OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Fraunhofer ACADEMY

Ostbayerische Technische Hochschule Regensburg
Fakultät Elektro- und Informationstechnik

Projektbericht zur Durchführung eines Penetrationstests

Prüfung zur hochschulübergreifenden Veranstaltung

Penetration Testing (PETE-M)

an der
Ostbayerischen Technischen Hochschule Amberg-Weiden

Vorgelegt von: Andreas Zinkl

Studiengang: Applied Research in Engineering Sciences

Dozent: Tilo Fischer

Abgabedatum: 10.09.2018

Inhaltsverzeichnis

1 Vereinbarung zum Penetrationstest

Zu Beginn des Penetrationstests werden in diesem Kapitel die Rahmenbedingungen und Vereinbarungen mit dem Kunden Frank Tope (wird im Folgenden als Kunde bezeichnet) festgehalten. Dabei wird zunächst die Motivation zur Durchführung des Penetrationstests erläutert. Im Anschluss daran werden die Ziele und der Ablauf des Penetrationstests definiert.

1.1 Einleitung

Frank Tope ist Software Entwickler und führt eine Webseite für seinen beruflichen Werdegang. Die Webseite dient ihm dabei als Resümee für potentielle Arbeitgeber. Auf der Webseite sind diesbezüglich Informationen zu seinen fachlichen Fähigkeiten, erlangten Abschlüssen, sowie seinem bisherigen Werdegang enthalten. Die Korrektheit dieser Daten ist für einen potentiellen Arbeitgeber, aber auch für den Kunden selbst von großer Wichtigkeit. Ein Angriff und eine damit einhergehende unbemerkte Veränderung der Webseite durch illegale oder rufschädigende Inhalte könnte potentielle Arbeitgeber abschrecken und die berufliche sowie private Zukunft des Kunden negativ beeinflussen.

Um sich vor Angriffen und der Verfälschung der Daten zu schützen, wird Andreas Zinkl (im Folgenden auch als Angreifer bezeichnet) beauftragt, die Webseite auf Sicherheitsschwachstellen zu überprüfen. Dabei sollen potentielle Schwachstellen entdeckt und Handlungsempfehlungen zur Behebung der Schwachstellen gegeben werden.

1.2 Das Ziel

Das Ziel des Penetrationstests ist der Webauftritt des Kunden (siehe Abbildung 1). Dieser ist über die Webadresse *http://192.168.1.99/* erreichbar. Weitere Informationen zur Webseite, dem verwendeten Server oder weiterer Software sind dem Angreifer nicht bekannt. Somit wird nach dieser Vereinbarung ein sogenannter „Blind“-Angriff durchgeführt. Der Kunde ist dabei über den Zeitpunkt (Samstag, 01.09.2018) und Ablauf des Angriffes informiert. Der Angreifer besitzt bis auf die genannte Webadresse keine weiteren Informationen über das Ziel.

1.3 Der Ablauf

Der Ablauf des Penetrationstests gliedert sich in zwei Hauptphasen. Zu Beginn des Tests wird in der ersten Phase das System des Kunden analysiert. Dabei wird ein sogenannter „non-invasiver“ Angriff durchgeführt. Ein „non-invasiver“Angriff beinhaltet die Informationsgewinnung über private Informationen, sowie Detailinformationen zum Ziel-System und eine zugehörige Risikobewertung ermittelter Schwachstellen. In der zweiten Phase wird ein „invasiver“ Angriff durchgeführt. Dieser Angriff nutzt die Ergebnisse des „non-invasiven“ Angriffes und beinhaltet das Eindringen des Angreifers in das Ziel-System. Mit Abschluss des Penetrationstests wird ein Fazit zum aktuellen Sicherheitsstand des Systems erläutert. Zudem werden für jede erkannte Schwachstelle Handlungsstrategien zur Behebung der Sicherheitsrisiken genannt.

