benutzerkonfiguration in die Analyse miteinbezogen werden oder nicht. Nachfolgend werden die einzelnen Schritte kurz erläutert.

1. **Analyse der URL:** In diesem Analyseschritt wird die URL der entsprechenden Webseite auf Basis von Blacklists/Whitelists sowie das Vorhandensein von bestimmten Features bzw. Merkmalen untersucht. Dazu kommen die folgenden Analysetasks zum Einsatz:
   - Blacklist-Abgleich
   - Whitelist-Abgleich
   - Domänenalter
   - IP-Adresse
   - Verdächtige URL
   - Anzahl der Punkte
   - Anzahl der Satzzeichen
2. **Analyse der Metadaten:** Hierbei werden sämtliche Metadaten (wie z.B. der Titel der Webseite oder die vorhandenen Meta-Tags) auf diverse Auffälligkeiten hin überprüft. Dieser Prozessschritt beinhaltet die folgenden Analysetasks:
   - Verdächtige Meta-Tags
   - Inhalt des Title-Tags

3. **Analyse der Struktur:** Dieser Schritt analysiert die Struktur des HTML-Codes und stellt das Vorhandensein von bestimmten Features bzw. Merkmalen mit Hilfe der folgenden Analysetasks fest:
   - Verdächtige Formulare
   - Eingabefelder
   - Verdächtige Frames
   - Keine Links im Body
   - Leere Links
   - Verdächtige Links
   - Nicht übereinstimmende URLs
   - Nur Script
   - Verdächtiger JavaScript-Code
   - Identität der Webseite
4. **Analyse des Inhaltes:** Bei der Analyse des Inhaltes wird ausschließlich nur der vorhandene Klartext der Webseite auf Basis der folgenden Analysetasks untersucht:
   - Verdächtiger Inhalt (Keywords/Phrasen)
   - Semantik (mit Hilfe von computerlinguistischen Methoden)
5. **Trained Learner (Classifier):** In diesem Schritt ermittelt ein mittels eines maschinellen Lernverfahrens trainierter Klassifikator, ob es sich bei der zu analysierenden Webseite um eine Phishing-Seite handelt oder nicht. Dazu wird dieser mit Hilfe von bekannten Phishing-Webseiten und deren Merkmalen (= Trainings-Set) trainiert und auf Basis dessen ein Modell erstellt, welches für die Klassifizierung der zu analysierenden Seite in Phishing oder legitim herangezogen werden kann.
6. **User Profile Matching:** Zuletzt wird noch überprüft, ob die angeforderte Webseite und deren Inhalte ins aktuelle Profil des jeweiligen Benutzers passt. Dazu wird mit Hilfe einer anonymisierten Identifikation für jeden Benutzer ein Verhaltensmuster erstellt und fortlaufend aktualisiert.

Sämtliche für die Analyse benötigten Daten werden hierbei am REST-Server in zwei verschiedenen Datenbanken gespeichert (siehe Abbildung 4.1). Die Datenbank „Common DB“ beinhaltet dazu sämtliche Daten, welche für alle Analysen gleich und somit benutzerunabhängig sind. Dazu zählen die allgemeine Blacklist und Whitelist sowie die allgemeinen Inhaltslisten. Jegliche benutzerspezifischen Daten werden wiederum in der Datenbank „User DB“ gespeichert. Dazu zählen einerseits die einzelnen Benutzerprofile und andererseits die vom jeweiligen Benutzer definierten Erweiterungen der allgemeinen Listen (Blacklist, Whitelist und Inhaltslisten).

Im Zuge dieser Arbeit wird ein Prototyp für das beschriebene Anti-Phishing Konzept implementiert. Dieser umfasst den Aufbau der gesamten Client-Server Architektur, die clientseitige Implementierung eines Browser-Plugins sowie die serverseitige Implementierung der Analyseschritte 1-4, wobei bei der Analyse des Inhaltes im Prozessschritt 4 die Untersuchung der Semantik und somit die Verwendung von computerlinguistischen Methoden nicht enthalten ist.

5 Implementierung Server

Dieses Kapitel beschreibt die Implementierung der serverseitigen Komponente. Dabei handelt es sich um ein RESTful (Representational State Transfer) Webservice, welches unter der Verwendung von Dropwizard [Dro] umgesetzt ist. Dropwizard ist ein Open-Source Java Framework, mit welchem relativ einfach und schnell zuverlässige, performante und produktionsreife RESTful Webservices implementiert werden können.

5.1 Aufbau von Dropwizard

Dropwizard besteht aus verschiedenen stabilen und ausgereiften Open-Source Bibliotheken und Frameworks aus dem Java-Ökosystem und bündelt diese zu einem einzigen und einfachen Paket zusammen. Die wichtigsten Bibliotheken und Frameworks werden nachfolgend kurz beschrieben.

Jetty: Dropwizard verwendet die Jetty HTTP Bibliothek [Jet] und bettet direkt einen HTTP-Server in die Anwendung ein. Dieser wird in der `main`-Methode hochgefahren und somit ist die Anwendung von keinem Anwendungsserver abhängig. Die gesamte Dropwizard-Applikation wird als ein einfacher Prozess ausgeführt, wodurch man sich einerseits um zahlreiche Aspekte in der Produktionsumgebung, wie z.B. Konfiguration und Wartung eines Anwendungsservers oder Probleme mit dem Klassenlader (engl. class loader), nicht kümmern muss und andererseits sämtliche Unix Prozessmanagement-Tools zur Überwachung verwenden kann.

Jersey: Jersey [Jer] ist ein Open-Source Framework für die Entwicklung von RESTful Webservices in Java und stellt alle JAX-RS (Java API for RESTful Web Services) APIs zur Verfügung und dient als Referenzimplementierung von JAX-RS. Zusätzlich erweitert es den Funktionsumfang von JAX-RS und ermöglicht unter anderem die einfache Implementierung von testbaren Klassen, welche HTTP-Anfragen auf Java-Objekte abbilden.

Jackson: Bei Jackson [Jac] handelt es sich um eine Java-API für die Verarbeitung des Datenformates JSON (JavaScript Object Notation). Damit sind unter anderem die Serialisierung und Deserialiserung von Java-Objekten sowie das Parsen und Generieren von JSON im Allgemeinen möglich.

Metrics: Metrics [Met] ist eine Java-Bibliothek für die Überwachung und Messung des Verhaltens der Anwendung in der Produktionsumgebung.

Guava: Guava [Gua] ist eine Sammlung von Java-Bibliotheken und wurde von Google entwickelt. Sie beinhaltet neue Collections, eine Graphenbibliothek sowie zahlreiche APIs bzw. Utilities für Nebenläufigkeit, Caching, Ein- und Ausgabe, Hashing, uvm.

JDBI: JDBI [JDB] ist eine Java-Bibliothek für den Zugriff und die einfache Verwendung einer relationalen Datenbank via SQL.

Logback: Bei Logback [Log] handelt es sich um ein Java-Framework für performantes und flexibles Logging.

Hybernate Validator: Hybernate Validator [Hyb] ist ein Java-Framework für die Validierung von Benutzereingaben und die Generierung von entsprechenden Fehlermeldungen.

5.2 Allgemeiner Aufbau einer Dropwizard-Anwendung

Ein Dropwizard-Projekt wird Empfehlungen zufolge mit dem Build-Management-Tool Maven [Mav] erstellt und verwaltet. Dazu stellt Dropwizard ein eigenes Projekt-Template (einen sogenannten „Archetype“) zur Verfügung, welches für die Generierung eines Dropwizard-Projektes mit allen benötigten Packages und Dateien verwendet werden kann. Der typische Aufbau bzw. die typische Struktur eines derartigen Projektes ist in Abbildung 5.1 ersichtlich und wird nachfolgend kurz näher betrachtet.

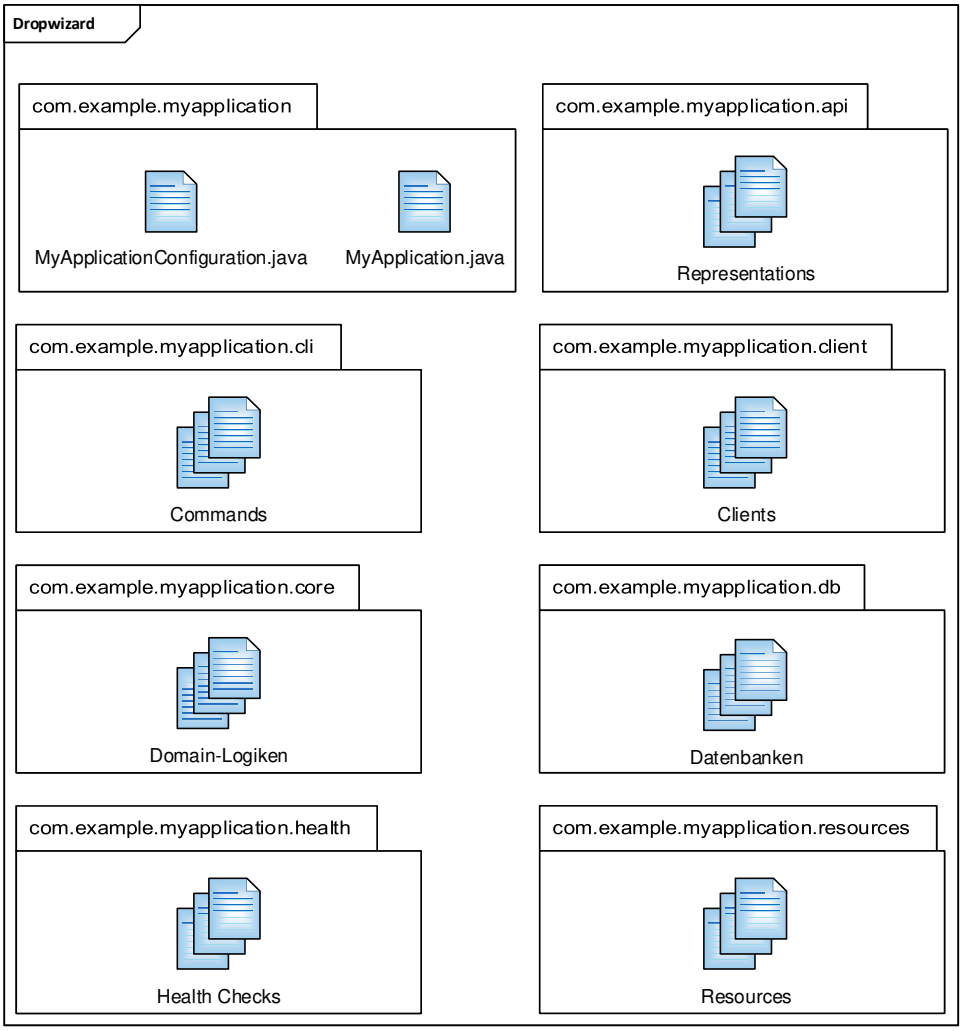

Abb. 5.1: Aufbau einer Dropwizard-Anwendung (vgl. [Dro])

Configuration: Jede Dropwizard-Anwendung besitzt eine Konfigurationsklasse, bei welcher es sich um eine Subklasse der `Configuration`-Klasse handelt. Diese beinhaltet umgebungsspezifische Parameter, welche in einer YAML-Konfigurationsdatei (YAML Ain't Markup Language) definiert und automatisch mittels Jackson zu einer Instanz der Konfigurationsklasse der Anwendung deserialisiert und validiert wird. Dropwizard stellt dazu eine Reihe von bereits vorhandenen build-in Parametern bereit, welche unter anderem eine einfache Konfiguration des Jetty-Servers, des Loggings, der Metriken, der Datenbanken sowie der HTTP-Clients ermöglichen.

Application: Die Applikationsklasse stellt den Haupteinstiegspunkt der Applikation dar und bildet zusammen mit der Konfigurationsklasse den Kern einer Dropwizard-Anwendung. Sie ist eine Subklasse der `Application`-Klasse und bekommt als Typparamter die dazugehörige Konfigurationsklasse übergeben. Ihre Aufgabe besteht in der Einrichtung der Laufzeitumgebung sowie des Startens und des Ausführens der gesamten Anwendung.

Representations: Representation-Klassen entsprechen den Entitäten der API und befinden sich im Package `api`. Dropwizard bevorzugt hier zwar stark JSON als Darstellungsformat, es können jedoch alle POJOs (Plain Old Java Object) auf beliebige Formate serialisiert und deserialsiert werden.

Commands: Grundsätzlich wird empfohlen, eine Dropwizard-Anwendung als „Fat-Jar-Datei" (eine einzige `.jar`-Datei, welche alle notwendigen Dateien zur Ausführung der Applikation beinhaltet) mit Maven zu erstellen. Dadurch kann die Anwendung ganz einfach von der Kommandozeile aus mit Hilfe dieser Jar-Datei und des folgenden Befehls gestartet werden:

```
java -jar <jarfilename>.jar server <configfilename>.yml
```

Dabei entspricht `server` dem einzigen standardmäßig verfügbaren Command, welcher bei der Ausführung als Argument angegeben werden kann und zusätzlich die Angabe einer YAML-Konfigurationsdatei erfordert. Mit Hilfe dieses Kommandos wird die Anwendung beispielsweise als HTTP-Server mit den in der Konfigurationsdatei enthaltenen Einstellungen gestartet. Dropwizard erlaubt jedoch auch die Erstellung von eigenen Commands, welche Subklassen der `Command`-Klasse darstellen und sich im Package `cli` befinden. Sie ermöglichen das Ausführen von bestimmten Aktionen basierend auf Argumenten, welche in der Befehlszeile beim Starten der Anwendung angegeben werden.

Clients: Im Package `client` befindet sich wiederum Client-Code, welcher auf externe HTTP-Services zugreift. Dazu stellt Dropwizard den HTTP-Client von Apache[1] und jenen von Jersey[2] zur Verfügung. Dadurch können externe Webservices in die eigene Anwendung integriert bzw. eingebunden werden.

Domain-Logiken: Domain-Logiken befinden sich im Package `core` und sind Klassen, welche Domain-Implementierungen enthalten. Darunter fallen sämtliche Objekte, die nicht in der API verwendet werden (wie z.B. POJOs, Validierungen, Verschlüsselungen, usw.).

Datenbanken: Für jene Klassen, welche auf Datenbanken zugreifen, ist das Package `db` vorgesehen. Dazu wird empfohlen, die in Dropwizard enthaltene JDBI-Bibliothek zur Erstellung von DAO-Klassen (Data Access Objects) als Interfaces zu verwenden.

[1] `http://hc.apache.org/httpcomponents-core-4.3.x/index.html`
[2] `https://jersey.github.io/documentation/2.24/client.html`

Health Checks: Bei Health Checks handelt es sich um Laufzeit-Tests, mit Hilfe dessen die korrekte Funktionsweise der Dropwizard-Anwendung in der Produktionsumgebung überprüft werden kann. Sie stellen eine Subklasse der `HealthCheck`-Klasse dar und befinden sich im Package `health`. Standardmäßig verfügt jede Dropwizard-Anwendung über den build-in Health Check `deadlock`, welcher überprüft, ob sich die vorhandenen Threads in einem Deadlock-Zustand befinden oder nicht. Durch das Senden eines GET-Requests an den REST-Endpunkt `/healthcheck` des Admin-Ports werden alle Health Checks ausgeführt und deren Ergebnisse retourniert.

Resources: Resources sind Subklassen der `Resource`-Klasse und befinden sich im Package `resources`. Sie modellieren die bereitgestellten Ressourcen der RESTful API und sind jeweils einem URI (Uniform Resource Identifier), über welchen sie erreichbar sind, zugeordnet. Wird nun auf eine Ressource zugegriffen, wird die eingehende HTTP-Anfrage auf ein Java-Objekt abgebildet und das entsprechende Ergebnis (in der Regel ein Representation-Objekt) wiederum auf die dazugehörige HTTP-Antwort. Wie bereits zuvor beim Aufbau von Dropwizard (siehe Kapitel 5.1) erwähnt, wird dazu Jersey verwendet.

5.3 Aufbau der implementierten Dropwizard-Anwendung

Neben einer Applikations- und einer Konfigurationsklasse, verfügt die Anwendung über eine `Resource`-Klasse namens `AnalysisResultResource`, einer dazugehörigen `Representation`-Klasse namens `AnalysisResult` sowie einer `HealthCheck`-Klasse namens `ComputerHealthCheck`. Diese Klassen befinden sich jeweils in den dazu vorgesehenen Packages (siehe vorheriges Kapitel). Die Klasse `AnalysisResult` repräsentiert hierzu das vom REST-Server zurückgelieferte Ergebnis einer Analyse und beinhaltet eine eindeutige ID (`id`), das Gesamtergebnis (`finalScore`) sowie das dazugehörige Ampelsymbol (`trafficLight`). Die Klasse `AnalysisResultResource` ist die einzige Ressource, die die Anwendung bereitstellt und ist unter der URI `/analyse` erreichbar. Sie nimmt HTTP-POST-Anfragen entgegen und erwartet im Body eine valide URL, welche die URL der zu analysierenden Webseite darstellt. Anschließend triggert sie den Analyseprozess und sendet das Ergebnis in Form eines `AnalysisResult`-Objektes (welches mittels Jackson in ein JSON-Format serialisiert wird) zurück. Die Klasse `ComputerHealthCheck` repräsentiert einen Health Check, welcher die korrekte Funktionsweise des gesamten Analyseprozesses zur Laufzeit der Anwendung überprüft.

Für die Implementierung der gesamten Analyselogik und der dazu notwendigen Klassen wurden eigene Packages namens `analyse`, `tasks`, `configuration`, `types` und `lists` angelegt. Sämtliche Klassen für das Herunterladen der beiden Korpusse und der darauf basierenden Berechnungen befinden sich hingegen im Package `corpus` (siehe nachfolgendes Kapitel).

5.4 Analyseprozess

In diesem Abschnitt wird der konkrete Analyseprozess am REST-Server näher erläutert. Dazu wird zunächst der Aufbau eines Analysetasks beschrieben und anschließend auf die verschiedenen Tasks und dessen Funktionsweise eingegangen. Danach werden die einzelnen Task-Parameter und Inhaltslisten zusammen mit deren Berechnung bzw. Bestimmung näher erläutert und infolge

dessen der konkrete Ablauf einer Analyse Schritt für Schritt erklärt. Am Ende wird noch auf die Zuordnung eines Analyseergebnisses zum entsprechenden Ampelsymbol eingegangen.