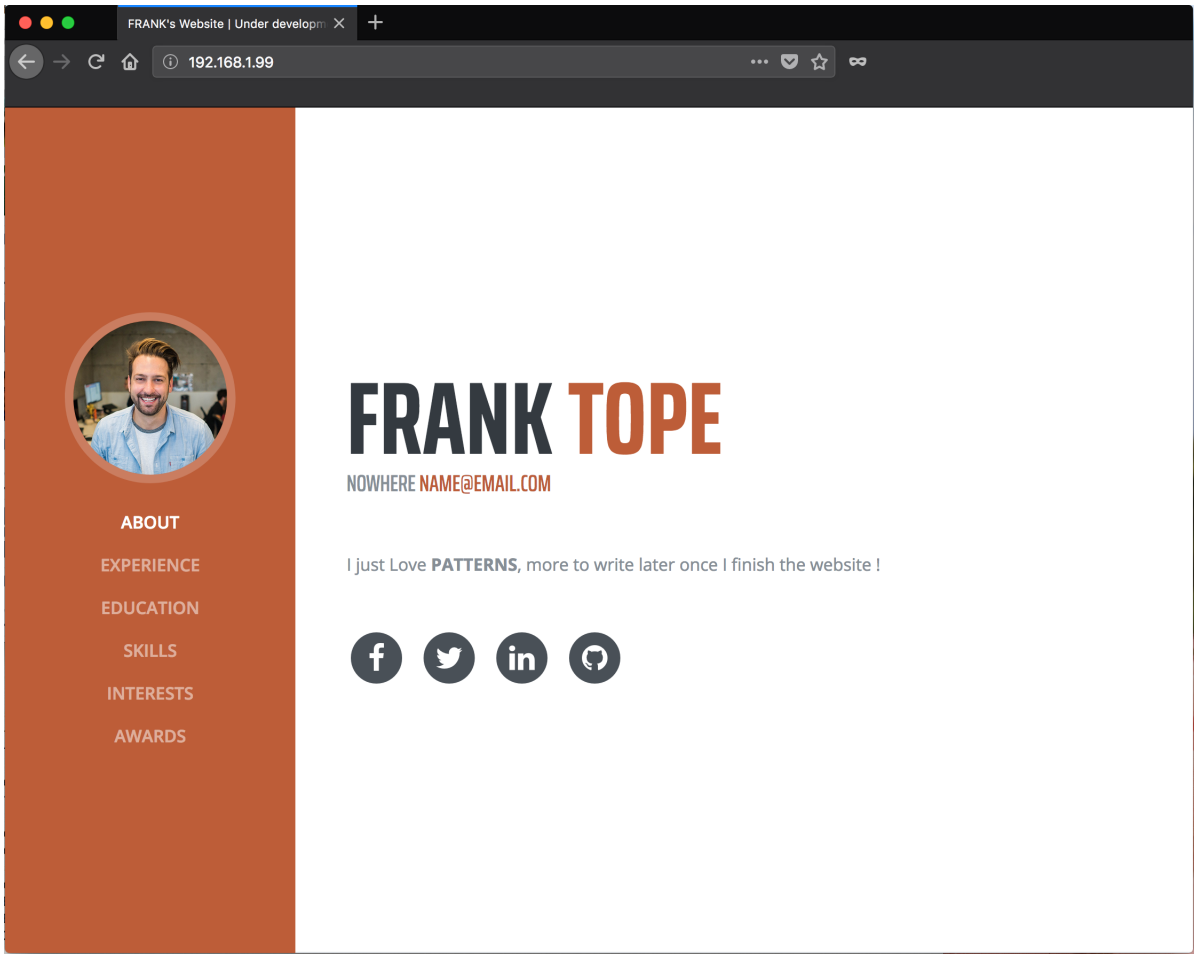

Abbildung 1: *Die Webseite des Kunden und Ziel des Penetrationstests*

2 Schwachstellenanalyse und Risikobewertung

In diesem Kapitel erfolgt zunächst eine allgemeine Einführung in die Analyse und Bewertung von Schwachstellen. Dabei werden zu Beginn die im Penetrationstest verwendeten Kennzahlen für eine Einordnung der Schwachstellen beschrieben. Anschließend folgt ein Überblick, sowie eine detaillierte Analyse der Schwachstellen. Im Überblick wird bereits eine zusammenfassende Risikoeinschätzung gegeben. Die Detailanalyse der Schwachstellen beinhaltet neben einer genauen Risikobewertung eine Handlungsempfehlung zum Umgang mit der jeweiligen Schwachstelle.

2.1 Allgemeine Einführung

Die Bewertung der Risiken einer Schwachstelle erfolgt anhand zweier Kennzahlen. Dies ist zum einen die Kennzahl der Eintrittswahrscheinlichkeit, sowie zum anderen die Kennzahl der Schadensbewertung. Jede Kennzahl beinhaltet eine Bewertung von verschiedenen Eigenschaften der jeweiligen Schwachstelle. Die Bewertung der Eintrittswahrscheinlichkeit wird anhand der in Tabelle 1 dargestellten Fragen und die Schadensbewertung anhand der in Tabelle 2 dargestellten Fragen durchgeführt.