5.4.1 Aufbau eines Analysetasks

Jeder der in Kapitel 4 beschriebenen Analysetasks ist am Server als eine eigene Klasse, welche das `Callable`-Interface erweitert, implementiert. Dadurch ist eine parallele Ausführung aller Tasks in einem Thread Pool mit Hilfe des Java Executor Frameworks möglich. Als Ergebnis liefert jeder dieser Tasks ein `TaskResult`-Objekt, welches den Namen, die Kategorie und den Score des jeweiligen Tasks beinhaltet. Der Name dient dabei der eindeutigen Zuordnung eines Task-Ergebnisses zu einem bestimmten Task. Die Kategorie gibt wiederum an, welchem übergeordneten Prozessschritt dieser Task zugeordnet ist. Dazu wurde ein eigener Enum-Aufzählungstyp namens `Category` angelegt, dessen einzelne Ausprägungen jeweils einen übergeordneten Prozessschritt repräsentieren:

- `URL` – Analyse der URL
- `METADATA` – Analyse der Metadaten
- `STRUCTURE` – Analyse der Struktur
- `CONTENT` – Analyse des Inhaltes
- `UNDEFINED` – Undefiniert (wird nur im Zuge der Berechnung der verschiedenen Grenzwerte als Standardwert benötigt, da hier die Kategorie keine Rolle spielt)

Der Score repräsentiert das berechnete Ergebnis des jeweiligen Tasks und stellt in der Regel eine reelle Zahl zwischen 0 und 1 dar ($Score \in [0, 1]$). 0,0 entspricht hierbei einem negativen Ergebnis und bedeutet, das der Task nicht angeschlagen hat und es sich seiner Analyse nach um eine legitime Webseite handelt. Ein Score größer als 0,0 entspricht hingegen einem positiven Ergebnis und der Wert kann als Maßzahl für die Stärke des Anschlages bzw. für die Wahrscheinlichkeit des Vorliegens einer Phishing-Webseite aus Sicht dieses Tasks angesehen werden. Sollte während der Ausführung eines Tasks jedoch ein Fehler auftreten (entweder erwartet oder unerwartet), so wird dieser Umstand mittels eines Scores von -1,0 im `TaskResult`-Objekt repräsentiert.

5.4.2 Beschreibung der verschiedenen Analysetasks

Nachfolgend werden die einzelnen Analysetasks der verschiedenen übergeordneten Prozessschritte (= Kategorien) näher beschrieben. Sämtliche dazu verwendeten Heuristiken, welche aus der Literatur stammen (teilweise jedoch adaptiert wurden), sind durch die Angabe der entsprechenden Quelle als solche gekennzeichnet. Für das Parsen und die Extraktion der dazu notwendigen Daten aus dem HTML-Code der zugrundeliegenden Webseite wurde der Java HTML-Parser jsoup[3] verwendet.

Analyse der URL

Analysetasks dieser Kategorie untersuchen die URL der angeforderten Webseite anhand der nachfolgenden Heuristiken bzw. Kriterien.

Blacklist-Abgleich: Der Server verfügt über eine interne Blacklist, welche URLs von bereits bekannten Phishing-Webseiten beinhaltet. Als Quelle dieser Blacklist wurde Phis-

[3] `https://jsoup.org/`

hTank[4] herangezogen – eine Community-basierte Webseite, auf welcher jeder Benutzer Phishing-Seiten melden kann und andere Benutzer daraufhin verifizieren können, ob es sich tatsächlich um eine Phishing-Seite handelt oder nicht. PhishTank stellt eine Liste aller verifizierten Phishing-Webseiten kostenlos in verschiedenen Formaten zum Download bereit und bietet zusätzlich für den programmatischen Zugriff ein Webservice an. Für diese Arbeit wurde die Blacklist im JSON-Format heruntergeladen und unter dem Namen `blacklist.json` unter den Ressourcen der Dropwizard-Anwendung abgelegt. Jeder Eintrag dieser Blacklist umfasst verschiedene Informationen über die jeweilige Phishing-Webseite, wie z.B. die ID des Eintrages, die URL, den Einreichungszeitpunkt, den Verifizierungszeitpunkt, usw. Da im Analyseprozess am Server jedoch nur die URLs aller Phishing-Webseiten benötigten werden, werden auch lediglich nur diese zusammen mit deren IDs während des Starts der Anwendung aus dieser Datei extrahiert und in eine Liste von `BlacklistEntry`-Objekten gespeichert. Für das Lesen der Datei und die Extraktion der erforderlichen Daten wird die von Jackson zur Verfügung gestellte Streaming API verwendet, mit deren Hilfe eine JSON-Datei Token für Token geparst werden kann. Der große Vorteil dieser API liegt darin, dass hierzu nicht der gesamte Inhalt der Datei in den Speicher geladen werden muss, wodurch beliebig große Dateien verarbeitet werden können.

Dieser Analysetask überprüft schließlich, ob die URL der zu analysierenden Webseite in dieser Liste von `BlacklistEntry`-Objekten erscheint oder nicht. Um hierbei Fehler beim Abgleich aufgrund von verschiedenen Groß- und Kleinschreibweisen zu vermeiden, wird beim Abgleich der URL-Zeichenketten die Schreibweise (Groß- und Kleinbuchstaben) ignoriert. Sollte die URL in der Blacklist erscheinen, liefert der Task einen Score von 1,0 als Ergebnis, anderenfalls einen Score von 0,0.

Whitelist-Abgleich: Neben einer Blacklist verfügt der Server auch über eine Whitelist mit sämtlichen Webseiten, welche definitiv legitim sind. Dazu wurde eine eigene JSON-Datei namens `whitelist.json` unter den Ressourcen der Anwendung abgelegt. Jeder Eintrag umfasst hierbei ebenfalls eine eindeutige ID sowie die URL der jeweiligen legitimen Seite. Da die Einträge dieser Liste in späterer Folge jeder Benutzer individuell für sich selbst festlegen können sollte, wird hierbei auf einen Bezug dieser Liste von einer externen Quelle verzichtet. Im Zuge der Entwicklung des Prototyps wurde ebenfalls auf eine umfangreiche Befüllung dieser Whitelist verzichtet. Diese wurde lediglich beispielhaft mit ein paar wenigen legitimen Webseiten zur Überprüfung der Funktionalität befüllt. Wie auch bei der Blacklist, werden während des Starts der Anwendung alle Einträge der Whitelist mittels der Jackson Streaming API eingelesen und in eine Liste von `WhitelistEntry`-Objekten gespeichert.

In diesem Analysetask wird nun überprüft, ob die URL der zu analysierenden Webseite in dieser Liste von `WhitelistEntry`-Objekten erscheint oder nicht (auch hier wird die Schreibweise ignoriert). Sollte dies der Fall sein, retourniert der Task einen Score von 1,0, anderenfalls einen Score von 0,0.

Domänenalter [XHRC11]: Dieser Task überprüft das Alter der zur URL dazugehörigen Domäne, da viele Phishing-Webseiten typischerweise sehr kurzlebig sind und oft auf erst kürzlich registrierten Domänen gehostet werden. Zur Feststellung des Registrierungszeit-

[4] `https://www.phishtank.com/`

punktes werden WHOIS-Abfragen verwendet. Dazu wird zunächst von der Domäne die Subdomäne `www` entfernt (falls vorhanden) und anschließend der noch verbleibende Teil mit Hilfe der Methode `String.split("\\.")` in dessen Einzeldomänen unterteilt. Sollten hierbei mehr als zwei Einzeldomänen vorhanden sein, wird die erste WHOIS-Abfrage wie folgt zusammengesetzt:

(1) `whois <third-level-domain>.<second-level-domain>.<top-level-domain>`

Da das Format der Rückgabe von Server zu Server unterschiedlich sein kann, wird anhand der Schlüsselwörter „creation date:“, „created:“, „registration date:“, „registered:“, „registered on:“ und „activated:“ versucht, das Datum der Registrierung zu parsen und anschließend mittels unterschiedlichen Datumsformaten in ein `Date`-Objekt zu speichern. Sollte dies gelingen, wird zusammen mit dem aktuellen Datum das Alter der Domäne in Monaten berechnet. Wenn dieses einen zuvor definierten Schwellenwert (Berechnung dieses Wertes siehe Kapitel 7.1) übersteigt, liefert dieser Task einen Score von 1,0 und anderenfalls einen Score von 0,0. Wenn hingegen der Registrierungszeitpunkt mittels der ersten WHOIS-Abfrage nicht ermittelt werden konnte, wird eine zweite Abfrage der folgenden Form durchgeführt:

(2) `whois <second-level-domain>.<top-level-domain>`

Wenn auch hiermit der Zeitpunkt der Registrierung nicht ermittelt werden kann, liefert der Task ebenfalls einen Score von 1,0 und somit ein positives Ergebnis. Im Falle, dass nach der Unterteilung des noch verbleibenden Teiles nur zwei Einzeldomänen vorhanden sind, wird nur die WHOIS-Abfrage (2) durchgeführt. Der Hintergrund für die Durchführung von zwei Abfragen bei mehr als zwei Einzeldomänen ist folgender: Typischerweise betreiben nur Top-Level-Domains WHOIS-Server. Vereinzelt können jedoch auch Second-Level-Domains über eigene WHOIS-Server verfügen, wie es z.B. bei der Domäne `co.uk` der Fall ist [Who]. Würde beispielsweise bei der Domäne `www.test.co.uk` nur die WHOIS-Abfrage (2) durchgeführt werden, würde der Registrierungszeitpunkt von `co.uk` ermittelt werden, welcher in diesem Fall jedoch für alle Subdomains gleich ist.

Des Weiteren ist zu erwähnen, dass Phishing-Webseiten, welche auf (möglicherweise kompromittierten) legitimen Domänen gehostet sind, von dieser Heuristik nicht erkannt bzw. berücksichtigt werden.

IP-Adresse [XHRC11]: Dieser Analysetask stellt fest, ob es sich beim Domänennamen der URL um eine IP-Adresse handelt oder nicht. Um eine Phishing-Webseite zu tarnen bzw. keinen Verdacht aufgrund der URL zu erwecken, verwenden Angreifer hierzu oft eine IP-Adresse anstelle des Domänennamens. Bei legitimen Webseiten ist dies jedoch nur sehr selten der Fall, da hierbei typischerweise zum Zwecke einer einfachen Einprägsamkeit Domänennamen bevorzugt werden. Für die Überprüfung des Vorhandenseins einer IPv4- oder IPv6-Adresse selbst wurde die `InetAddressValidator`-Klasse des Common Apache Validator Frameworks[5] verwendet. Sollte das Ergebnis der Überprüfung positiv sein, liefert der Task einen Score von 1,0 und anderenfalls einen Score von 0,0.

Verdächtige URL [XHRC11]: Dieser Task überprüft, ob in der URL ein At-Zeichen (@) oder im Domänennamen ein Bindestrich (-) vorhanden ist. Sollte ein At-Zeichen enthalten

[5] `http://commons.apache.org/proper/commons-validator/`

sein, bleibt die gesamte Zeichenkette davor (links vom At-Zeichen) unberücksichtigt und die Zeichenkette danach (rechts vom At-Zeichen) wird als tatsächliche URL für die Abfrage der Webseite herangezogen. Da die Adressleiste eines Browser nur eine beschränkte Größe aufweist und lange URLs somit oft nicht gänzlich für den Benutzer ersichtlich sind, können mit dieser Methode URLs erstellt werden, dessen vorderer (in der Adressleiste sichtbarer) Teil zwar legitim erscheint, aber dessen hinterer (nicht in der Adressleiste sichtbarer) Teil aufgrund des Vorhandenseins eines At-Zeichens den Browser tatsächlich zum Aufrufen einer anderen Webseite veranlasst. Sollte hingegen ein Bindestrich enthalten sein, kann dies ebenfalls ein Hinweis auf eine verdächtige Seite sein, da diese in URLs von legitimen Webseiten typischerweise eher selten vorkommen. Wenn das Ergebnis einer dieser beiden Überprüfungen positiv ist, wird ein Score von 1,0 und anderenfalls ein Score von 0,0 retourniert.

Anzahl der Punkte [XHRC11]: Diese Heuristik analysiert die Anzahl der in der URL vorhandenen Punkte. Hierbei neigen URLs von Phishing-Webseiten zu einer höheren Anzahl im Gegensatz zu legitimen Seiten. Um eine höhere Genauigkeit zu erreichen, wird nicht die Gesamtanzahl an Punkten, sondern die Anzahl an „zusätzlichen“ Punkten bewertet. Dazu wird zunächst die Gesamtanzahl p_{total} ermittelt und anschließend unterschieden, ob die URL die Subdomain `www` beinhaltet oder nicht. Sollte dies der Fall sein, sind zumindest zwei Punkte üblich (ein Punkt nach der Subdomain `www` und ein Punkt vor der Top-Level-Domain) und die Anzahl an zusätzlichen Punkten ergibt sich aus $p_{add} = p_{total} - 2$. Sollte die URL jedoch die Subdomain `www` nicht beinhalten, ist nur jener Punkt vor der Top-Level-Domain üblich und es gilt $p_{add} = p_{total} - 1$. Wenn die Anzahl der zusätzlichen Punkte einen bestimmten Schwellenwert (Berechnung siehe Kapitel 7.1) überschreitet, liefert dieser Task einen Score von 1,0 und ansonsten einen Score von 0,0.

Anzahl der Satzzeichen [BBVV+17]: Dieser Task untersucht die Interpunktion in der URL, da hierbei Phishing- im Gegensatz zu legitimen Seiten typischerweise ebenfalls eine höhere Anzahl aufweisen. Dazu wird zuerst die Gesamtanzahl an Vorkommnissen der in einer URL zulässigen Satzzeichen . ! # $ % & , ; ’ gezählt und diese anschließend mit einem bestimmten Schwellenwert (Berechnung siehe ebenfalls Kapitel 7.1) verglichen. Bei einer Überschreitung liefert dieser Task einen Score von 1,0 und anderenfalls einen Score von 0,0.

Analyse der Metadaten

Analysetasks dieser Kategorie analysieren anhand der nachfolgenden Kriterien die Metadaten der Webseite.

Verdächtige Meta-Tags: Diese Heuristik gleicht den Inhalt von sämtlichen `<meta>`-Tags mit einer internen Liste von verdächtigen Meta-Inhalten ab. Bei dieser Inhaltsliste handelt es sich um eine JSON-Datei namens `meta_content.json`, welche unter den Ressourcen der Dropwizard-Anwendung abgelegt ist und ein Array von Einträgen mit der folgenden Struktur beinhaltet:

```
{"id":<id>,"content":<meta-content>,"confidence":<confidence>}
```

Mit Hilfe der `id` kann ein Eintrag eindeutig identifiziert werden, `content` beinhaltet den verdächtigen Inhalt (entweder Klartext oder Attribut-Wert) und `confidence` gibt die Konfidenz (bzw. Zuverlässigkeit) des Eintrages an. Bei dieser Konfidenz handelt es sich um ein

Maß für die Vertraulichkeit, dass es sich bei der zu analysierenden Webseite tatsächlich um eine Phishing-Webseite handelt, wenn der jeweilige Inhalt in den Meta-Tags vorkommt. Beispielsweise kann der Klartext „email“ zwar ein Indikator für die Abfrage der E-Mail-Adresse auf einer Phishing-Webseite sein, jedoch kommt er auf zahlreichen legitimen Webseiten ebenfalls vor und hat dementsprechend eine niedrigere Konfidenz. Etwas dubiosere Inhalte, wie z.B. „mswebdialog“ oder „pc9hpjwvbgk“, stellen hingegen einen signifikanteren Hinweis auf eine Phishing-Webseite dar und sind dementsprechend mit einer höheren Konfidenz versehen. Wie diese verdächtigen Inhalte bestimmt werden und dementsprechend die Liste befüllt wird, ist in Kapitel 5.4.3 näher erläutert. Sämtliche Einträge dieser Inhaltsliste werden während des Starts der Anwendung mit Hilfe der Jackson Streaming API aus dieser JSON-Datei gelesen und in eine Liste von `MetalistEntry`-Objekten gespeichert.

Dieser Task extrahiert nun zunächst alle `<meta>`-Tags aus dem HTML-Code der Webseite und stellt anhand der zuvor beschriebenen Liste fest, ob verdächtige Inhalte darin vorkommen (Groß- und Kleinschreibung werden beim Abgleich nicht berücksichtigt). Sollte dies der Fall sein, wird die höchste Konfidenz aller gefundenen Inhalte als Score zurückgegeben. Konnte hingegen kein verdächtiger Inhalt festgestellt werden, liefert der Task einen Score von 0,0.

Wurden z.B. die folgenden verdächtigen Inhalte mit den in den Klammern angegebenen Konfidenzen gefunden, retourniert der Task einen Score von 1,0: „email“ (0,71), „payment“ (0,92), „pc9hpjwvbgk“ (1,0).

Inhalt des Title-Tags [K.Vi10]: Der Inhalt des `<title>`-Tags einer Webseite wird dem Benutzer unter anderem in der Titelleiste des Browserfensters bzw. des jeweiligen Tabs angezeigt. Angreifer verwenden hierzu den Namen der nachgeahmten Webseite, um so den Benutzer in den Glauben zu versetzen, dass er eine legitime Seite besuche. Diese Heuristik überprüft nun, ob zumindest ein Wort aus dem Inhalt des `<title>`-Tags auch in der Domäne der jeweiligen Webseite vorkommt. Dazu wird der `<title>`-Tag aus dem HTML-Code extrahiert und dessen Inhalt mittels der Methode `String.split("[\\W_]+")` in einzelne Wörter unterteilt. Anschließend wird wiederum ohne Berücksichtigung der Schreibweise überprüft, ob zumindest eines dieser Wörter auch in der Domäne der URL der Webseite vorkommt oder nicht. Sollte dies der Fall sein, liefert der Task einen Score von 0,0 und anderenfalls (auch wenn der Titel nicht vorhanden oder leer ist) einen Score von 1,0.

Analyse der Struktur

Diese Kategorie von Analyseschritten untersucht die Struktur des HTML-Codes und stellt das Vorhandensein von bestimmten Features bzw. Merkmalen fest, welche nachfolgend aufgelistet sind.

Verdächtige Formulare [XHRC11]: HTML-Formulare bilden die Basis von vielen Phishing-Angriffen, da sie als Eingabemöglichkeit für sensible und persönliche Informationen herangezogen werden können. Anhand deren konkreter Implementierung gibt es unterschiedliche Hinweise, welche auf ein potentiell schädliches bzw. verdächtiges Phishing-Formular hinweisen können. Diese Heuristik stellt nun das Vorhandensein von derart verdächtigen Formularen auf der zu analysierenden Webseite fest und liefert genau dann einen Score von 1,0 und somit ein positives Ergebnis, wenn alle der folgenden Bedingungen erfüllt sind:

1. Die Webseite verfügt über ein HTML-Formular, gekennzeichnet mittels eines `<form>` -Tags.
2. Im Scope des Formulars befindet sich mindestens ein Dateneingabefeld in Form eines `<input>`-Tags.
3. Im Scope des Formulars befinden sich entweder verdächtige Keywords bzw. Phrasen, die auf eine Aufforderung zur Eingabe von sensiblen Informationen oder auf eine Phishing-Webseite hinweisen (wie z.B. „password“ oder „credentials“), oder der gesamte Scope beinhaltet überhaupt keinen Text sondern ausschließlich nur Bilder in Form von `<img>`-Tags.
4. Keine Verwendung des HTTPS-Protokolls in der URL des Action-Feldes (`action`-Attribut des `<form>`-Tags) wenn eine absolute URL angegeben ist, oder keine Verwendung des HTTPS-Protokolls in der URL der zu analysierenden Webseite wenn eine relative URL angegeben ist oder das Action-Feld leer ist.

Zur Überprüfung des Vorhandenseins von verdächtigen Keywords und Phrasen im Scope des Formulars wird eine Liste namens `form_content.json` mit insgesamt 71 suspekten Inhalten (Array von Strings) herangezogen, wobei auch hier beim Abgleich zwischen Groß- und Kleinbuchstaben nicht unterschieden wird. Auf die Bestimmung bzw. Feststellung dieser Inhalte wird in Kapitel 5.4.3 näher eingegangen.

Um eine derart text-basierte Erkennung von Formularen zu vermeiden bzw. zu umgehen, ersetzen Angreifer häufig sämtlichen Text durch entsprechende Textbilder. Auch diese Art von HTML-Formularen werden von dieser Heuristik als verdächtig eingestuft. Wenn kein verdächtiges Formular gefunden wurde, liefert der Task einen Score von 0,0 und somit ein negatives Ergebnis.

Eingabefelder: Diese Heuristik überprüft, ob die Webseite generell über Dateneingabemöglichkeiten in Form von `<input>`-Tags verfügt (unabhängig ob im Scope eines Formulars oder nicht). Einige bereits vorhandene Lösungen (wie z.B. [TCWS16] und [SrPa17]) nutzen diese Heuristik als Vorfilter zur Erkennung einer Phishing-Webseite und stufen eine Seite vor der eigentlichen Analyse bereits als harmlos ein, wenn sie über keine Eingabefelder verfügt (da in diesem Fall keine Daten auf der Seite eingegeben und somit „gephisht“ werden können). Bei der in dieser Arbeit vorgeschlagenen Lösung wird jedoch nur der statische HTML-Code der Webseite vom REST-Server geladen und analysiert. Im Gegensatz zum Laden einer Webseite mit Hilfe eines Browsers bzw. einer Browser-Engine wird dadurch sämtlicher JavaScript-Code vor der Analyse nicht ausgeführt und interpretiert. Da die Struktur der Seite durch die Ausführung von JavaScript-Code noch geändert werden kann, kann auch ein dynamisches Hinzufügen von Eingabefeldern während des Ladevorganges nicht ausgeschlossen werden. Aufgrund dessen ist diese Heuristik als Vorfilter in dieser Anwendung nicht sinnvoll, da ansonsten jene Phishing-Webseiten im Vorhinein schon als harmlos eingestuft werden würden, bei welchen erst beim Laden der Seite dynamisch Eingabefelder eingebunden werden. Deswegen wurde diese Heuristik als Analysetask und nicht als Vorfilter implementiert. Beim Vorhandensein von zumindest einem `<input>`-Tag liefert der Task einen Score von 1,0 und ansonsten 0,0.

Verdächtige Frames: Diese Heuristik untersucht die vorhandenen Frames der Webseite. Wie bereits in Kapitel 3.3.2 beschrieben, können dadurch entweder schadhafte Inhalte oder sogar legitime Seiten zur Gänze in Phishing-Webseiten eingebettet werden. Um dies erkennen zu können, werden alle `<iframe>`- und `<frame>`-Tags aus dem HTML-Code extrahiert und überprüft, ob die Domäne der im `src`-Attribut angegebenen URL in Beziehung mit der Domäne der aktuellen Webseite steht oder nicht. Dies ist genau dann der Fall, wenn entweder die beiden Domänen ident sind, oder wenn es sich bei einer Domäne um eine Subdomäne der anderen handelt. Zur Feststellung dieses Umstandes wird mit Hilfe der Methode `String.endsWith()` überprüft, ob eine der beiden Domänen jeweils mit der anderen endet oder nicht. Sollte ein derart verdächtiger Frame gefunden werden, liefert der Task einen Score von 1,0 und 0,0 anderenfalls.

Keine Links im Body [RaAl15]: Diese Heuristik überprüft, ob die zu analysierende Webseite über zumindest einen `<input>`-Tag, aber keinen Link im HTML-Body verfügt. Legitime Webseiten, welche die Eingabe von Benutzerinformationen (wie z.B. Anmeldedaten und dergleichen) fordern, verfügen typischerweise über einen Link zum Registrieren, einen Link im Falle des Vergessens des Passwortes oder ähnliche Links. Bei Phishing-Webseiten werden jedoch häufig Seiteninhalte durch entsprechende Bilder ersetzt und somit beinhaltet der HTML-Body zwar `<input>`-Tags, aber keinen einzigen Link. Sollte dies der Fall sein, liefert diese Heuristik einen Score von 1,0 und anderenfalls einen Score von 0,0.

Leere Links [RaAl15]: Dieser Analysetask überprüft, ob die zu analysierende Webseite leere Links beinhaltet. Ein Link wird hierbei als leer bezeichnet, wenn er auf die aktuelle Seite „weiterleitet“. Viele Phishing-Webseiten verfügen zwar optisch über alle Links der nachgeahmten (legitimen) Seite, jedoch manipulieren die Angreifer das tatsächliche Verweisziel (`href`-Attribut) damit das Opfer immer auf der selben (schadhaften) Webseite verweilt. Dieser Task extrahiert nun alle Links in Form von `<a>`-Tags aus dem HTML-Code und zählt die Anzahl aller, bei welchen der Inhalt des `href`-Attributs entweder

- ein leerer String ist (`<a href="">`),
- mit einem Hash startet (z.B. `<a href="#">` oder `<a href="#skip">`)
- oder mit Hilfe von JavaScript zu einem leeren Link manipuliert wird (z.B. `<a href="javascript:;">` oder `<a href="javascript:void(0);">`).

Anschließend wird das Verhältnis der leeren Links zur Gesamtanzahl aller auf der Webseite vorhandenen Links berechnet und wenn dieses einen bestimmten Schwellenwert (Berechnung siehe Kapitel 7.1) überschreitet, liefert der Task einen Score von 1,0 und ansonsten einen Score von 0,0.

Verdächtige Links [ZhHC07]: Dieser Analysetask führt die gleiche Überprüfung, wie auch der Task zur Bestimmung einer verdächtigen URL (siehe oben), auf allen Links der Webseite durch. Dementsprechend liefert er einen Score von 1,0, wenn zumindest ein Link entweder über ein At-Zeichen (@) in der gesamten URL verfügt oder der Domänenname der URL einen Bindestrich (-) beinhaltet.

Nicht übereinstimmende URLs [FeST07]: Diese Heuristik überprüft, ob die zu analysierende Webseite Links mit nicht übereinstimmenden URLs beinhaltet. Darunter versteht man `<a>`-Tags, bei welchen der angezeigte Text eine URL ist, deren Domäne jedoch nicht

mit der im `href`-Attribut vorhandenen Domäne übereinstimmt. Dadurch können dem Benutzer Links auf scheinbar legitime Webseiten angezeigt werden, welche tatsächlich jedoch auf Phishing-Seiten verlinken. Ein Beispiel eines derartigen Links sieht wie folgt aus:

```
<a href="http://www.myphishingsite.com">http://www.raiffeisen.at</a>.
```

Wenn auch nur ein verdächtiger Link vorhanden ist, liefert der Task einen Score von 1,0 und anderenfalls einen Score von 0,0.

Nur Script: Dieser Task analysiert, ob der Body der Webseite komplett leer ist, aber `<script>` -Tags vorhanden sind. Einige Phishing-Seiten verfügen über keinen HTML-Inhalt, sondern lediglich über JavaScript-Code, welcher beispielsweise den Benutzer auf eine andere (schadhafte) Webseite umleitet oder anderweitige Schadbefehle ausführt. Bei legitimen Seiten ist hingegen der Body typischerweise mit Inhalten befüllt und nicht komplett leer. Eine positive Überprüfung dieses Umstandes ergibt einen Score von 1,0 und eine negative einen Score von 0,0.

Verdächtiger JavaScript-Code: Diese Heuristik stellt wiederum das Vorhandensein von verdächtigen JavaScript-Codeteilen bzw. -Befehlen fest. Dazu verfügt die Dropwizard-Anwendung über eine interne JSON-Datei namens `script_content.json`, welche ein Array von Einträgen mit der folgenden Struktur beinhaltet:

```
{"id":<id>,"content":<content>,"confidence":<confidence>}
```

Auch bei dieser Datei dient das Feld `id` der eindeutigen Identifizierung eines Eintrages, `content` beinhaltet den jeweiligen JavaScript-Codeteil bzw. -Befehl und `confidence` gibt wiederum die Konfidenz (bzw. Zuverlässigkeit) des Eintrages an. Der Inhalt dieser Datei wird ebenfalls während des Starts der Anwendung mit Hilfe der Jackson Streaming API gelesen und in eine Liste von entsprechenden `ScriptlistEntry`-Objekten gespeichert. Die Bestimmung bzw. Feststellung der Inhalte dieser Liste wird in Kapitel 5.4.3 erläutert.

Dieser Analysetask extrahiert zunächst sämtliche `<script>`-Tags aus dem HTML-Code der Webseite und stellt mit Hilfe dieser Liste fest, ob verdächtige JavaScript-Codeteile bzw. -Befehle darin vorkommen. Dazu werden alle Leerzeichen aus dem gesamten Inhalt der einzelnen Tags entfernt und der dadurch erhaltene String einfach mittels der beiden Trennzeichen „=“ (Gleichheitszeichen) und „;“ (Strichpunkt) in einzelne Codeteile zerlegt, welche anschließend für den Abgleich mit der Liste herangezogen werden (auch hier wird wiederum die Groß- und Kleinschreibung nicht beachtet). Durch diese Herangehensweise kann das Nichterkennen von verdächtigen Inhalten aufgrund von unterschiedlichen Schreibweisen sowie unterschiedlicher Setzung von Leerzeichen (wie z.B. `location.href=` und `location.href␣=`) vermieden werden.

Sollte die Webseite über verdächtige Codeteile verfügen, wird wiederum die höchste Konfidenz aller gefundenen Einträge als Score zurückgegeben. Konnte hingegen kein Treffer erzielt werden, liefert der Task einen Score von 0,0.

Identität der Webseite [RaAl15]: Dieser Task bestimmt zunächst auf Basis der vorhandenen Links die Identität der Webseite. Bei legitimen Seiten ist die Häufigkeit von Links, welche auf die eigene Domäne verweisen, im Vergleich zur Häufigkeit von Links, welche auf eine fremde Domäne verweisen, typischerweise sehr hoch. Da Angreifer das Verhalten der abgezielten legitimen Webseite zu imitieren versuchen, fügen sie oft in die Phishing-

Webseite Links ein, welche auf die nachgeahmte Seite verweisen. Auf diesem Umstand basiert die Berechnung der Identität einer Webseite. Dazu wird von allen vorhandenen Links (`<a>`-Tags) der Domänenname der URL aus dem `href`-Attribut extrahiert und dessen Häufigkeit berechnet. Jene Domäne, welche die größte Häufigkeit aufweist, wird als Identität der Webseite herangezogen und mit der tatsächlichen Domäne der Webseite verglichen. Sollten diese nicht in Beziehung miteinander stehen (wenn sie nicht ident sind und auch keine der beiden Domänen eine Subdomäne der jeweils anderen ist), wird die zu analysierende Webseite als Phishing-Seite, welche auf die Domäne mit der größten Häufigkeit (Identität der Webseite) abzielt, in Betracht gezogen und der Task liefert einen Score von 1,0. Wenn die Identität jedoch mit der Domäne der Webseite in Beziehung steht, ist der Score des Tasks 0,0.

Analyse des Inhaltes

Analysetasks dieser Kategorie untersuchen ausschließlich nur den vorhandenen Klartext der Webseite.

Der Prototyp umfasst dazu nur einen einzigen Analysetask, welcher das Vorhandensein von verdächtigen Inhalten feststellt. Auch dazu verfügt der Server über eine eigene Inhaltsliste in Form von einer JSON-Datei namens `plaintext_content.json`. Diese umfasst ein Array von Einträgen mit der folgenden Struktur:

```
{"id":<id>,"text":<keyword>,"lang":<language>,"confidence":<confidence>}
```

Wie auch bei allen anderen Inhaltslisten dient das Feld `id` der eindeutigen Identifizierung eines Eintrages, `text` beinhaltet den verdächtigen Klartext (Keyword), `lang` gibt die dazugehörige Sprache an und `confidence` stellt wiederum die Konfidenz (bzw. Zuverlässigkeit) des Eintrages dar. Der Inhalt des `lang`-Feldes wird derzeit noch nicht berücksichtigt, wurde jedoch für eine zukünftige Erweiterung des Prototyps schon miteinbezogen. Der Inhalt dieser Datei wird ebenfalls während des Starts der Anwendung mit Hilfe der Jackson Streaming API gelesen und in eine Liste von entsprechenden `ContentlistEntry`-Objekten gespeichert. Die Bestimmung bzw. Feststellung der Inhalte dieser Liste wird in Kapitel 5.4.3 erläutert.

Der Task extrahiert zunächst den gesamten Klartext aus dem HTML-Code und überprüft anschließend mit Hilfe der zuvor beschriebenen Inhaltsliste, ob dieser verdächtige Keywords enthält (Groß- und Kleinschreibung werden beim Abgleich wiederum nicht berücksichtigt). Sollte dies der Fall sein, liefert der Task die höchste Konfidenz aller gefundenen Einträge der Liste als Score zurück. Beinhaltet der Klartext hingegen keine verdächtigen Inhalte, liefert der Task einen Score von 0,0.

5.4.3 Analysetask-Parameter und Inhaltslisten

Jeder der im vorherigem Abschnitt beschriebenen Analysetasks (bis auf die beiden Tasks für den Blacklist- und Whitelist-Abgleich) besitzt bestimmte Parameter, welche zur Berechnung des schlussendlichen Gesamtergebnisses der Analyse benötigt werden. Diese Parameter und die einzelnen Inhaltslisten wurden auf Basis eines Trainingskorpus von insgesamt 8.000 legitimen und 8.000 Phishing-Webseiten berechnet bzw. bestimmt (nähere Details siehe Kapitel 7). Alle Parameter der einzelnen Tasks sind in einer JSON-Konfigurationsdatei namens `config.json` abgelegt und werden nachfolgend beschrieben.

Konfidenz, Grenzwert, Sensitivität und Spezifität

Die Konfidenz (engl. Confidence) eines Tasks ist eine reelle Zahl zwischen 0 und 1 ($Konfidenz \in [0, 1]$) und gibt dessen Zuverlässigkeit bzw. Vertrauensgrad an. Dabei handelt es sich um die Erfolgsrate des jeweiligen Analysetasks in Bezug auf den gesamten Trainingskorpus. Zur Berechnung dieser Erfolgsrate ist zusätzlich ein Grenzwert (engl. Threshold) notwendig. Dieser gibt an, bis zu welchem Score aus Sicht des jeweiligen Tasks die zu analysierende Webseite als legitim und ab welchem Score als Phishing-Webseite eingestuft wird. Dieser Grenzwert ist eine reelle Zahl zwischen 0 und 1 ($Grenzwert \in [0, 1]$) und kennzeichnet jenen Schwellwert, ab welchem der Task positiv anschlägt. Der trivialste Ansatz wäre, diesen immer auf 0,0 zu setzen, jedoch wird in diesem Fall die „Stärke“ des Anschlags nicht berücksichtigt. Aus diesem Grund wurde der Grenzwert zusammen mit der Konfidenz wie in Algorithmus 1 ersichtlich berechnet. Des Weiteren bestimmt dieser Algorithmus für den gefundenen Grenzwert und der dazugehörigen Konfidenz auch die entsprechende Sensitivität und Spezifität des Tasks. Die Sensitivität (auch Richtig-Positiv-Rate genannt) gibt den Anteil aller korrekt als Phishing eingestuften Webseiten unter allen Phishing-Seiten an. Die Spezifität (auch Richtig-Negativ-Rate genannt) gibt wiederum den Anteil aller korrekt als legitim eingestuften Webseiten unter allen legitimen Seiten an. Diese beiden Werte können wie folgt berechnet werden:

$$Sensitivität = \frac{RP}{RP + FN} \tag{5.1}$$

$$Spezifität = \frac{RN}{RN + FP} \tag{5.2}$$

wobei RP die Anzahl aller richtig positiv eingestuften Phishing-Seiten, FN die Anzahl aller falsch negativ eingestuften Phishing-Seiten, RN die Anzahl aller richtig negativ eingestuften legitimen Seiten und FP die Anzahl aller falsch positiv eingestuften legitimen Seiten darstellt.