Eintrittswahrscheinlichkeit	
Ist die verwendete Anwendung / der verwendete Service bekannt?	JA / NEIN
Existiert ein zugehöriger Eintrag im „Common Vulnerabilities and Exposures“ (CVE)?	JA / NEIN
Wurde über die Schwachstelle berichtet? (z.B. Nachrichten, Blogs)	JA / NEIN
Existiert ein Open-Source-Exploit?	JA / NEIN
Kann der Open-Source-Exploit einfach angewandt werden?	JA / NEIN
Funktioniert der Open-Source-Exploit jedes Mal?	JA / NEIN
Ist es eine Remote-Attacke? (Zugriff auf die Maschine des Opfers)	JA / NEIN

Tabelle 1: *Zu bewertende Eigenschaften in der Analyse der Eintrittswahrscheinlichkeit einer Schwachstelle*

Schadensanalyse	
Beeinflusst es das Tagesgeschäft?	JA / NEIN
Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand?	JA / NEIN
Gibt es einen finanziellen Schaden beim Opfer?	JA / NEIN
Gibt es eine mögliche Rufschädigung des Opfers?	JA / NEIN
Können administrative Rechte im System erlangt werden?	JA / NEIN

Tabelle 2: *Zu bewertende Eigenschaften in der Analyse des potentiellen Schadens bei Eintritt der Schwachstelle*

Somit ergeben sich für die Eintrittswahrscheinlichkeit und Schadensbewertung Kennzahlen von 0 (keine Eintrittswahrscheinlichkeit bzw. kein Risiko) bis maximal 7 (sehr hohe Eintrittswahrscheinlichkeit bzw. sehr hohes Risiko). Abschließend kann anhand der beiden ermittelten Kenn-

zahlen eine Gesamtkennzahl des Gesamtrisikos ermittelt werden. Dabei wird das Gesamtrisiko über den Mittelwert der Eintrittswahrscheinlichkeit und der Schadensbewertung berechnet. Je höher das Gesamtrisiko, desto dringender ist der Handlungsbedarf des Kunden.

2.2 Übersicht der entdeckten Schwachstellen und Risiken

Im Rahmen des Penetrationstests werden sieben wesentliche Schwachstellen festgestellt (siehe Tabelle 3). Damit besteht für die Webseite des Kunden zum aktuellen Zeitpunkt ein hohes Sicherheitsrisiko. Es besteht dringender Handlungsbedarf zur Sicherung des Systems. Bei vereinzelten Schwachstellen existieren bereits bekannte Exploits, welche das Eindringen in das System ermöglichen können. Ein Beweis zum erfolgreichen Eindringen in das System wird in Kapitel 3.2 beschrieben.

Nr.	**Schwachstelle**	**Eintreten**	**Schaden**	**Gesamtrisiko**
1	Veraltete Software-Versionen (SSH, FTP, Apache)	6	2	4
2	Anonymer Standard-FTP-Zugang (wenn auch eingeschränkt)	6	3	5
3	Administration per PHP-API-Schnittstelle	5	4	5
4	Schlechte SSH-Verschlüsselungsalgorithmen	4	6	5
5	Dateien auf dem Server enthalten Authentifizierungsdaten	5	6	6
6	Unzureichende Validierung von Datei-Uploads	6	6	6
7	Local-File-Intrusion (LFI) Schwachstelle	7	7	7

Tabelle 3: *Übersicht der Schwachstellen inklusive zugehöriger Risikobewertung*

2.3 Schwachstelle: Veraltete verwendete Software-Versionen

Service: FTP (Port: 21) / SSH (Port: 22) / HTTP (Port: 80 und 8011)

Beschreibung: Die Versionen des FTP-Services (vsftp 2.3.5), des SSH-Services (OpenSSH 5.9.p1), des HTTP-Services (Apache httpd 2.2.22) sowie das installierte PHP (5.3.10) sind sehr stark veraltet. Dies bedeutet eine Vielzahl von Schwachstellen hinsichtlich jedes einzelnen Services. Bei der Verwendung einer stark veralteten Software besteht meist ein hohes Risiko öffentlich bekannter Schwachstellen.

Risikobewertung: 4 (= mittleres Gesamtrisiko)

Eintrittswahrscheinlichkeit: 6 (= hohe Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - JA
* Wurde über die Schwachstelle berichtet? - JA
* Existiert ein Open-Source-Exploit? - JA
* Kann der Open-Source-Exploit einfach angewandt werden? - JA
* Funktioniert der Open-Source-Exploit jedes Mal? - JA
* Ist es eine Remote-Attacke? - NEIN

Schadensanalyse: 2 (= geringes Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - NEIN
* Gibt es einen finanziellen Schaden beim Opfer? - NEIN
* Gibt es eine mögliche Rufschädigung des Opfers? - NEIN
* Können administrative Rechte im System erlangt werden? - NEIN

Handlungsempfehlung: Jeder Service sollte stets auf der aktuellsten Version betrieben werden. Somit wird dem Kunden eine Aktualisierung des installierten PHP-Frameworks sowie der Software für den FTP-, SSH- und HTTP-Service empfohlen.

2.4 Schwachstelle: Anonymer Standard-FTP-Zugang

Service: FTP (Port: 21)

Beschreibung: Über den FTP-Service ist es möglich, mit Hilfe des anonymen Standard-Zugangs (Benutzer: *anonymous* und Passwort: *anonymous*) Zugriff auf den FTP-Server zu erhalten. Der Zugriff ist zwar beschränkt, da keinerlei Modifikationen auf dem System durchgeführt werden können, jedoch besteht eine Lese-Berechtigung. Somit können auf dem FTP-Zugang zur Verfügung gestellte Informationen von Angreifern abgefangen werden. Außerdem existieren bekannte CVE für Denial-of-Service-Angriffe zum gezielten Angriff auf Webserver.