Algorithmus 1 berechnet zunächst für jeden möglichen Grenzwert die dazugehörige Konfidenz, Sensitivität sowie Spezifität. Dazu wird in jedem inneren Schleifendurchlauf über alle Webseiten des Trainingskorpus iteriert und mitgezählt, wie oft der Task beim derzeitigen Grenzwert richtig liegt. Dies ist genau dann der Fall, wenn

- entweder die derzeitige Webseite legitim ist und der Task einen Score kleiner gleich dem derzeitigen Grenzwert liefert
- oder die derzeitige Webseite eine Phishing-Seite ist und der Task einen Score größer als dem derzeitigen Grenzwert liefert.

Mit Hilfe der ermittelten Anzahl an richtig eingestuften Webseiten und der Gesamtanzahl an erfolgreich analysierten Seiten wird die Konfidenz berechnet, mit der Anzahl an richtig positiv eingestuften Phishing-Seiten und der Gesamtanzahl an Phishing-Seiten die Sensitivität und mit der Anzahl an richtig negativ eingestuften legitimen Seiten und der Gesamtanzahl an legitimen Seiten die Spezifität. Der derzeitige Grenzwert wird zusammen mit den berechneten Werten in eine Hashtabelle gespeichert. Als Ergebnis liefert der Algorithmus schlussendlich jenen Eintrag der Hashtabelle (bestehend aus Grenzwert und dazugehöriger Konfidenz, Sensitivität und Spezifität), bei welchem die Konfidenz am höchsten ist. Sollten hierbei mehrere Einträge in Fra-

Algorithmus 1 Berechnung Konfidenz, Grenzwert, Sensitivität und Spezifität eines Tasks

```
1: procedure calculateConfidence(corpus)
2:     Let H be an empty hash map
3:     for currentThreshold = 0.00 to 0.99,  + 0.01 do
4:         correct ← 0
5:         total ← 0
6:         truePositive ← 0
7:         trueNegative ← 0
8:         numberOfPhishs ← 0
9:         numberOfLegitimates ← 0
10:        for webpage : corpus do
11:            if score ≠ −1.0 then
12:                total + +
13:                if webpage == legitimate then
14:                    numberOfLegitimates + +
15:                else
16:                    numberOfPhishs + +
17:                end if
18:                if webpage == legitimate && taskScore ≤ currentThreshold then
19:                    correct + +
20:                    trueNegative + +
21:                else if webpage == phishing && taskScore > currentThreshold then
22:                    correct + +
23:                    truePositive + +
24:                end if
25:            end if
26:        end for
27:        currentConfidence ← (correct/total)
28:        sensitivity ← (truePositive/numberOfPhishs)
29:        specificity ← (trueNegative/numberOfLegitimates)
30:        H.put(currentThreshold, [currentConfidence, sensitivity, specificity])
31:    end for
32:    return entry with max confidence of H
33: end procedure
```

ge kommen (weil sie alle die höchste Konfidenz besitzen), wird der Eintrag mit dem kleinsten Grenzwert herangezogen.

Sicherheitsgrenzen

Neben der Konfidenz, Sensitivität, Spezifität und des Grenzwertes besitzt jeder Task zusätzlich eine untere Sicherheitsgrenze (engl. Lower Safety Limit) und eine obere Sicherheitsgrenze (engl. Upper Safety Limit), bei welchen es sich ebenfalls um reelle Zahlen zwischen 0 und 1 handelt ($Sicherheitsgrenzen \in [0, 1]$). Mit Hilfe dieser Grenzen kann bereits anhand des Scores des Tasks eine vorzeitige Entscheidung bezüglich der Legitimität der zu analysierenden Webseite getroffen werden, wenn eine der beiden folgenden Bedingungen erfüllt ist:

- $Score > obere\ Sicherheitsgrenze \Rightarrow Phishing - Webseite$
- $Score < untere\ Sicherheitsgrenze \Rightarrow legitime\ Webseite$

Bei einer exakten Feststellung bzw. Berechnung dieser Sicherheitsgrenzen muss sichergestellt werden, dass auf Basis des Trainingskorpus keine legitime Webseite existiert, bei welcher der Task einen Score größer als dessen obere Sicherheitsgrenze liefert bzw. dass keine Phishing-Seite existiert, bei welcher der Task einen Score kleiner als dessen untere Sicherheitsgrenze liefert. Aufgrund der Tatsache, dass die einzelnen Tasks keine eindeutigen Indikatoren für eine legitime bzw. eine Phishing-Webseite darstellen, sondern erst durch deren Kombination diesbezüglich eine akkurate Aussage getroffen werden kann, können diese dementsprechend auch häufig falsch anschlagen. Damit dennoch Sicherheitsgrenzen gefunden werden können, wurde zu deren Berechnung zusätzlich eine Genauigkeit (engl. Accuracy) und somit eine maximal zulässige Fehlerrate eingeführt. Abbildung 5.2 veranschaulicht diesen Grundgedanken.

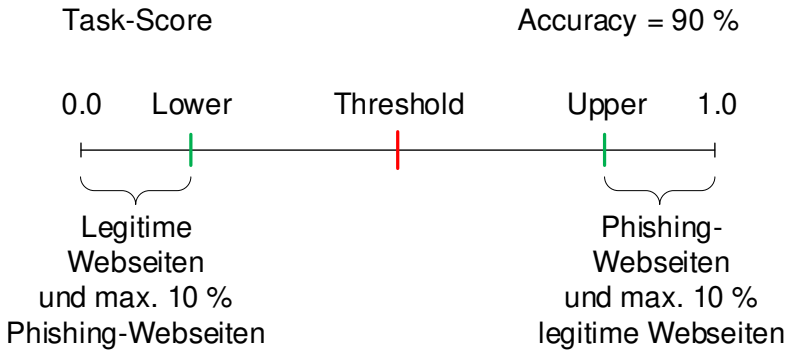

Abb. 5.2: Sicherheitsgrenzen bei einer Genauigkeit von 90 %

Werden die Sicherheitsgrenzen beispielsweise mit einer Genauigkeit von 90 % (wie in Abbildung 5.2 dargestellt) berechnet, bedeutet dies dementsprechend Folgendes:

- Unter allen Webseiten des Trainingskorpus, bei welchen der Task einen Score größer als die obere Sicherheitsgrenze (in Abbildung 5.2 mit „Upper“ bezeichnet) liefert, befinden sich nur Phishing-Webseiten und maximal 10 % legitime Seiten.
- Unter allen Webseiten des Trainingskorpus, bei welchen der Task einen Score kleiner als die untere Sicherheitsgrenze (in Abbildung 5.2 mit „Lower“ bezeichnet) liefert, befinden sich nur legitime Webseiten und maximal 10 % Phishing-Seiten.

Zur Bestimmung bzw. Berechnung der oberen Sicherheitsgrenze wurde Algorithmus 2 entwickelt. Dieser startet mit einer oberen Sicherheitsgrenze von 0,00 und iteriert in jedem Schleifendurchlauf über den gesamten Korpus. Im Zuge dessen wird einerseits die Anzahl aller Webseiten gezählt, bei welchen der Score des Tasks größer als die derzeitige Sicherheitsgrenze liegt (*total*) und zusätzlich noch, wie viele dieser Webseiten vom Task falsch eingestuft wurden (*errors*). Zur Beurteilung der Korrektheit des Tasks wird der im vorherigen Abschnitt beschriebene Grenzwert (*threshold*) herangezogen. Am Ende jedes Durchlaufs wird mit Hilfe von *total* und *errors* die Fehlerrate berechnet und sollte diese in Abhängigkeit der Genauigkeit im erlaubten Bereich sein, handelt es sich um eine zulässige obere Sicherheitsgrenze. Wenn hingegen in allen Durchläufen keine derartige Grenze gefunden wurde, liefert der Algorithmus ein Ergebnis von 1,0.

Die untere Sicherheitsgrenze wird nach dem gleichen Prinzip wie die obere berechnet. Der Pseudocode dieser Berechnung ist in Algorithmus 3 ersichtlich. Da wie in Abbildung 5.2 dargestellt $lower \leq upper$ gilt, startet die äußere Schleife hierzu bei der zuvor berechneten oberen Sicher-

Algorithmus 2 Berechnung der oberen Sicherheitsgrenze

```
1: procedure calculateUpperLimit(corpus, accuracy)
2:     for currentUpper = 0.00 to 0.99,  + 0.01 do
3:         errors ← 0
4:         total ← 0
5:         for webpage : corpus do
6:             if score ≠ −1.0 then
7:                 if taskScore > currentUpper then
8:                     total + +
9:                     if webpage == legitimate && taskScore > threshold then
10:                        errors + +
11:                    else if webpage == phishing && taskScore ≤ threshold then
12:                        errors + +
13:                    end if
14:                end if
15:            end if
16:        end for
17:        if (errors/total) ≤ (1.0 − accuracy) then
18:            return currentUpper
19:        end if
20:    end for
21:    return 1.0
22: end procedure
```

heitsgrenze und verringert diese in jedem Durchlauf um 0,01. Innerhalb der äußeren Schleife wird ebenfalls für jede mögliche untere Sicherheitsgrenze über den gesamten Korpus iteriert und die entsprechende Fehlerrate ermittelt. Sollte diese sich wiederum in Abhängigkeit der Genauigkeit im erlaubten Bereich befinden, wurde eine zulässige untere Sicherheitsgrenze gefunden. Anderenfalls liefert der Algorithmus ein Ergebnis von 0,0.

Inhaltslisten

Neben den verschiedenen Berechnungen der einzelnen Task-Parameter, wird der Trainingskorpus an legitimen und Phishing-Webseiten auch für die Befüllung der unterschiedlichen Inhaltslisten herangezogen. Nachfolgend wird das Prozedere der Extraktion der verschiedenen Inhalte kurz beschrieben.

Liste der verdächtigen Meta-Inhalte: Für die Extraktion aller verdächtigen Meta-Inhalte wurde die Klasse `MetaContentExtractor` implementiert. Diese iteriert im ersten Schritt über alle Phishing-Webseiten des Korpus und erstellt im Zuge dessen ein Wörterbuch von verdächtigen Inhalten. Dazu werden für jede Seite alle `<meta>`-Tags extrahiert und anschließend deren Inhalt mit Hilfe des regulären Ausdrucks `[\W_]+` (entspricht allen nichtalphanumerischen Zeichen inklusive dem Unterstrich) anhand seiner Wortgrenzen in einzelne Wörter bzw. Substrings unterteilt. Diese werden in einer Hashmap gespeichert (wobei im Vorhinein alle Groß- zu Kleinbuchstaben umgewandelt werden, um die Schreibweise zu ignorieren) und mitgezählt, wie oft jeder einzelne Eintrag in allen `<meta>`-Tags aller Phishing-Webseiten vorkommt. Das Ergebnis des ersten Schrittes ist somit ein Wörterbuch bestehend aus allen enthaltenen Wörtern bzw. Substrings inklusive deren Anzahl an

Algorithmus 3 Berechnung der unteren Sicherheitsgrenze

```
1: procedure calculateLowerLimit(corpus, upperLimit, accuracy)
2:     for currentLower = upperLimit to 0.01, − 0.01 do
3:         errors ← 0
4:         total ← 0
5:         for webpage : corpus do
6:             if score ≠ −1.0 then
7:                 if taskScore < currentLower then
8:                     total + +
9:                     if webpage == legitimate && taskScore > threshold then
10:                        errors + +
11:                    else if webpage == phishing && taskScore ≤ threshold then
12:                        errors + +
13:                    end if
14:                end if
15:            end if
16:        end for
17:        if (errors/total) ≤ (1.0 − accuracy) then
18:            return currentLower
19:        end if
20:    end for
21:    return 0.0
22: end procedure
```

Vorkommnissen c_p. Um ein möglichst für alle Phishing-Webseiten repräsentatives Wörterbuch zu erhalten (und nicht nur für den hierzu verwendeten Korpus), werden im zweiten Schritt alle Einträge mit einer niedrigen Anzahl ($c_p < 1$ % aller analysierten Webseiten – bei 8.000 Seiten $c_p < 80$) aus der Hashmap entfernt. Im dritten Schritt wird über alle legitimen Seiten des Korpus iteriert und mit Hilfe des Wörterbuches bzw. der Hashmap gezählt, wie oft jedes dieser Wörter bzw. jeder dieser Substrings in den `<meta>`-Tags aller legitimen Seiten vorkommt und der daraus resultierende Zähler c_l ebenfalls in der Hashmap zu dem jeweiligen Eintrag gespeichert. Das Ergebnis des dritten Schrittes ist somit eine Hashmap bestehend aus allen verdächtigen Inhalten inklusive deren Anzahl an Vorkommnissen auf allen Phishing- (c_p) und deren Anzahl an Vorkommnissen auf allen legitimen Seiten (c_l). Mit Hilfe dieser Zähler wird die Konfidenz jedes Eintrages wie folgt berechnet:

$$confidence = \frac{c_p}{c_p + c_l} \tag{5.3}$$

Um die Genauigkeit des dazugehörigen Tasks (Verdächtige Meta-Tags) zu erhöhen, werden abschließend alle Einträge mit $confidence \geq 0,7$ in die Inhaltsliste aller verdächtigen Meta-Inhalte (`meta_content.json`) aufgenommen.

Liste der verdächtigen Formular-Inhalte: Zur Extraktion aller verdächtigen Formular-Inhalte dient die Klasse `FormContentExtractor`. Das Prozedere ist hierbei bis auf wenige Unterschiede nahezu ident wie bei der zuvor beschriebenen Meta-Inhaltsliste. Zunächst werden wiederum alle `<form>`-Tags aus allen Phishing-Webseiten extrahiert. Da sich im

Unterschied zu einem <meta>-Tag innerhalb eines <form>-Tags diverse andere Tags befinden können, wird nicht der gesamte Inhalt mittels des regulären Ausdrucks `[\W_]+` in einzelne Substrings unterteilt, sondern nur sämtliche Attributwerte aller Tags sowie der gesamte Klartext innerhalb des <form>-Tags. Damit wird vermieden, dass Tag-Namen und Attributnamen im Wörterbuch erscheinen, welche fälschlicherweise auf die Eingabe von sensiblen Information hindeuten können (wie z.B. das Attribut `allowpaymentrequest` eines <iframe>-Tags, oder das `id`-Attribut diverser anderer Tags). Nachdem alle aus den Phishing-Seiten extrahierten Inhalte gezählt wurden, werden alle Einträge mit einer Anzahl $c_p < 2\,\%$ aller analysierten Webseiten (bei 8.000 Seiten $c_p < 160$) aus dem Wörterbuch entfernt. Im Gegensatz zur Meta-Inhaltsliste wurden hier 2 % und nicht 1 % gewählt, da mit dieser Vorgehensweise bessere Ergebnisse erzielt werden konnten. Nachdem die Anzahl an Vorkommnissen aller Einträge auf allen legitimen Seiten (c_l) bestimmt wurde, wird anschließend ebenfalls gleichermaßen die Konfidenz berechnet. Aus der dadurch entstehenden Liste wurden schließlich manuell alle Einträge mit $confidence \geq 0,7$, welche auf die Eingabe von sensitiven Informationen hinweisen, in die Inhaltsliste aller verdächtigen Formular-Inhalte (`form_content.json`) aufgenommen.

Liste der verdächtigen Script-Inhalte: Für die Extraktion der verdächtigen Script-Inhalte wurde die Klasse `ScriptContentExtractor` implementiert. Die Vorgehensweise ist hierbei bis auf einen kleinen Unterschied bei der Extraktion der Inhalte ident mit jener der Meta-Inhaltsliste. Wie schon beim dazugehörigen Task (Verdächtiger JavaScript-Code) werden auch hier bei der Extraktion zunächst alle Leerzeichen aus dem gesamten Inhalt der einzelnen <script>-Tags entfernt und anschließend der gesamte String mittels des regulären Ausdruckes `[=;]+` in einzelne Codeteile zerlegt. Ansonsten ist das Prozedere und die Befüllung der Inhaltsliste aller verdächtigen Script-Inhalte (`script_content.json`) gleich.

Liste der verdächtigen Klartext-Inhalte: Für die Befüllung der Klartext-Inhalte dient die Klasse `PlaintextContentExtractor`. Auch hier ist die Vorgehensweise nahezu ident mit jener zur Befüllung der Meta-Inhaltsliste. Der einzige Unterschied besteht in der Erstellung des Wörterbuches. Hierzu wird der gesamte Klartext extrahiert und ebenfalls mittels des regulären Ausdrucks `[\W_]+` in einzelne Wörter bzw. Substrings unterteilt. Damit hier jedoch wirklich nur Klartextwörter und nicht Zahlenkombinationen, Adressen oder Ähnliches in das Wörterbuch mitaufgenommen werden, wird dies zuvor mit Hilfe des regulären Ausdrucks `[a-zA-Z]+` sichergestellt. Ansonsten bleibt auch hier das restliche Prozedere und damit die Befüllung der Inhaltsliste aller verdächtigen Klartext-Inhalte (`plaintext_content.json`) gleich.

5.4.4 Ablauf

In diesem Abschnitt wird der konkrete Ablauf des Analyseprozesses am REST-Server näher erläutert. Wie in Abbildung 5.3 ersichtlich, kann dieser grob in insgesamt vier verschiedene Schritte eingeteilt werden. In diesen Schritten wird die Webseite anhand der zuvor beschriebenen Analysetasks untersucht und auf Basis der Einzelergebnisse ($Score \in [0,1]$) ein Gesamtergebnis ($finalScore \in [0,1]$) berechnet, welches schlussendlich bei einer erfolgreichen Analyse die Seite wie in Kapitel 4 beschrieben in ein grünes, gelbes oder rotes Ampelsymbol einstuft. Sollte es jedoch im Zuge der Analyse am Server zu einem unerwarteten Fehler kommen, endet

der Analyseprozess mit einem Gesamtergebnis von -1,0. Für diesen Umstand wurde zusätzlich ein graues Ampelsymbol eingeführt.