Risikobewertung: 5 (= mittleres bis hohes Gesamtrisiko)

Eintrittswahrscheinlichkeit: 6 (= hohe Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - JA
* Wurde über die Schwachstelle berichtet? - JA
* Existiert ein Open-Source-Exploit? - JA
* Kann der Open-Source-Exploit einfach angewandt werden? - JA
* Funktioniert der Open-Source-Exploit jedes Mal? - NEIN
* Ist es eine Remote-Attacke? - JA

Schadensanalyse: 3 (= geringes Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - NEIN
* Gibt es einen finanziellen Schaden beim Opfer? - NEIN
* Gibt es eine mögliche Rufschädigung des Opfers? - NEIN
* Können administrative Rechte im System erlangt werden? - NEIN

Handlungsempfehlung: Sollte der FTP-Zugang nicht benötigt werden, wird eine Deaktivierung des Services empfohlen. Dateien können über die Webseite selbst verlinkt und zum Download zur Verfügung gestellt werden, somit wäre ein FTP-Zugang nicht zwingend erforderlich. Sollte ein Zugang jedoch benötigt werden, so sollte der anonyme Zugang deaktiviert werden, um bekannte Denial-of-Service-Attacken auf den Ziel-Server zu vermeiden.

2.5 Schwachstelle: Administration per PHP-API-Schnittstelle

Service: HTTP (Port: 80)

Beschreibung: Mittels ausführlicher Recherche können Muster zur Dateinennung sowie verfügbare, geplante und private Services in der öffentlich zugänglichen API der Webseite ermittelt werden (siehe Abbildung 2). Dies gibt einem Angreifer viele Informationen zur Infrastruktur und zum Aufbau der Webseite. Anhand dieser Informationen könnten mögliche Schwachstellen identifiziert werden. Zudem sind für PHP mit der Version 5.3.10 bereits viele CVE-Schwachstellen bekannt.

Risikobewertung: 5 (= mittleres bis hohes Gesamtrisiko)

Eintrittswahrscheinlichkeit: 5 (= mittlere bis hohe Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - JA
* Wurde über die Schwachstelle berichtet? - NEIN
* Existiert ein Open-Source-Exploit? - JA
* Kann der Open-Source-Exploit einfach angewandt werden? - JA
* Funktioniert der Open-Source-Exploit jedes Mal? - JA
* Ist es eine Remote-Attacke? - NEIN

Schadensanalyse: 4 (= mittleres Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - NEIN
* Gibt es einen finanziellen Schaden beim Opfer? - JA
* Gibt es eine mögliche Rufschädigung des Opfers? - JA
* Können administrative Rechte im System erlangt werden? - NEIN

Handlungsempfehlung: Die umfangreiche Auskunft über die verfügbare API zur Administration der Webseite ist ggf. nicht notwendig, da ausschließlich der Kunde selbst die API verwendet. Im Allgemeinen stellt sich zudem die Frage, ob eine solche PHP-API notwendig ist. Die Administration kann dabei auch über einen direkten Zugang erfolgen, was ebenfalls die Angriffsfläche verkleinert. Aktuelle Technologien (z.B. aktuelle Javascript-Frameworks) für die Entwicklung der API können die Sicherheit ebenso verbessern. Somit wäre eine Empfehlung die Deaktivierung der API und eine Administration über einen direkten SSH-Zugang über eine sichere VPN-Verbindung. Hierbei könnten ebenso Skripte auf dem Server zur Vereinfachung der Administration angewandt werden.

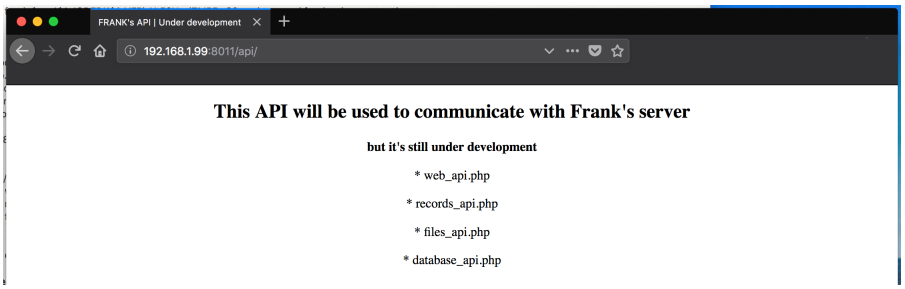

Abbildung 2: *Auflistung der, über die private API, verfügbaren PHP-Skripte und Funktionalitäten*

2.6 Schwachstelle: Schlechte und veraltete Verschlüsselungsalgorithmen

Service: SSH (Port: 22)

Beschreibung: Der SSH-Service verwendet eine Vielzahl an schlechten und veralteten Verschlüsselungsalgorithmen. Somit ist eine sichere Nutzung des SSH-Services nicht gewährleistet. Die Vielzahl der Verschlüsselungsalgorithmen weist ebenso auf eine Standard-Konfiguration des SSH-Services hin. Somit bestehen ggf. zusätzliche Risiken durch mögliche Falschkonfigurationen des SSH-Services.

Risikobewertung: 5 (= mittleres bis hohes Gesamtrisiko)

Eintrittswahrscheinlichkeit: 4 (= mittlere Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - JA
* Wurde über die Schwachstelle berichtet? - NEIN
* Existiert ein Open-Source-Exploit? - JA
* Kann der Open-Source-Exploit einfach angewandt werden? - NEIN
* Funktioniert der Open-Source-Exploit jedes Mal? - NEIN
* Ist es eine Remote-Attacke? - JA

Schadensanalyse: 6 (= hohes Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - NEIN
* Gibt es einen finanziellen Schaden beim Opfer? - NEIN
* Gibt es eine mögliche Rufschädigung des Opfers? - NEIN
* Können administrative Rechte im System erlangt werden? - JA

Handlungsempfehlung: Bei der Nutzung des SSH-Services sollte stets auf eine starke und sichere Verschlüsselung geachtet werden. Dabei müssen schwache Algorithmen für den Service deaktiviert werden, um das System zu schützen. Eine Auflistung der zu deaktivierenden Verschlüsselungsalgorithmen ist dabei aus Abbildung 3 zu entnehmen. Hierbei sind die schlechten bzw. schwachen Verschlüsselungsalgorithmen rot umrandet.

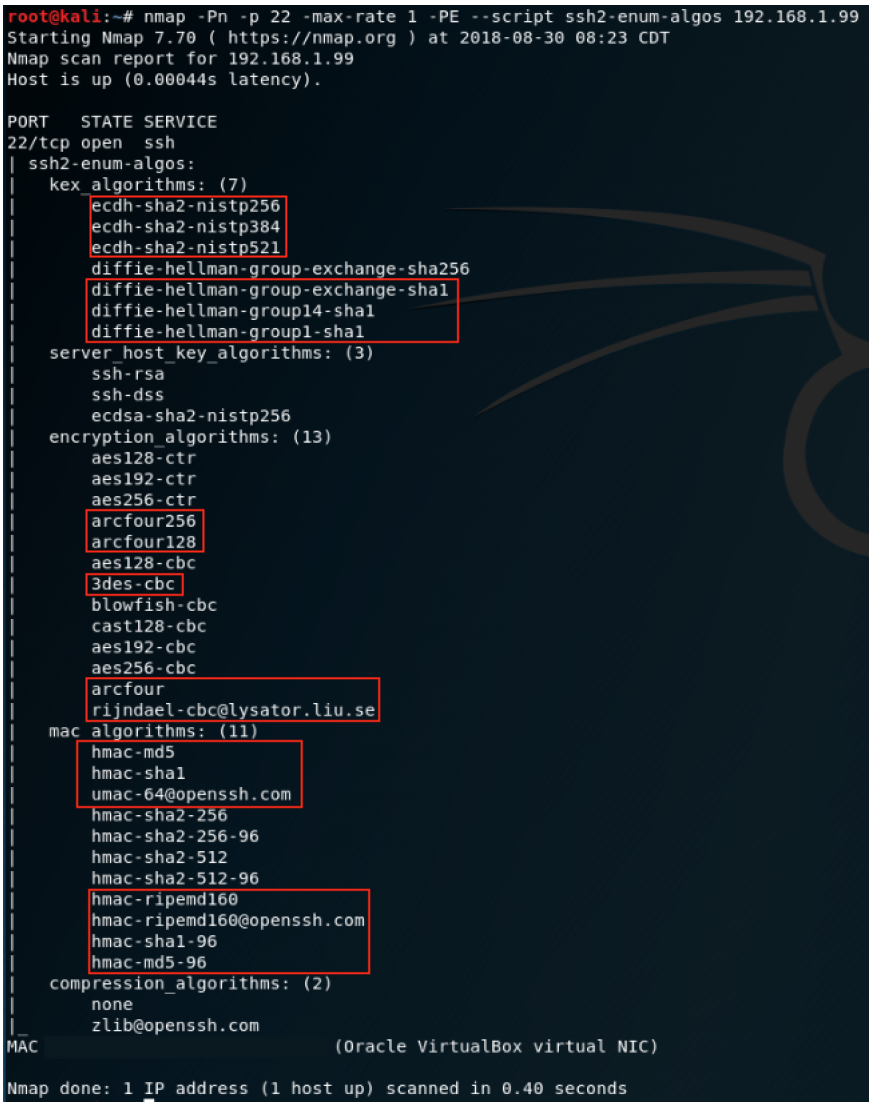

Abbildung 3: *Auflistung der verwendeten Verschlüsselungsalgorithmen für den SSH-Service*