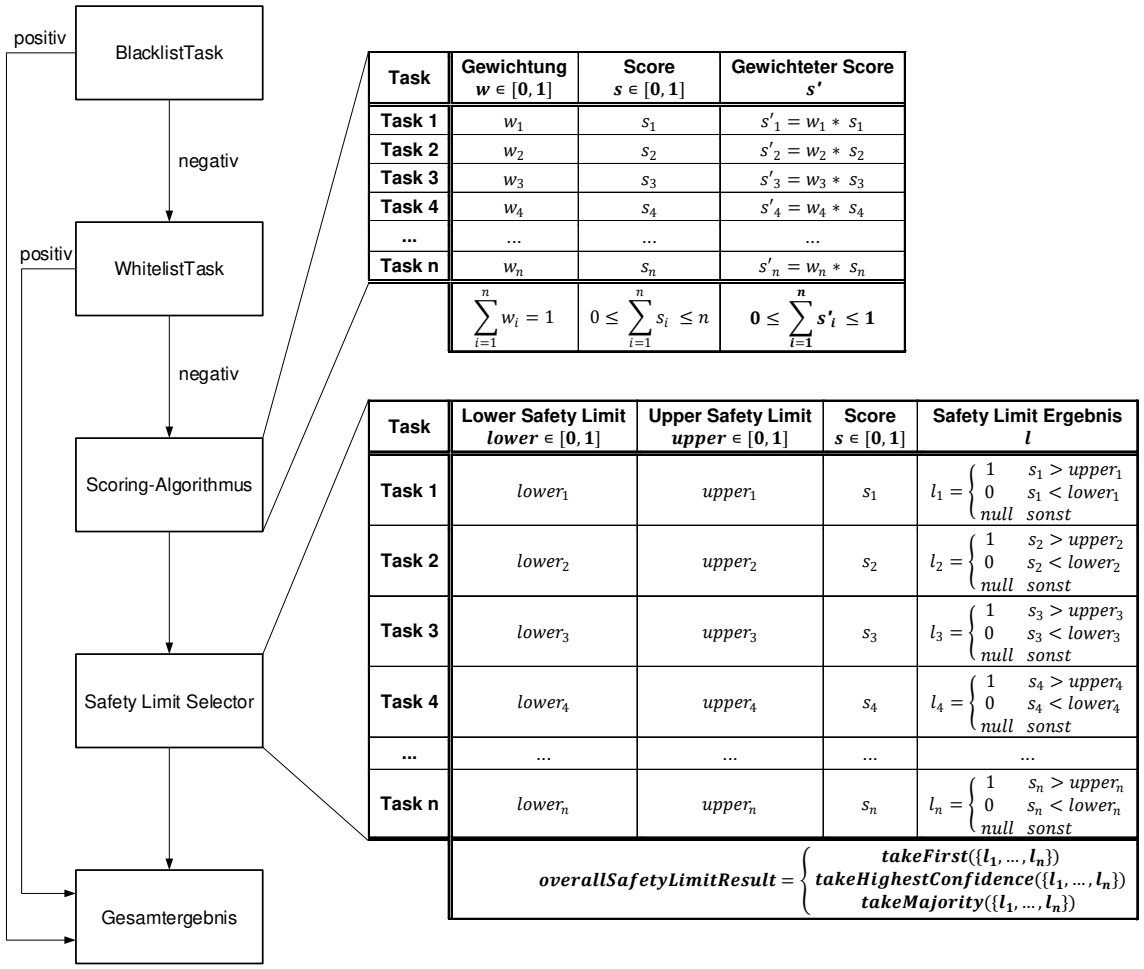

Abb. 5.3: Konkreter Ablauf des Analyseprozesses am Server

Am Beginn wird zunächst immer der Blacklist-Abgleich durchgeführt. Wenn dieser Task bereits ein positives Ergebnis liefert (Score von 1,0) und somit die URL der zu analysierenden Webseite in der internen Blacklist erscheint, wird die Seite definitiv als Phishing-Webseite angesehen und der Analyseprozess endet mit einem Gesamtergebnis von 1,0. Liefert dieser Task hingegen ein negatives Ergebnis (einen Score von 0,0) wird anschließend der Whitelist-Abgleich durchgeführt.

Wenn der Task für den Whitelist-Abgleich hingegen anschlägt (Score von 1,0) und sich die URL somit in der internen Whitelist befindet, wird die zu analysierende Webseite an dieser Stelle als definitiv legitim eingestuft und die Analyse endet mit einem Gesamtergebnis von 0,0. Anderenfalls startet anschließend der Scoring-Algorithmus des Analyseprozesses.

Scoring-Algorithmus

Im Zuge des Scoring-Algorithmus werden alle noch verbleibenden Analysetasks (bis auf die beiden Tasks für den Blacklist- und Whitelist-Abgleich) in einem Thread Pool parallel mit Hilfe des Java Executor Frameworks ausgeführt und deren retournierten Ergebnisse in eine Liste von

`TaskResult`-Objekten gespeichert. Sobald alle Tasks ihre jeweiligen Analyseaufgaben abgearbeitet haben und somit die Liste aller Task-Ergebnisse vollständig ist, wird das Gesamtergebnis am Server berechnet. Dabei müssen alle Einzelergebnisse möglichst optimal kombiniert und auf einen reellen Wert zwischen 0 und 1 ($finalScore \in [0, 1]$) abgebildet werden, um schließlich eine Aussage bezüglich der Legitimität der Webseite treffen zu können (grünes, gelbes oder rotes Ampelsymbol). Dabei handelt es sich um ein typisches Klassifizierungsproblem, für welches in der Literatur bereits zahlreiche verschiedene Algorithmen und Ansätze existieren. Aus Gründen der Einfachheit wurde ein lineares Modell der folgenden Form verwendet (siehe obere Tabelle in Abbildung 5.3, vgl. [ZhHC07]):

$$finalScore = \sum_{i=1}^{n} w_i * s_i \tag{5.4}$$

Wobei es sich bei w_i um die Gewichtung und bei s_i um den Score eines Tasks handelt. Für die Berechnung der Gewichtungen werden die vorhin beschriebenen Task-Parameter herangezogen, welche während des Starts der Anwendung aus der JSON-Konfigurationsdatei (`config.json`) geladen werden. Insgesamt wurden dazu zwei unterschiedliche Ansätze implementiert, welche nachfolgend erläutert werden.

Normalisierung der Konfidenz: Bei diesem Modell basiert die Berechnung der Gewichtung auf der Konfidenz der einzelnen Tasks. Dabei wird die normalisierte Konfidenz eines Tasks als dessen Gewichtung w_i herangezogen:

$$w_i = \frac{c_i}{\sum_{j=1}^{n} c_j} \tag{5.5}$$

Normalisierung des Effekts: Der große Nachteil des vorherigen Gewichtungsmodells liegt in der Tatsache, dass bei dieser Art der Berechnung nicht berücksichtigt wird, wie effektiv der jeweilige Task in der Erkennung einer Phishing-Seite ist. Beispielweise kann sich eine Konfidenz von 50 % durch eine Richtig-Positiv-Rate (Sensitivität) von 0 % und eine Falsch-Positiv-Rate (ergibt sich aus 1− Spezifität) von 100 % oder umgekehrt ergeben. In beiden Fällen hätte der jeweilige Task bei der obigen Methode die gleiche Gewichtung, obwohl im ersten Fall der Task nichts zur Erkennung einer Phishing-Webseite beitragen würde. Aus diesem Grund wurde ein zweiter Algorithmus, welcher auch in der Literatur häufig wiederzufinden ist (wie z.B. in [ZhHC07]), als Vergleich implementiert. Dieser berechnet die Gewichtung w_i basierend auf dem Effekt e_i, welcher sich aus der Sensitivität und der Spezifität eines Tasks berechnen lässt:

$$\begin{aligned} e_i &= f(sensitivity_i - (1 - specifity_i)) \\ &= f(truePositiveRate_i - falseNegativeRate_i) \end{aligned} \tag{5.6}$$

wobei f eine einfache Schwellenwertfunktion der folgenden Form ist:

$$f(x) = \begin{cases} f(x) & x > 0 \\ 0 & \text{sonst} \end{cases} \tag{5.7}$$

$$w_i = \frac{e_i}{\sum_{j=1}^{n} e_j} \tag{5.8}$$

Mit Hilfe der Gewichtung und des retournierten Scores eines Tasks wird dessen gewichteter Score berechnet ($s'_i = w_i * s_i$). Jene Tasks, welche jedoch aufgrund eines Fehlers einen Score von -1,0 liefern, werden in diesem Schritt aussortiert und somit im weiteren Verlauf nicht mehr berücksichtigt. Das vorläufige Gesamtergebnis ergibt sich durch die Summe aller gewichteten Scores und entspricht, wie bereits erwähnt, einer reellen Zahl zwischen 0 und 1.

Safety Limit Selector

Der Safety Limit Selector stellt den letzten Schritt der Berechnung dar. Hierbei werden die Scores der einzelnen Tasks mit deren Sicherheitsgrenzen (untere Tabelle in Abbildung 5.3) abgeglichen. Wenn der Score hierbei größer als die untere Sicherheitsgrenze ist, ergibt dies ein Safety Limit Ergebnis von $l_i = 1$. Ist er hingegen kleiner als die untere Grenze, ergibt dies ein Ergebnis von $l_i = 0$. Liegt er hingegen zwischen den beiden Grenzen, ist das Ergebnis des Abgleichs *null* (Tasks mit Score -1,0 werden hierbei nicht mehr berücksichtigt).

Sobald auch nur ein Safety Limit Ergebnis ungleich *null* ist, kommt es definitiv zu einer Überschreibung des vorläufigen Gesamtergebnisses. Zur Auswahl des in diesem Fall herangezogenen Ergebnisses wurden insgesamt drei verschiedene Methoden implementiert, welche nachfolgend kurz beschrieben werden.

$takeFirst(l_1, ..., l_n)$: Bei dieser Methode wird über alle Task-Ergebnisse iteriert und deren Safety Limit Ergebnisse bestimmt. Sobald hierbei das erste Ergebnis ungleich *null* gefunden wurde, wird dieses unabhängig von allen anderen herangezogen und zurückgeliefert.

$takeHighestConfidence(l_1, ..., l_n)$: Diese Methode iteriert ebenfalls über alle Task-Ergebnisse und ermittelt dessen Safety Limit Ergebnisse. Am Ende wird jenes Ergebnis herangezogen, dessen dazugehöriger Task die höchste Konfidenz besitzt. Wenn hierzu zwei unterschiedliche Safety Limit Ergebnisse in Frage kommen (eines mit Wert 1 und eines mit Wert 0), da die dazugehörigen Tasks jeweils die gleiche (höchste) Konfidenz besitzen, wird die Webseite als Phishing-Webseite angesehen und 1 retourniert.

$takeMajority(l_1, ..., l_n)$: Ähnlich wie bei der vorherigen Methode wird auch hier über alle Task-Ergebnisse iteriert und die dazugehörigen Safety Limit Ergebnisse berechnet. Anschließend werden alle mit Wert 1 und alle mit Wert 0 gezählt und jenes mit der größeren Anzahl zurückgeliefert. Wenn die Anzahl hierbei gleich ist, wird wiederum jenes herangezogen, dessen dazugehöriger Task die höchste Konfidenz besitzt. Sollten auch hierbei mehrere in Frage kommen, wird die Webseite ebenfalls als Phishing-Webseite eingestuft und ein Ergebnis von 1 zurückgeliefert.

Unabhängig welche Methode verwendet wird, liefert der Safety Limit Selector schlussendlich eines der folgenden Ergebnisse:

- 0: Bei mindestens einem Task lag der Score nicht innerhalb seiner Sicherheitsgrenzen und das vorläufige Gesamtergebnis wird mit 0,0 überschrieben und somit die Webseite definitiv als legitim eingestuft.
- 1: Bei mindestens einem Task lag der Score nicht innerhalb seiner Sicherheitsgrenzen und das vorläufige Gesamtergebnis wird mit 1,0 überschrieben und somit die Webseite definitiv als Phishing-Webseite eingestuft.
- *null*: Bei allen Tasks lag der Score innerhalb seiner Sicherheitsgrenzen und es kommt zu keiner Überschreibung des vorläufigen Gesamtergebnisses. In diesem Fall wird das vorläufige Gesamtergebnis als schlussendliches Gesamtergebnis herangezogen.

5.4.5 Ranges

Zuletzt muss das Gesamtergebnis ($finalScore \in [0, 1]$) noch einem entsprechenden Ampelsymbol (grün, gelb oder rot) zugeordnet werden. Die dazu notwendigen Ranges bzw. Intervalle wurden ebenfalls aus dem in Kapitel 5.4.3 beschriebenen Trainingskorpus berechnet und in die JSON-Konfigurationsdatei `config.json` gespeichert. Dafür wurde eine eigene Klasse namens `RangesCalculator` implementiert, welche zunächst alle Gesamtergebnisse ($finalScores$) der im Korpus vorhandenen Webseiten berechnet (wobei die beiden Tasks für den Blacklist- und Whitelist-Abgleich nicht in die Analyse miteinbezogen wurden) und in eine sortierte Liste ($= finalScoresList$) speichert. Die Gesamtanzahl der in dieser Liste vorhandenen Einträge wird anschließend gedrittelt ($= interval$) und die Ranges bzw. Intervalle für die einzelnen Ampelsymbole wie folgt bestimmt:

- Grün: $[0,\ finalScoresList[interval])$
- Gelb: $[finalScoresList[interval],\ finalScoresList[2 * interval])$
- Rot: $[finalScoresList[2 * interval],\ 1.0]$

Durch diese Vorgehensweise werden die Ranges basierend auf den Häufigkeiten aller tatsächlichen Gesamtergebnisse aller Webseiten des Korpus bestimmt. Je nach dem, in welches Intervall das Gesamtergebnis einer Analyse fällt, wird es schlussendlich dem entsprechenden Ampelsymbol zugeordnet.

6 Implementierung Client

Dieses Kapitel beschreibt die Implementierung des clientseitigen Browser-Plugins, bei welchem es sich um eine Erweiterung für den Mozilla Firefox handelt. Dazu stellt Mozilla die WebExtensions APIs [Web] zur Verfügung, welche ein Cross-Browser-System zur Entwicklung von Browsererweiterungen für den Firefox darstellen. Mit Hilfe einer derartigen Erweiterung können die Funktionen eines Web-Browsers unter der Verwendung der Standard-Webtechnologien JavaScript, HTML und CSS sowie ein paar bestimmten JavaScript APIs erweitert und modifiziert werden.

6.1 Aufbau einer Firefox-Erweiterung

Eine Erweiterung für Firefox besteht aus mehreren verschiedenen Dateien, welche für die Verteilung und Installation zu einem Package verpackt werden. Die wichtigsten Dateien sind in Abbildung 6.1 veranschaulicht und werden nachfolgend kurz erläutert.

manifest.json: Die JSON-formatierte Datei `manifest.json` ist die einzige Datei, die jede mittels der WebExtension APIs entwickelte Erweiterung zwingend beinhalten muss. Sie enthält einerseits grundlegende Metadaten über die Erweiterung (wie etwa den Namen, die Version, den Autor oder welche Berechtigungen benötigt werden) und andererseits Pointers auf weitere notwendige Dateien der Erweiterung (wie z.B. Background Scripts, Content Scripts oder Browser Actions).

Background Pages: Background Scripts enthalten jenen Code, welcher unabhängig von der Lebensdauer einer bestimmten Webseite oder eines bestimmten Browserfensters im Hintergrund ausgeführt wird und somit der Abarbeitung von lang andauernden Operationen dient. Sie werden gleichzeitig mit der Erweiterung geladen und bleiben bis zu deren Deaktivierung oder Deinstallation erhalten. Ein Background Script wird dabei nicht im Kontext der aktuellen Webseite, sondern im Kontext einer speziellen Seite, auch Background Page genannt, ausgeführt. Dadurch hat es ebenfalls auf ein globales `window`-Objekt und dessen bereitgestellten Document Object Model (DOM) APIs Zugriff. Des Weiteren können innerhalb eines Background Scripts alle WebExtension APIs verwendet (solange die Erweiterung die erforderlichen Berechtigungen besitzt), Daten dynamisch und domänenübergreifend von einer beliebigen URL mittels eines XMLHttpRequests (XHR-Request) abgerufen sowie auf Klick-Events einer Browser Action reagiert werden. Außerdem hat ein Background Script keinen direkten Zugriff auf Webseiten, kann aber dennoch durch das Laden von Content Scripts über diese indirekt darauf zugreifen.

Content Scripts: Ein Content Script wird wiederum direkt in eine Webseite geladen und wird dementsprechend auch im Kontext dieser bestimmten Seite ausgeführt. Es ermöglicht

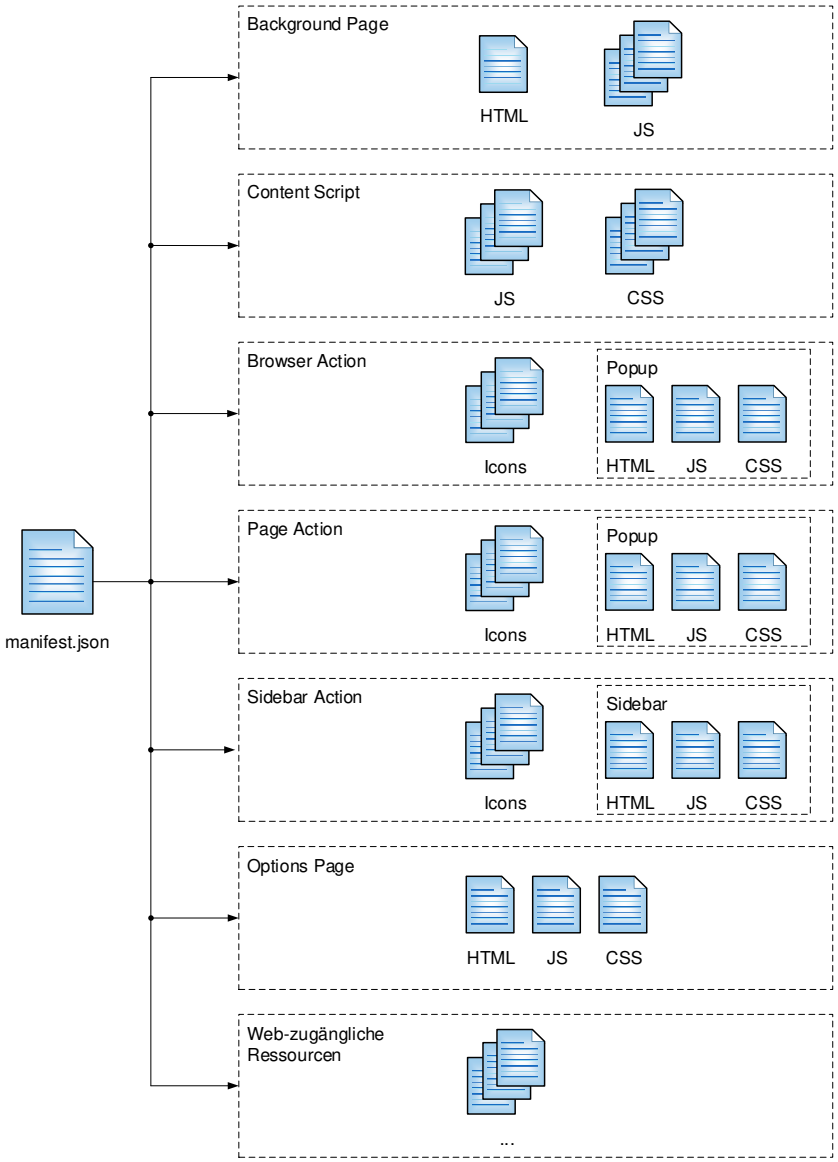

Abb. 6.1: Aufbau einer Erweiterung (vgl. [Web])

somit den Zugriff und die Manipulation von Webseiten, unterscheidet sich jedoch von gewöhnlichen Scripts, welche direkt von einer Webseite selbst geladen werden (wie z.B. jene, die mittels `<script>`-Tags definiert sind). Wie auch gewöhnliche Scripts haben sie Zugriff auf das Document Object Model (DOM), können aber im Gegensatz zu diesen zusätzlich domänenübergreifende XHR-Requests ausführen, auf bestimmte WebExtensions APIs zugreifen und Nachrichten mit Background Scripts austauschen.

Browser Action Dateien: Mit Hilfe einer Browser Action kann ein Button zur Toolbar des Browsers hinzugefügt werden. Dieser Button ist anschließend in allen Tabs des Browsers in der Toolbar sichtbar und kommt grundsätzlich dann zum Einsatz, wenn dessen Funktionalität auf allen Webseiten anwendbar sein soll. Eine Browser Action wird immer in der Datei `manifest.json` spezifiziert, wobei es hier zwei verschiedene Möglichkeiten gibt: entweder mit einem Popup oder ohne. Ein Popup wird dabei mittels HTML, CSS und

JavaScript erstellt und unterscheidet sich somit nicht von einer normalen Webseite. Sämtlicher darin laufende JavaScript-Code hat, wie auch ein Background Script, Zugriff auf alle WebExtension APIs, wird jedoch nicht im Kontext der aktuellen Webseite, sondern im Kontext des Popups ausgeführt. Sollte bei der Spezifikation der Browser Action nun ein Popup angegeben werden, wird dieses bei Klick auf den Button angezeigt und dessen Inhalt geladen. Der Benutzer kann nun anschließend mit dem Popup interagieren und es schließt automatisch, sobald außerhalb des erscheinenden Fensters geklickt wird. Sollte bei der Spezifikation jedoch kein Popup angegeben werden, wird bei Klick auf den Button ein Event ausgelöst, welches wiederum in der Erweiterung abgefangen und entsprechend darauf reagiert werden kann.

Page Action Dateien: Ähnlich wie ein Toolbar Button, kann mit Hilfe einer Page Action ein Button zur Adressleiste des Browsers hinzugefügt werden. Standardmäßig ist dieser Button in allen Tabs des Browsers unsichtbar und kann bei Bedarf per JavaScript ein- und ausgeblendet werden. Ein Button in der Adressleiste unterscheidet sich grundsätzlich nicht von einem Button in der Toolbar, kommt jedoch im Gegensatz dazu dann zum Einsatz, wenn dessen Funktionalität nur auf bestimmten Webseiten anwendbar sein soll.

Options Pages: Mit Hilfe einer Options Page können Einstellungen, welche jeder Benutzer für sich individuell vornehmen kann, definiert und angezeigt werden. Diese Seite kann mittels HTML, CSS und JavaScript definiert und entweder über den Addons-Manager des Browsers oder programmatisch in der Erweiterung per JavaScript aufgerufen werden. Der darin ausgeführte JavaScript-Code hat ebenfalls Zugriff auf sämtliche WebExtension APIs und kann insbesondere die Storage API zur persistenten Speicherung der vornehmbaren Einstellungen verwenden.

Sidebar Action Dateien: Eine Sidebar ist eine Leiste, welche am linken Rand des Browsers angezeigt wird. Hierzu stellt jeder Browser ein User Interface für die Anzeige und Auswahl sämtlicher verfügbaren Sidebars zur Verfügung. In einer Erweiterung kann mit Hilfe einer Sidebar Action eine eigene Sidebar zum Browser hinzugefügt werden. Wie auch bei einem Browser Action oder Page Action Popup, wird der Inhalt der Sidebar ebenfalls mittels eines HTML-Dokumentes definiert. Sobald der Benutzer die Seitenleiste öffnet, wird dieses Dokument jeweils als eine eigene Instanz in alle offenen Browserfenster geladen. Sidebar-Dokumente haben dabei Zugriff auf die selben JavaScript APIs wie auch Background und Popup Scripts, haben ebenfalls Zugriff auf Background Pages und können auch mit Content Scripts interagieren. Eine Instanz eines Sidebar-Dokumentes wird gelöscht, sobald das dazugehörige Browserfenster oder die gesamte Sidebar geschlossen wird.

Web-zugängliche Ressourcen: Web-zugängliche Ressourcen sind Ressourcen (wie z.B. Bilder, HTML, CSS oder JavaScript), welche in der Erweiterung beinhaltet und gleichzeitig mittels eines speziellen URI-Schemas über das Web zugänglich sind. Wenn beispielsweise mit Hilfe eines Content Scripts Bilder zur aktuellen Webseite hinzugefügt werden sollen, dann können diese Bilder in die Erweiterung gepackt und web-zugänglich gemacht werden. Dadurch können anschließend in einem Content Script `<img>`-Tags definiert und mittels des `src`-Attributes unter Verwendung des URI-Schemas diese Bilder von einer anderen Webseite aus referenziert werden.

6.2 Implementierung der Erweiterung

In diesem Kapitel wird die Implementierung der Browser-Erweiterung unter der Verwendung der von Mozilla zur Verfügung gestellten WebExtensions APIs beschrieben. Dazu wird zuerst der Aufbau des Plugins kurz erläutert und anschließend die Anzeige eines Analyseergebnisses dargestellt.

6.2.1 Aufbau

Der Funktionsumfang bzw. die Komplexität der Browser-Erweiterung ist äußerst gering. Sie verfügt lediglich über eine Manifest-Datei namens `manifest.json`, ein Background Script namens `background.js` sowie ein Popup namens `info.html`.

Manifest-Datei: In der Manifest-Datei werden unter anderem die zur korrekten Funktionsweise der Erweiterung erforderlichen Berechtigungen definiert. Hierzu sind die Folgenden notwendig:

- `webRequest`: Diese Berechtigung erlaubt der Browser-Erweiterung die Verwendung der webRequest API. Mit Hilfe dieser kann auf HTTP-Anfragen zugegriffen und diese auch modifiziert und abgebrochen werden. Abhängig von der Phase bzw. des Fortschritts der Anfrage können die in Abbildung 6.2 ersichtlichen Events durch entsprechende Event-Listeners abgefangen werden.

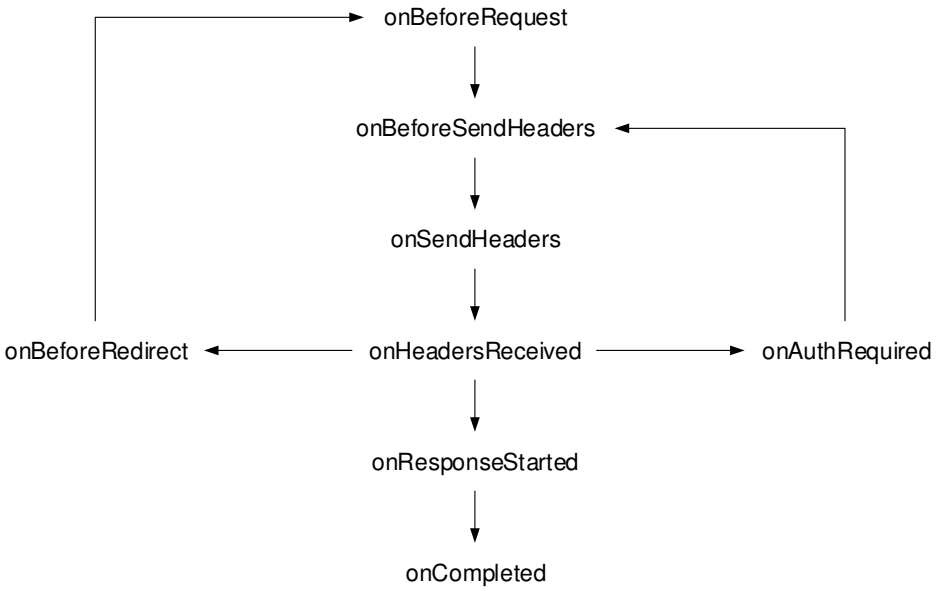

Abb. 6.2: Reihenfolge der Events einer HTTP-Anfrage (vgl. [Web])

Das Event `onBeforeRequest` wird ausgelöst, bevor die Anfrage ausgeführt wird und noch bevor die Header-Daten zur Verfügung stehen. Sobald alle HTTP-Header-Informationen vorhanden sind, wird `onBeforeSendHeaders` getriggert. Das Ereignis `onSendHeaders` wird wiederum unmittelbar bevor die Header-Daten gesendet werden ausgelöst und `onHeadersReceived` kennzeichnet, dass die Header-Informationen der zu einer Anfrage dazugehörigen Antwort empfangen wurden. Das Event `onAuthRequired`

gibt an, dass der Server Anmeldeinformationen zur Authentifizierung anfordert und `onBeforeRedirect`, dass der Server eine Umleitung initiiert hat. Zuletzt werden noch die beiden Ereignisse `onResponseStarted` ausgelöst, sobald das erste Byte der Antwort empfangen wurde und `onCompleted`, wenn die Anfrage abgeschlossen wurde. Des Weiteren kann während des gesamten HTTP-Requests bei Auftritt eines Fehlers jederzeit das Event `onErrorOccurred` ausgelöst werden.

- `tabs`: Dadurch wird die Verwendung der tabs API erlaubt, welche die Interaktion mit dem Tab-System des Browsers ermöglicht. Mit Hilfe dieser API können beispielsweise sämtliche Informationen über alle im Browser geöffneten Tabs abgerufen (wie z.B. der Tab-Titel, die Tab-URL oder der Tab-Inhalt) oder einzelne Tabs geöffnet, aktualisiert, verschoben, neu geladen oder gelöscht werden.
- `<all_urls>`: Um von bestimmten Hosts mit Hilfe der webRequest API Events empfangen zu können, muss die Erweiterung für diese Hosts zusätzliche Berechtigungen anfordern – sogenannte Host Permissions. Diese werden mit Hilfe von Match Pattern der Form `<scheme>://<host><path>` spezifiziert und geben jene URLs an, für die die Erweiterung diese zusätzlichen Berechtigungen anfordert. Bei `<all_urls>` handelt es sich um ein spezielles Pattern, welches repräsentativ für alle möglichen URLs ist.

Neben den Berechtigungen werden in der Manifest-Datei außerdem noch das Background Script (`background.js`) sowie ein Button in der Toolbar des Browsers in Form einer Browser Action definiert. Für den Button ist zusätzlich ein Popup (`info.html`) spezifiziert, welches bei Klick auf diesen Button erscheint und dem Benutzer im Browser-Fenster angezeigt wird.

Background Script: Das Background Script (`background.js`) beinhaltet den gesamten JavaScript-Code der Erweiterung und ist für die Kommunikation mit dem REST-Server zuständig. Hierzu wird mit Hilfe eines Listeners und der webRequest API für jede HTTP-Anfrage das Event `onBeforeRequest`, wie im folgenden Beispiel 6.1 ersichtlich, abgefangen.

```
browser.webRequest.onBeforeRequest.addListener(
  logURL,
  {urls: ["<all_urls>"], types: ["main_frame"]}
);
```

Src. 6.1: Abfangen des Events `onBeforeRequest`

Der erste Parameter gibt die Listener-Funktion an, welche beim Auftritt des Ereignisses aufgerufen wird. Der zweite Parameter kennzeichnet einen Filter, mit dessen Hilfe die Ereignisse, welche an diesen Listener gesendet werden, eingeschränkt werden können:

- `urls: ["<all_urls>"]`: Gibt an, dass der Listener nur für HTTP-Anfragen aufgerufen wird, deren Ziel-URLs den angegebenen Match Patterns entsprechen – in diesem Fall für alle URLs.
- `types: ["main_frame"]`: Gibt an, dass der Listener nur für Ressourcen (wie z.B. Stylesheets, Bilder, Scripts, usw.) aufgerufen wird, welche den angegebenen Typen entsprechen. `main_frame` kennzeichnet alle Top-Level-Dokumente und somit wird der Listener im Zuge des Ladens einer Webseite nur einmal aufgerufen und nicht für jede einzelne Ressource erneut.

Sobald das Ereignis abgefangen wurde, wird innerhalb der Methode `logURL()` die URL der angeforderten Webseite extrahiert und im Hintergrund asynchron ein `XMLHttpRequest` an den Server gesendet. Dieser beinhaltet im POST-Body die URL der zu analysierenden Webseite und triggert am REST-Server den Analyseprozess. Aus der dazugehörigen Antwort wird das Gesamtergebnis und das Ampelsymbol geparst und das Icon des Toolbar Buttons dementsprechend geändert. Da im Browser zur gleichen Zeit mehrere Tabs geöffnet sein können, wird zusätzlich für jeden Tab das derzeitige Analyseergebnis des Servers in einem Dictionary abgelegt. Wenn der Benutzer den derzeitigen Tab wechselt, wird das zu diesem Tab dazugehörige Ergebnis abgerufen (falls vorhanden) und wiederum das Icon des Toolbar Buttons entsprechend aktualisiert.

Popup: Dabei handelt es sich um ein kleines Popup (`info.html`), welches dem Benutzer bei Klick auf den Toolbar Button angezeigt wird. Dieses beinhaltet lediglich ein paar grundlegende Informationen über die Browser-Erweiterung (Name, Version, Icon, kurze Beschreibung und Autor).

6.2.2 Anzeige des Analyseergebnisses

Wenn der Benutzer im Browser eine neue URL aufruft und das dazugehörige Analyseergebnis des Servers am Client verfügbar ist, wird dieses in Form eines Ampelsymbols rechts in der Toolbar, wie in Abbildung 6.3 ersichtlich, dargestellt.

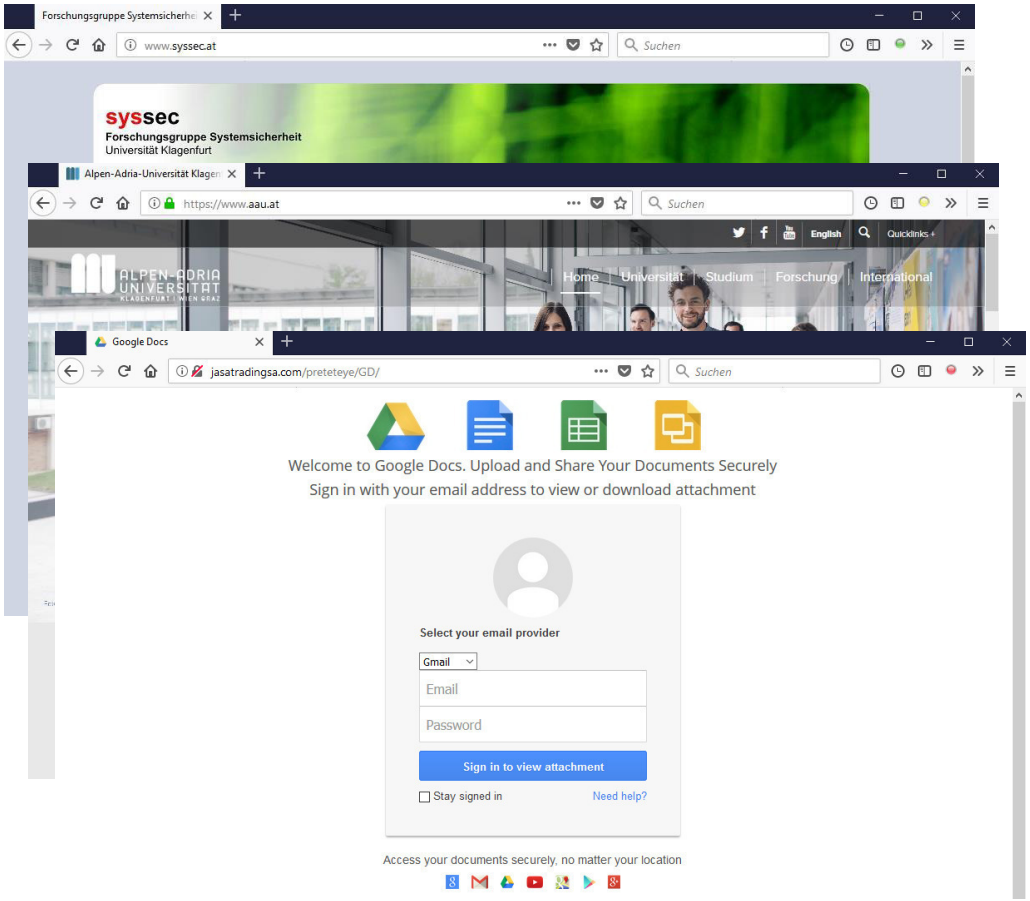

Abb. 6.3: Anzeige des Analyseergebnisses im Browser

7 Evaluierung und Ergebnisse

In diesem Kapitel wird die Evaluierung der einzelnen Tasks als auch des gesamten Systems beschrieben und außerdem werden sämtliche Ergebnisse präsentiert. Dazu wurden ein Trainingskorpus bestehend aus insgesamt 8.000 legitimen und 8.000 Phishing-Webseiten sowie ein Testkorpus bestehend aus insgesamt 3.200 legitimen und 3.200 Phishing-Webseiten verwendet. Die dazu in Betracht gezogenen legitimen URLs stellen eine zufällig ausgewählte Teilmenge des gesamten in [BBVV+17] verwendeten Korpus an legitimen URLs dar (bezogen von Common Crawl[1]). Die Phishing-URLs wurden ebenfalls zufällig aus der von PhishTank bezogenen Liste ausgewählt. Für das Herunterladen des zu den einzelnen URLs dazugehörigen HTML-Codes wurde eine eigene Klasse namens `CorpusCollector` implementiert, welche die URL und den dazugehörigen HTML-Inhalt jeder Webseite in eine eigene JSON-Datei speichert. Vor allem viele der von PhishTank bezogenen Phishing-Seiten waren zum Zeitpunkt der Erstellung der Korpusse bereits offline oder wurden in der Zwischenzeit vom jeweiligen Domänenanbieter gesperrt. In diesem Fall ist die heruntergeladene Webseite entweder leer oder beinhaltet lediglich ein kurzes Statement des Anbieters oder eine „404 Not Found“-Meldung. Eine inhaltsbasierte Analyse derartiger Webseiten ist einerseits nicht sinnvoll und verfälscht andererseits alle Berechnungen und Ergebnisse. Da jedoch eine manuelle Überprüfung dieses Umstandes bei dieser großen Anzahl an Webseiten zu viel Zeit in Anspruch nehmen würde, wurden im Zuge der Erstellung des Trainingskorpus zunächst zufällig 10.000 legitime und 10.000 Phishing-Webseiten heruntergeladen und die jeweils 2.000 mit dem geringsten Speicherbedarf vorhandenen Seiten aussortiert. Beim Testkorpus wurden hingegen 4.000 legitime und 4.000 Phishing-Seiten heruntergeladen und die jeweils 800 mit dem geringsten Speicherbedarf vorhandenen Seiten aussortiert. Dadurch kann zwar nicht gewährleistet werden, dass einerseits alle und andererseits wirklich nur derartige Seiten aussortiert werden, jedoch konnte mit dieser Herangehensweise nach stichprobenartiger Überprüfung ein Großteil davon beseitigt werden. Zusätzlich wurde sichergestellt, dass die beiden Korpusse disjunkt sind und somit keine Webseite in beiden vorkommt.