2.7 Schwachstelle: Authentifizierungsinformationen in frei verfügbaren Dateien

Service: HTTP (Port: 80)

Beschreibung: Auf der Webseite ist eine alte Sicherungsdatei (*index.html.bak*) der Datei *index.html* vorhanden. Darin enthalten sind Authentifizierungsdaten für den Benutzer *frank* mit dem zugehörigen Passwort (siehe Abbildung 4). Das enthaltene Passwort ist hier zwar als Hash hinterlegt, kann jedoch reproduziert werden und entspricht dem Text *frank!!!* (siehe Abbildung 5). Somit sind Benutzer und Passwort für den Entwickler-Zugang (Link: *http://192.168.1.99/development*) öffentlich bekannt.

Risikobewertung: 6 (= hohes Gesamtrisiko)

Eintrittswahrscheinlichkeit: 5 (= mittlere bis hohe Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - NEIN
* Wurde über die Schwachstelle berichtet? - NEIN
* Existiert ein Open-Source-Exploit? - NEIN
* Kann der Open-Source-Exploit einfach angewandt werden? - NEIN
* Funktioniert der Open-Source-Exploit jedes Mal? - NEIN
* Ist es eine Remote-Attacke? - JA

Schadensanalyse: 6 (= hohes Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - JA
* Gibt es einen finanziellen Schaden beim Opfer? - JA
* Gibt es eine mögliche Rufschädigung des Opfers? - JA
* Können administrative Rechte im System erlangt werden? - NEIN

Handlungsempfehlung: Die Datei *index.html.bak* sollte umgehend vom Server gelöscht werden. Falls diese jedoch auf dem Server beibehalten werden soll, ist es dringend notwendig den Kommentar mit den Authentifizierungsinformationen aus der Datei zu entfernen.

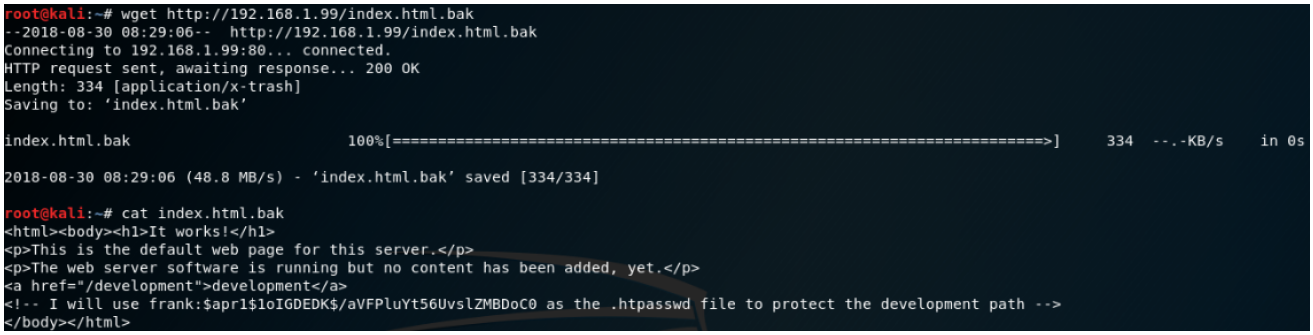

Abbildung 4: *Enthaltene Authentifizierungsdaten innerhalb der frei zugänglichen Sicherungsdatei index.html.bak*

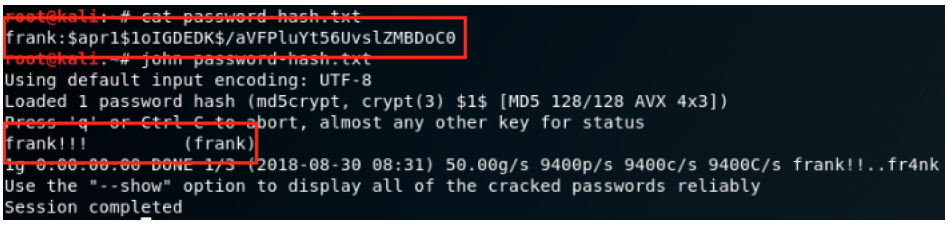

Abbildung 5: *Analyse der Authentifizierungsdaten aus der frei zugänglichen Sicherungsdatei index.html.bak*

2.8 Schwachstelle: Unzureichende Validierung von Datei-Uploads

Service: HTTP (Port: 80)

Beschreibung: Unter der Upload-Seite (Link: *http://192.168.1.99/development/uploader*) ist aktuell nur eine unzureichende Validierung der Dateien implementiert. Dateien werden ausschließlich anhand des Dateitypen validiert. Somit können Dateien unter Vortäuschung eines erlaubten Dateitypen auf den Server geladen werden.