7.1 Evaluierung der Tasks

Zur Evaluierung und Berechnung der einzelnen Task-Parameter wurde der Trainingskorpus verwendet. Zunächst wurden für jene Tasks, welche einen internen Schwellenwert erfordern, die optimale Wahl dieser aus diesem Korpus berechnet (siehe Tabelle 7.1) und in die Konfigurationsdatei (`config.json`) übernommen.

Anschließend wurden sämtliche Task-Parameter berechnet. Wie in Tabelle 7.2 zusammengefasst, zählen die Tasks „Inhalt des Title-Tags“, „Verdächtiger JavaScript-Code“ und „Verdächtiger

[1] `http://commoncrawl.org`

Tab. 7.1: Interne Task-Schwellenwerte

Task	Evaluierter Bereich	Optimaler Schwellenwert
Domänenalter	$0 - 12$	12
Anzahl der Punkte	$0 - 20$	1
Anzahl der Satzzeichen	$0 - 70$	23
Leere Links	$0,0 - 1,0$	$0,33$

Klartext“ aus Sicht der Konfidenz zu den Top-Performers, „Verdächtige Links“, „Eingabefelder“, „Verdächtige Frames“ und „Nicht übereinstimmende URLs“ hingegen zu den Low-Performers. Sieht man sich die Sensitivität und Spezifität an, so heben sich ebenfalls die Tasks „Inhalt des Title-Tags“, „Verdächtiger JavaScript-Code“ und „Verdächtiger Klartext“ aufgrund ihres hohen Effektes von allen anderen ab, „Eingabefelder“, „Verdächtige Frames“, „Verdächtige Links“ und „Nichtübereinstimmende URLs“ sind hingegen auch hier ganz hinten angereiht. Die ersichtlichen Sicherheitsgrenzen wurden mit einer Genauigkeit (Accuracy) von 99 % berechnet. Hierbei konnte für keinen einzigen Task eine zulässige untere (Lower Limit), dafür jedoch für insgesamt fünf eine zulässige obere Grenze (Upper Limit) gefunden werden. Da es sich bei den meisten implementierten Heuristiken um binäre Analysetasks handelt (Score entweder 0,0 oder 1,0) liegen die Grenzwerte, bis auf wenige Ausnahmen, auch wie erwartet bei 0,0.

Tab. 7.2: Zusammenfassung aller berechneten Analystetask-Paramter (Sicherheitsgrenzen mit Genauigkeit 99 %)

Task	Konfidenz [%]	Sensitivität [%]	Spezifität [%]	Lower Limit [0, 1]	Upper Limit [0, 1]	Grenzwert [0, 1]
Domänenalter	61,97	28,18	94,58	0,0	1,0	0,0
IP-Adresse	50,25	0,50	100,00	0,0	**0,0**	0,0
Verdächtige URL	54,80	13,45	96,15	0,0	1,0	0,0
Anzahl der Punkte	56,27	27,16	85,38	0,0	1,0	0,0
Anzahl der Satzzeichen	50,64	1,74	99,54	0,0	1,0	0,0
Verdächtige Meta-Tags	63,29	36,18	90,40	0,0	**0,99**	**0,75**
Inhalt des Title-Tags	**77,78** ↑	**76,74** ↑	**78,83** ↑	0,0	1,0	0,0
Verdächtige Formulare	66,62	50,94	82,30	0,0	1,0	0,0
Eingabefelder	**40,18** ↓	**71,36** ↓	**8,99** ↓	0,0	1,0	0,0
Verdächtige Frames	**45,32** ↓	**6,75** ↓	**83,89** ↓	0,0	1,0	0,0
Keine Links im Body	55,69	11,41	99,98	0,0	**0,0**	0,0
Leere Links	65,51	34,61	96,40	0,0	1,0	0,0
Verdächtige Links	**39,33** ↓	**11,63** ↓	**67,03** ↓	0,0	1,0	0,0
Nicht übereinstimmende URLs	**49,84** ↓	**0,16** ↓	**99,53** ↓	0,0	1,0	0,0
Nur Script	51,59	3,20	99,98	0,0	**0,0**	0,0
Verdächtiger JavaScript-Code	**70,78** ↑	**43,06** ↑	**98,50** ↑	0,0	**0,99**	**0,97**
Identität der Webseite	58,27	23,53	93,01	0,0	1,0	0,0
Verdächtiger Klartext	**70,63** ↑	**55,76** ↑	**85,49** ↑	0,0	1,0	**0,87**

7.2 Evaluierung der Gewichtungsmodelle

Tabelle 7.3 zeigt einen Vergleich der erhaltenen Gewichtungen bei Anwendung der beiden in Kapitel 5.4.4 beschriebenen Methoden. Wird die normalisierte Konfidenz als Gewichtung herangezogen, liegen die erhaltenen Werte relativ knapp beisammen und weisen nur eine geringe Streuung von in etwa $\pm 0,04$ auf, obwohl einige Tasks erheblich besser als andere sind. Wird hingegen der Effekt als Maß für die Gewichtung verwendet, erhöht sich die Streuung auf in etwa $\pm 0,13$ und somit haben bessere bzw. effektivere Tasks eine weitaus höhere Auswirkung auf das Gesamtergebnis. Zusätzlich werden bei diesem Modell Tasks mit Effekt ≤ 0 als schlecht bzw. nicht effektiv angesehen und bekommen automatisch eine Gewichtung von 0, wodurch sie bei der Berechnung des Gesamtergebnisses nicht berücksichtigt bzw. ausgeschlossen werden.

Tab. 7.3: Vergleich der beiden Gewichtungsmodelle (Abweichungen aufgrund von Rundungsfehlern möglich)

Task	Konfidenz [%]	Sensitivität [%]	Spezifität [%]	Effekt [%]	Gewichtung Konfidenz [0, 1]	Gewichtung Effekt [0, 1]
Domänenalter	61,97	28,18	94,58	22,76	0,0602	0,0741
IP-Adresse	50,25	0,50	100,00	0,50	**0,0488**	**0,0016** ↓
Verdächtige URL	54,80	13,45	96,15	9,60	0,0533	0,0313
Anzahl der Punkte	56,27	27,16	85,38	12,54	0,0547	0,0408
Anzahl der Satzzeichen	50,64	1,74	99,54	1,28	**0,0492**	**0,0042** ↓
Verdächtige Meta-Tags	63,29	36,18	90,40	26,58	0,0615	0,0866
Inhalt des Title-Tags	77,78	76,74	78,83	55,57	**0,0756**	**0,1810** ↑
Verdächtige Formulare	66,62	50,94	82,30	33,24	**0,0648**	**0,1083** ↑
Eingabefelder	40,18	71,36	8,99	**0,0**	**0,0391**	**0,0** ↓
Verdächtige Frames	45,32	6,75	83,89	**0,0**	**0,0441**	**0,0** ↓
Keine Links im Body	55,69	11,41	99,98	11,39	0,0541	0,0371
Leere Links	65,51	34,61	96,40	31,01	**0,0637**	**0,1010** ↑
Verdächtige Links	39,33	11,63	67,03	**0,0**	**0,0382**	**0,0** ↓
Nicht übereinstimmende URLs	49,84	0,16	99,53	**0,0**	**0,0485**	**0,0** ↓
Nur Script	51,59	3,20	99,98	3,18	0.0501	0,0103
Verdächtiger JavaScript-Code	70,78	43,06	98,50	41,56	**0,0688**	**0,1354** ↑
Identität der Webseite	58,27	23,53	93,01	16,54	0,0566	0,0539
Verdächtiger Klartext	70,63	55,76	85,49	41,25	**0,0687**	**0,1344** ↑

7.3 Evaluierung des gesamten Systems

Für die Evaluierung des gesamten Systems wurde der Testkorpus bestehend aus 3.200 legitimen und 3.200 Phishing-Webseiten herangezogen. Um anhand des Gesamtergebnisses (finalScore), welches einen reellen Wert zwischen 0,0 und 1,0 darstellt, eine Aussage bezüglich der Legitimität einer Webseite treffen zu können, ist zusätzlich noch ein Gesamtgrenzwert (overallThreshold) notwendig. Dieser wurde nach dem gleichen Prinzip wie auch schon die Grenzwerte der einzelnen Tasks aus dem Trainingskorpus berechnet und kennzeichnet jenes Gesamtergebnis, bis zu

welchem eine Webseite als legitim bzw. ab welchem als Phishing-Seite eingestuft wird. Dieser Wert wird mit Hilfe einer eigenen Klasse namens `OverallThresholdCalculator` berechnet und muss manuell in die Konfigurationsdatei (`config.json`) übertragen werden.

Für die Evaluierung wurden die folgenden Metriken in Betracht gezogen:

- Genauigkeit = $\frac{TP+TN}{TP+TN+FP+FN}$
- Richtig-Positiv-Rate = $\frac{TP}{TP+FN}$
- Richtig-Negativ-Rate = $\frac{TN}{TN+FP}$
- Falsch-Positiv-Rate = $\frac{FP}{TN+FP}$
- Falsch-Negativ-Rate = $\frac{FN}{TP+FN}$

Die Genauigkeit des Systems wurde mit den beiden verschiedenen Gewichtungsmodellen und mit unterschiedlichen Genauigkeiten zur Berechnung der Sicherheitsgrenzen evaluiert. Als Methode für die Bestimmung des Safety Limit Selector Ergebnisses wurde jeweils immer die Methode `takeMajority()` herangezogen, da sich hier bei den drei verschiedenen Ansätzen ab einer bestimmten Genauigkeit keine Unterschiede ergeben. Tabelle 7.4 fasst alle Ergebnisse dazu zusammen.

Tab. 7.4: Ergebnisse mit unterschiedlichen Konfigurationen (mit Safety Limit Selector Methode `takeMajority()`)

Gewichtungsmodell	Sicherheitsgrenzen Genauigkeit [%]	Genauigkeit [%]	Richtig-Positiv [%]	Richtig-Negativ [%]	Falsch-Negativ [%]	Falsch-Positiv [%]
Konfidenz	90,00	74,20	80,13	68,28	19,87	31,72
	95,00	82,06	75,25	88,88	24,75	11,12
	99,00	83,69	72,56	94,81	27,44	5,19
	100,00	78,69	71,59	85,78	28,41	14,22
Effekt	90,00	74,47	83,00	65,94	17,00	34,06
	95,00	81,83	75,25	88,41	24,75	11,59
	99,00	**84,17**	**78,13**	**90,22**	**21,87**	**9,78**
	100,00	81,72	74,75	88,69	25,25	11,31

Wie aus Tabelle 7.4 entnommen werden kann, ist die Genauigkeit des Systems mit 84,17 % bei Verwendung des Effekts als Gewichtungsmodell und bei einer Berechnung der Sicherheitsgrenzen mit einer Genauigkeit von 99 % am höchsten. Die Sensitivität (Richtig-Positiv-Rate) liegt hierbei bei 78,13 % und die Spezifität (Richtig-Negativ-Rate) bei 90,22 %.

Die Ergebnisse widerspiegeln ebenfalls die bei der Evaluierung der Gewichtungsmodelle auftretende Vermutung, dass mit dem Effekt als Gewichtungsmodell grundsätzlich bessere Ergebnisse und somit eine höhere Genauigkeit erzielt werden kann (bis auf eine Ausnahme). Bezüglich der Berechnung der Sicherheitsgrenzen ist ersichtlich, dass die Ergebnisse mit steigender Genauigkeit besser werden und ihren Höhepunkt bei einer Berechnung mit 99 % erreichen. Die Wahl von 100 % ist hier hingegen zu restriktiv und lässt die Ergebnisse wiederum fallen.

8 Fazit und Ausblick

Methoden und Ansätze des Social Engineerings werden aufgrund ihrer hohen Erfolgsaussichten mittlerweile zu immer wesentlicheren Bestandteilen von herkömmlichen Hacker-Angriffen. Der Fokus zur Kompromittierung eines Systems oder zur Erlangung von vertraulichen Informationen liegt hierbei nicht auf technischen Aspekten und Sicherheitslücken, sondern ist hauptsächlich auf den Faktor Mensch als schwächstes Glied der Sicherheitskette gerichtet. Phishing zählt hierbei zu den effektivsten und zugleich auch am weitest verbreiteten Methoden dieser Art und stellt laut aktuellen Daten eine jährlich steigende Bedrohung für unsere Gesellschaft dar. Mit Hilfe von gefälschten Nachrichten und manipulierten Webseiten versuchen Angreifer mit dieser Masche über die verschiedensten Kanäle personenbezogene und vertrauliche Informationen zu ergaunern, was in weiterer Folge für das Opfer einen Schaden beliebiger Natur nach sich ziehen kann. Um Angriffen dieser Art vorzubeugen und diese zu vermeiden, wurden in dieser Arbeit einige mögliche Gegenmaßnahmen aufgezeigt, wobei vor allem unter Berücksichtigung der nichttechnischen Maßnahmen ein effektiver und maximaler Schutz gewährleistet werden kann. Zwar kann der Benutzer dabei mit Hilfe eines bereits breitem Spektrums an technischen Lösungen unterstützt werden, jedoch bleiben vor allem die Awareness und Bildung die beiden maßgeblichen Erfolgsfaktoren.

Der in dieser Arbeit vorgestellte holistische Ansatz zur Erkennung von Phishing-Webseiten dient als letzte Sicherheitsmaßnahme, um einen Phishing-Angriff erfolgreich entdecken und abwehren zu können. Erst wenn der Benutzer tatsächlich auf eine trügerische Nachricht hereingefallen ist und schließlich auf den schadhaften Link klickt, kann er durch die automatisierte Analyse der erscheinenden Webseite noch vor einem derartigen Angriff und den daraus resultierenden Folgen bewahrt werden. Das Konzept dahinter basiert auf einer Client-Server-Architektur und beinhaltet einen aus insgesamt sechs übergeordneten Schritten bestehenden Analyseprozess (1. Analyse der URL, 2. Analyse der Metadaten, 3. Analyse der Struktur, 4. Analyse des Inhaltes, 5. Trained Learner und 6. User Profile Matching), welcher auf Basis der Einzelergebnisse ein Gesamtergebnis berechnet und die Webseite schlussendlich in ein grünes, gelbes oder rotes Ampelsymbol einstuft.

Der implementierte Prototyp umfasst den Aufbau der gesamten Architektur sowie die clientseitige Implementierung eines Browser-Plugins und die serverseitige Implementierung der Analyseschritte 1-4. Dazu werden neben dem Abgleich der URL mittels einer White- und einer Blacklist insgesamt 18 verschiedene Heuristiken in Betracht gezogen, welche sowohl die URL als auch die Metadaten, die Struktur und den Inhalt der Webseite unter die Lupe nehmen. Zur Berechnung sämtlicher dazu notwendigen Parameter und zur Evaluierung des gesamten Systems wurden zwei verschiedene Korpusse, bestehend aus insgesamt 11.200 Phishing-Webseiten von PhishTank und insgesamt 11.200 legitimen Seiten von Common Crawl, verwendet. Mit einer

Genauigkeit von 84,17 %, einer Sensitivität von 78,13 % und einer Spezifität von 90,22 % zeigten die Ergebnisse, dass bereits mit dem implementierten Prototyp durchaus gute Resultate erzielt werden konnten. Aufgrund der doch zahlreichen Konfigurationsmöglichkeiten des gesamten Systems kann hier zusätzlich erwähnt werden, dass allein durch eine optimale und somit noch besseren Abstimmung der einzelnen Parameter mit dem derzeitigen Prototyp durchwegs noch bessere Ergebnisse möglich sind.