Risikobewertung: 6 (= hohes Gesamtrisiko)

Eintrittswahrscheinlichkeit: 6 (= hohe Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - JA
* Wurde über die Schwachstelle berichtet? - NEIN
* Existiert ein Open-Source-Exploit? - JA
* Kann der Open-Source-Exploit einfach angewandt werden? - JA
* Funktioniert der Open-Source-Exploit jedes Mal? - JA
* Ist es eine Remote-Attacke? - JA

Schadensanalyse: 6 (= hohes Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - JA
* Gibt es einen finanziellen Schaden beim Opfer? - JA
* Gibt es eine mögliche Rufschädigung des Opfers? - JA
* Können administrative Rechte im System erlangt werden? - NEIN

Handlungsempfehlung: Ein Upload von Dateien auf den Server darf nur unter einer vollständig funktionierenden Validierung stattfinden. Somit sollte die aktuelle Version des Uploads deaktiviert werden und die Implementierung der Validierung von Dateien zunächst nur auf einem eigens dafür eingerichteten lokalen Test-System entwickelt und getestet werden. Erst nach einer Vervollständigung der entwickelten Validierung sollte das Upload-Portal online zur Verfügung gestellt werden.

2.9 Schwachstelle: Local-File-Intrusion Schwachstelle

Service: HTTP (Port: 8011)

Beschreibung: Die unter Port 8011 zur Verfügung gestellte API zur Administration der Webseite enthält eine sogenannte „Local-File-Intrusion“ (LFI) Schwachstelle. Eine LFI-Schwachstelle beschreibt dabei die Möglichkeit, durch Angabe eines Dateipfades, wie in Abbildung 6 dargestellt, auf Dateien im System zuzugreifen. Dies ermöglicht zum einen das Auslesen und zum anderen auch das Ausführen von zugänglichen Dateien. Hier können Remote-Shell-Attacken durch im System platzierte Dateien und über die Eigenschaften der LFI-Schwachstelle ausgeführt werden. Wie bereits durch Abbildung 6 gezeigt, besteht durch die LFI-Schwachstelle ein Lesezugriff auf den gesamten Server.

Risikobewertung: 7 (= hohes Gesamtrisiko)

Eintrittswahrscheinlichkeit: 7 (= hohe Eintrittswahrscheinlichkeit)

* Ist die verwendete Anwendung bekannt? - JA
* Existiert ein zugehöriger CVE-Eintrag? - JA
* Wurde über die Schwachstelle berichtet? - JA
* Existiert ein Open-Source-Exploit? - JA
* Kann der Open-Source-Exploit einfach angewandt werden? - JA
* Funktioniert der Open-Source-Exploit jedes Mal? - JA
* Ist es eine Remote-Attacke? - JA

Schadensanalyse: 7 (= hohes Schadensrisiko)

* Beeinflusst es das Tagesgeschäft? - JA
* Bedeutet eine Wiederherstellung des letzten sicheren Zustandes einen großen Aufwand? - JA
* Gibt es einen finanziellen Schaden beim Opfer? - JA
* Gibt es eine mögliche Rufschädigung des Opfers? - JA
* Können administrative Rechte im System erlangt werden? - JA

Handlungsempfehlung: Eine Änderung der API ist hier von großer Wichtigkeit. Sofern das Ausführen und das Auslesen von Dateien über das System nicht notwendig ist, sollte dieser Teil der API deaktiviert werden. Die LFI-Schwachstelle kann durch eine Änderung der API ebenso verhindert werden. Dabei ist eine Validierung der Eingabe erforderlich, um keine Systemdateien auslesen zu können. Außerdem könnten nur vom Kunden ausgewählte Dateien vom System ausgeführt werden dürfen, was das Risiko weiter reduziert. Dies verhindert beispielsweise das Ausführen von Dateien, die durch einen Angreifer im

System platziert wurden. Das Ausführen von Dateien könnte dabei generell verhindert und nur ein Lesezugriff auf ausgewählte Dateien und Dateiinhalte ermöglicht werden.

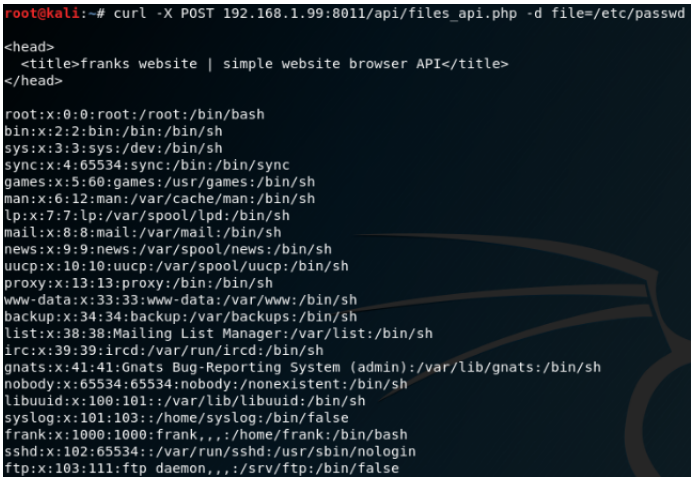

Abbildung 6: *Darstellung der Local-File-Intrusion Schwachstelle in der Ausnutzung der API-Schnittstelle*