Ansonsten ist der Prototyp und zugleich auch das gesamte Konzept grundsätzlich an allen Ecken und Kanten ausbaufähig bzw. erweiterbar. Neben der bereits erwähnten Feinabstimmung der Konfiguration kann auch die Verbesserung der einzelnen Analysetasks durchaus gewinnbringend sein. Beispielsweise beschränkt sich die Analyse des Klartextes derzeit ohne Berücksichtigung der Sprache nur auf Keywords. Hier kann das Potenzial durch die Implementierung eines Spracherkennungsmoduls, das Miteinbeziehen von Phrasen sowie die Untersuchung mit Hilfe von computerlinguistischen Methoden (zur Erkennung von Rechtschreib- und Grammatikfehlern, falschen Satzstellungen, unvollständigen Sätzen, usw.), um nur einige Verbesserungsvorschläge bzw. Erweiterungen zu nennen, erheblich ausgeschöpft werden. Des Weiteren wird der auf einer Webseite vorhandene JavaScript-Code derzeit ebenfalls nur mit Hilfe eines naiven Ansatzes analysiert, welcher jedoch bereits angemessene Resultate liefert. Aufgrund der Tatsache, dass in vielen Fällen der JavaScript-Code einer Webseite der Schlüssel zur erfolgreichen Erkennung einer Phishing-Webseite ist, wäre eine Erweiterung in diese Richtung äußerst sinnvoll. Beispielsweise kann hierzu die Analyse auf bestimmte Patterns erweitert werden. Weiters wird derzeit nur der statische HTML-Code geladen und analysiert, wobei die Ausführung und somit Veränderungen durch den vorhandenen JavaScript-Code nicht berücksichtigt werden. Hierbei würde das Laden mittels einer Browser-Engine (wie z.B. Selenium[1]) und somit die Analyse des interpretierten Codes vermutlich ebenfalls zu deutlich akkurateren Ergebnissen führen.

Wie bereits im Konzept beschrieben, ist ebenfalls ein Klassifikator, ein Benutzerprofilabgleich sowie eine sichere SSL/TLS-Verbindung zwischen Client und Server vorgesehen. Vor allem die Einbindung eines auf maschinellen Lernens beruhenden Klassifikators (wie z.B. eines Random Forests oder Ähnlichem) wird sich vermutlich in deutlich bessere Resultaten widerspiegeln. Derzeit wird das Gesamtergebnis anhand der Einzelergebnisse und der dazugehörigen Sicherheitsgrenzen bestimmt, jedoch bleibt die Kombination der anschlagenden bzw. nicht-anschlagenden Analysetasks unberücksichtigt – dieser Aspekt wird beispielsweise von solchen Klassifikatoren ebenfalls analysiert. Durch die Erstellung eines individuellen Benutzerprofils kann außerdem jede aufgerufene Webseite mit dem derzeitigen Surfverhalten abgeglichen und somit gravierende Abweichungen festgestellt werden. Diese können anschließend entweder direkt in das Gesamtergebnis miteinbezogen oder in Form einer zusätzlichen Ampel im Browser dem Benutzer visuell dargestellt werden. Die Kommunikation zwischen Client und Server erfolgt derzeit noch über einen ungesicherten Kanal, wobei in weiterer Folge die Einrichtung einer sicheren SSL/TLS-Verbindung empfohlen wird.

Neben den bereits aufgezählten Ausbaumöglichkeiten und Verbesserungsvorschlägen, kann der Prototyp weiters aufgrund seines modularen Aufbaus relativ einfach um weitere Heuristiken bzw. Analysetasks, Gewichtungsmodelle und Methoden zur Bestimmung des Ergebnisses des Safety Limit Selectors erweitert werden.

[1] `http://www.seleniumhq.org/`

Literaturverzeichnis

[AaMa17] G. Aaron, R. Manning: Phishing Activity Trends Report, 4th Quarter 2016. http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf (2017), [Stand 27. Juli 2017].

[AaRa16] G. Aaron, R. Rasmussen: Global Phishing Survey: Trends and Domain Name Use. http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf (2016), [Stand 04. August 2017].

[Ali15] A. Ali: Social Engineering: Phishing latest and future techniques. https://www.researchgate.net/publication/274194484 (2015), [Stand 29. Juli 2017].

[Alse14] I. M. A. Alseadoon: The impact of users' characteristics on their ability to detect phishing emails. Dissertation, Queensland University of Technology (2014), https://eprints.qut.edu.au/72873/.

[ASKV+12] N. Adhikary, R. Shrivastava, A. Kumar, S. K. Verma, M. Bag, V. Singh: Battering keyloggers and screen recording software by fabricating passwords. In: *International Journal of Computer Network and Information Security*, 4, 5 (2012), 13.

[BaBa13] M. N. Banu, S. M. Banu: A comprehensive study of phishing attacks. In: *International Journal of Computer Science and Information Technologies*, 4, 6 (2013), 783–786.

[BBVV+17] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, F. A. González: Classifying phishing URLs using recurrent neural networks. *In: 2017 APWG Symposium on Electronic Crime Research (eCrime)* (2017), 1–8.

[BuVE13] W. Burgers, R. Verdult, M. van Eekelen: Prevent Session Hijacking by Binding the Session to the Cryptographic Network Credentials, Springer Berlin Heidelberg, Berlin, Heidelberg (2013), 33–50, https://doi.org/10.1007/978-3-642-41488-6_3.

[CCCL+17] K. Chandrasekar, G. Cleary, O. Cox, H. Lau, B. Nahorney, B. O Gorman, D. O'Brien, S. Wallace, P. Wood, C. Wueest: Internet Security Threat Report. https://s1.q4cdn.com/585930769/files/doc_downloads/lifelock/ISTR22_Main-FINAL-APR24.pdf (2017), [Stand 28. Juli 2017].

[ChCR16] J. A. Chaudhry, S. A. Chaudhry, R. G. Rittenhouse: Phishing attacks and defenses. In: *International Journal of Security and Its Applications*, 10, 1 (2016), 247–256.

[CLTM+04] N. Chou, R. Ledesma, Y. Teraguchi, J. C. Mitchell, : Client-Side Defense Against Web-Based Identity Theft. *In: NDSS* (2004).

[Dro] Dropwizard. http://www.dropwizard.io/1.2.0/docs/, [Stand 15. Oktober 2017].

[FaLR17] W. Fan, K. Lwakatare, R. Rong: Social engineering: Ie based model of human weakness for attack and defense investigations. In: *International Journal of Computer Network and Information Security*, 9, 1 (2017), 1.

[FeST07] I. Fette, N. Sadeh, A. Tomasic: Learning to Detect Phishing Emails. *In: Proceedings of the 16th International Conference on World Wide Web*, WWW '07, ACM, New York, NY, USA (2007), 649–656, `http://doi.acm.org/10.1145/1242572.1242660`.

[GaJa12] M. M. Gaurav, A. Jain: Anti-Phishing Techniques: A Review. In: *International Journal of Engineering Research and Applications*, 2, 2 (2012), 350–355.

[Goo] Google Safe Browsing APIs. `https://developers.google.com/safe-browsing/v4/`, [Stand 24. August 2017].

[Gran01] S. Granger: Social engineering fundamentals, part I: hacker tactics. In: *Security Focus, December*, 18 (2001).

[GTJA16] B. B. Gupta, A. Tewari, A. K. Jain, D. P. Agrawal: Fighting against phishing attacks: state of the art and future challenges. In: *Neural Computing and Applications* (2016), `https://doi.org/10.1007/s00521-016-2275-y`.

[Gua] Guava. `https://github.com/google/guava`, [Stand 15. Oktober 2017].

[GuAP17] B. B. Gupta, N. A. G. Arachchilage, K. E. Psannis: Defending against phishing attacks: taxonomy of methods, current issues and future directions. In: *Telecommunication Systems* (2017), `https://doi.org/10.1007/s11235-017-0334-z`.

[Gupt08] M. Gupta: Social and Human Elements of Information Security: Emerging Trends and Countermeasures: Emerging Trends and Countermeasures. IGI Global (2008).

[HaWi10] C. Hadnagy, P. Wilson: Social Engineering: The Art of Human Hacking. Wiley (2010).

[Hong12] J. Hong: The Current State of Phishing Attacks. In: *Communications of the ACM*, 55, 1 (2012), 74–81.

[Hyb] Hybernate Validator. `http://hibernate.org/validator/`, [Stand 15. Oktober 2017].

[Jac] Jackson. `https://github.com/FasterXML/jackson`, [Stand 15. Oktober 2017].

[JaGu16] A. K. Jain, B. B. Gupta: A novel approach to protect against phishing attacks at client side using auto-updated white-list. In: *EURASIP Journal on Information Security*, 2016, 1 (2016), 9, `https://doi.org/10.1186/s13635-016-0034-3`.

[JaGu17] A. K. Jain, B. Gupta: Phishing Detection: Analysis of Visual Similarity Based Approaches. In: *Security and Communication Networks*, 2017 (2017).

[JDB] JDBI. `http://www.jdbi.org/`, [Stand 15. Oktober 2017].

[Jer] Jersey. `https://jersey.github.io/`, [Stand 15. Oktober 2017].

[Jet] Jetty. `http://www.eclipse.org/jetty/`, [Stand 15. Oktober 2017].

[JMSP17] J. W. Joo, S. Y. Moon, S. Singh, J. H. Park: S-Detector: an enhanced security model for detecting Smishing attack for mobile computing. In: *Telecommunication Systems*, 66, 1 (2017), 29–38, https://doi.org/10.1007/s11235-016-0269-9.

[Kapo15] K. K. Kapoor: Phishing Attacks and their preventing Methodologies. In: *International Journal of Scientific and Research Publications (IJSRP)*, 5, 8 (2015).

[KaSe14] R. Karthikeyan, R. Sethuraman: Phishing Website Detection and Prevention of Phishing Attacks: a Field Experiment. In: *International Journal of Science, Engineering and Technology Research (IJSETR)*, 3, 2 (2014).

[Kee08] J. Kee: Social engineering: Manipulating the source. In: *GCIA Gold Certification* (2008).

[KHHW15] K. Krombholz, H. Hobel, M. Huber, E. Weippl: Advanced social engineering attacks. In: *Journal of Information Security and Applications*, 22 (2015), 113 – 122, http://www.sciencedirect.com/science/article/pii/S2214212614001343, special Issue on Security of Information and Networks.

[KiAS04] M. Kilger, O. Arkin, J. Stutzman: Profiling. In: *The Honeynet Project (2 nd Ed.), Know your enemy. Addison Wesley Professional* (2004).

[KiKr05] E. Kirda, C. Kruegel: Protecting Users Against Phishing Attacks with AntiPhish. *In: Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01*, COMPSAC '05, IEEE Computer Society, Washington, DC, USA (2005), 517–524, http://dx.doi.org/10.1109/COMPSAC.2005.126.

[Klen08] J. Klensin: Simple Mail Transfer Protocol. RFC 5321, RFC Editor (2008), http://www.rfc-editor.org/rfc/rfc5321.txt, http://www.rfc-editor.org/rfc/rfc5321.txt.

[K.Vi10] P. K. Sengar, K. Vijay: Client-Side Defense against Phishing with PageSafe. In: , 4 (2010).

[Lips09] M. Lipski: Social Engineering: Der Mensch als Sicherheitsrisiko in der IT. Diplomica-Verlag (2009).

[LiVi05] R. Lininger, R. D. Vines: Phishing: Cutting the identity theft line. John Wiley & Sons (2005).

[Log] Logback. https://logback.qos.ch/, [Stand 15. Oktober 2017].

[LoMi11] J. Long, K. Mitnick: No Tech Hacking: A Guide to Social Engineering, Dumpster Diving, and Shoulder Surfing. Elsevier Science (2011).

[Ma13] Q. Ma: The process and characteristics of phishing attacks-A small international trading company case study. In: *Journal of Technology Research*, 4 (2013), 1.

[Mann12] I. Mann: Hacking the Human: Social Engineering Techniques and Security Countermeasures. Ashgate Publishing Limited (2012).

[Mav] Maven. http://maven.apache.org/, [Stand 15. Oktober 2017].

[MeKK08] E. Medvet, E. Kirda, C. Kruegel: Visual-similarity-based Phishing Detection. *In: Proceedings of the 4th International Conference on Security and Privacy in Commu-*

nication Netowrks, SecureComm '08, ACM, New York, NY, USA (2008), 22:1–22:6, http://doi.acm.org/10.1145/1460877.1460905.

[Met] Metrics. http://metrics.dropwizard.io/3.2.3/, [Stand 15. Oktober 2017].

[Mill05] J. Milletary: Technical Trends in Phishing Attacks. Tech. Rep., US-CERT (2005), https://www.us-cert.gov/sites/default/files/publications/phishing_trends0511.pdf.

[MiSi06] K. Mitnick, W. Simon: Die Kunst der Täuschung: Risikofaktor Mensch. mitp (2006).

[MMLV14] F. Mouton, M. M. Malan, L. Leenen, H. S. Venter: Social engineering attack framework. *In: Information Security for South Africa (ISSA), 2014*, IEEE (2014), 1–9.

[MoTM15] R. M. Mohammad, F. Thabtah, L. McCluskey: Tutorial and Critical Analysis of Phishing Websites Methods. In: *Comput. Sci. Rev.*, 17, C (2015), 1–24, http://dx.doi.org/10.1016/j.cosrev.2015.04.001.

[Net04] Netcraft Toolbar. http://toolbar.netcraft.com/ (2004), [Stand 20. August 2017].

[Nohl08] M. Nohlberg: Securing Information Assets : Understanding, Measuring and Protecting against Social Engineering Attacks. Dissertation, Stockholm University, Department of Computer and Systems Sciences (2008).

[NoKo08] M. Nohlberg, S. Kowalski: The cycle of deception: a model of social engineering attacks, defenses and victims. *In: Second International Symposium on Human Aspects of Information Security and Assurance (HAISA 2008), Plymouth, UK, 8-9 July 2008*, University of Plymouth (2008), 1–11.

[Ollm07] G. Ollmann: The Phishing Guide - Understanding & Preventing Phishing Attacks. Tech. Rep., Next Generation Security Software Limited (2007), http://www-935.ibm.com/services/us/iss/pdf/phishing-guide-wp.pdf, [Stand 27. Juli 2017].

[O'Mc12] G. O'Gorman, G. McDonald: The elderwood project. Symantec Corporation (2012).

[PaJK10] K. Pandove, A. Jindal, R. Kumar: Launching Email Spoofing Attacks. In: *International Journal of Computer Applications*, 5, 1 (2010), 21–22.

[PfPf15] C. P. Pfleeger, S. L. Pfleeger: Security in Computing. Prentice Hall Professional Technical Reference (2015).

[PKKG10] P. Prakash, M. Kumar, R. R. Kompella, M. Gupta: PhishNet: Predictive Blacklisting to Detect Phishing Attacks. *In: 2010 Proceedings IEEE INFOCOM* (2010), 1–5.

[RaAl15] R. S. Rao, S. T. Ali: PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach. In: *Procedia Computer Science*, 54 (2015), 147 – 156, http://www.sciencedirect.com/science/article/pii/S1877050915013411, eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015,

Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India.

[RaNa14] U. H. Rao, U. Nayak: The InfoSec Handbook: An Introduction to Information Security, Apress, Berkeley, CA, Kap. Social Engineering (2014), 307–323, http://dx.doi.org/10.1007/978-1-4302-6383-8_15.

[RJMB+05] B. Ross, C. Jackson, N. Miyake, D. Boneh, J. C. Mitchell: Stronger Password Authentication Using Browser Extensions. *In: Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, USENIX Association, Berkeley, CA, USA (2005), 2–2, http://dl.acm.org/citation.cfm?id=1251398.1251400.

[SEF] The Social Engineering Framework. http://www.social-engineer.org/framework/, [Stand 27. Juli 2017].

[Shak12] I. Shakeel: Iframe & the Security Risk. http://resources.infosecinstitute.com/iframe-security-risk/ (2012), [Stand 05. August 2017].

[ShKh14] J. L. Shah, A. I. Khan: Cross Site Scripting (XSS): The dark side of HTML. In: *International Journal Of Engineering And Computer Science*, 3, 3 (2014), 4066–4068.

[ShSa12] J. Shi, S. Saleem: CSc 566, Computer Security Research Reports: Phishing. Tech. Rep., University of Arizona (2012), https://www.cs.arizona.edu/~collberg/Teaching/466-566/2014/Resources/presentations/2012/reports.pdf, [Stand 29. März 2016].

[SrPa17] R. Srinivasa Rao, A. R. Pais: Detecting Phishing Websites Using Automation of Human Behavior. *In: Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, CPSS '17, ACM, New York, NY, USA (2017), 33–42, http://doi.acm.org/10.1145/3055186.3055188.

[TCWS16] C. L. Tan, K. L. Chiew, K. Wong, S. N. Sze: PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder. In: *Decision Support Systems*, 88 (2016), 18 – 27, http://www.sciencedirect.com/science/article/pii/S0167923616300781.

[Web] Mozilla Firefox WebExtensions API. https://developer.mozilla.org/en-US/Add-ons/WebExtensions, [Stand 15. Oktober 2017].

[Who] Whois. https://de.wikipedia.org/wiki/Whois, [Stand 17. November 2017].

[XHRC11] G. Xiang, J. Hong, C. P. Rose, L. Cranor: CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. In: *ACM Trans. Inf. Syst. Secur.*, 14, 2 (2011), 21:1–21:28, http://doi.acm.org/10.1145/2019599.2019606.

[Yane06] J. Yaneza: Spy-phishing: A new breed of blended threats. *In: Proc. Virus Bulletin Conference* (2006), Bd. 2006.

[YuWa10] C. Yue, H. Wang: BogusBiter: A Transparent Protection Against Phishing Attacks. In: *ACM Trans. Internet Technol.*, 10, 2 (2010), 6:1–6:31, http://doi.acm.org/10.1145/1754393.1754395.

[ZhHC07] Y. Zhang, J. I. Hong, L. F. Cranor: Cantina: A Content-based Approach to Detecting Phishing Web Sites. *In: Proceedings of the 16th International Conference on World Wide Web*, WWW '07, ACM, New York, NY, USA (2007), 639–648, http://doi.acm.org/10.1145/1242572.1242659.

[ZhLK09] C. V. Zhou, C. Leckie, S. Karunasekera: Collaborative Detection of Fast Flux Phishing Domains. In: *JNW*, 4, 1 (2009), 75–84.