3 Verlauf des Penetrationstests

In diesem Kapitel werden die zwei Phasen des Penetrationstests protokolliert. Dies ermöglicht das Nachvollziehen der ermittelten Schwachstellen für den Kunden. Zudem wird in Phase 2 ein Proof-of-Concept bezüglich eines Einbruchs in das System dargestellt und für den Kunden dokumentiert.

3.1 Phase 1 - Informationsgewinnung

Die Informationsgewinnung wird mit Hilfe der Tools *Firefox*, *dirb*, *nmap*, *uniscan*, sowie der jeweiligen Standard-Tools zur Nutzung der einzelnen verfügbaren Services durchgeführt.

Schritt 1 - Analyse der Webseite: Mit Beginn des Penetrationstests wird zunächst die Webseite des Kunden analysiert. Dabei wird auf der Webseite nach, für einen Angreifer, nützlichen Informationen gesucht. Hierbei werden ebenso die unter der Webseite erreichbaren Dateien ermittelt. Folgende Informationen zum Nutzer können dabei gesammelt werden:

- Vollständiger Name des Kunden
- Der Kunde ist sehr an Mustern (engl. „Patterns“) interessiert
- Interessante Daten zu Abschlüssen und Tagen

Bei der Analyse der Startseite (*index.html*) konnten keine weiteren Informationen zur Webseite entdeckt werden. Die Analyse verfügbarer Dateien unter der Adresse *http://192.168.1.99/* ergibt jedoch ein Resultat der Datei (*index.html.bak*). Hierbei sind Authentifizierungsinformationen enthalten, welche zwar als Hash-Wert hinterlegt sind, jedoch aber ohne weiteres nachberechnet werden können. Die ermittelten Ergebnisse sind hierbei:

- Anmeldename: *frank*
- Passwort: *frank!!!*
- Login-Adresse: *http://192.168.1.99/development*

Ein Versuch der Eingabe der Daten für die vorgesehene Login-Adresse zeigt eine Webseite, die auf ein Upload-Tool hinweist. Eine Analyse weiterer Sub-Adressen führt nun zur Ermittlung der folgenden Sub-Adressen für die Adresse *http://192.168.1.99/development*:

- */uploader*
- */uploader/FRANKuploads*

Erstere Seite ist für den Upload von Dateien zuständig. Dabei können die möglichen Dateiformate anhand des Validierungsskriptes ermittelt werden. Dies sind die Dateiformate JPG, JPEG, PNG und GIF. Unter der zweiten ermittelten Upload-Adresse können die über das Upload-Tool auf den Server geladenen Dateien eingesehen werden.

Eine Analyse weiterer Sub-Adressen unter der Hauptadresse *http://192.168.1.99/* ergibt zusätzlich die Sub-Adresse */api*. Diese Sub-Adresse führt dabei zu einer Übersichtsseite, welche auf mehrere PHP-Skripte zur Administration der Webseite hinweisen. Dabei ist aktuell lediglich die API-Schnittstelle */api/files_api.php* verfügbar. Eine Analyse dieses Dienstes ergibt dabei bereits die Möglichkeit einer „Local-File-Intrusion“ Schwachstelle, welche jedoch erst in Phase zwei näher betrachtet wird.

Schritt 2 - Analyse verfügbarer Services: Im zweiten Schritt der Informationsgewinnung werden nun die verfügbaren Services und Informationen zum Zielsystem ermittelt. Dabei werden die in Tabelle 4 dargestellten Services entdeckt. Zudem kann nun das Betriebssystem des Zielsystems auf Ubuntu mit einem Linux-Kernel 2.6.35 eingeschränkt werden.

Service	Port	Version
FTP	21	vsftpd 2.3.5
SSH	22	OpenSSH 5.9p1 Debian
HTTP	80	Apache httpd 2.2.22 Ubuntu
HTTP	8011	Apache httpd 2.2.22 Ubuntu

Tabelle 4: *Übersicht der ermittelten, verfügbaren Services auf dem Ziel-Server*

Schritt 3 - Analyse des FTP-Services: Die Analyse des FTP-Services ergibt neben der Software-Version *vsftp 2.3.5* ebenso das Resultat der Verfügbarkeit des Standard-Anonymous-Nutzers. Mit Hilfe des Nutzers kann ein Lese-Zugriff auf den FTP-Server erlangt werden. Auf dem FTP-Server sind jedoch keinerlei Informationen zu finden. Die Version *2.3.5* behebt zudem bekannte Exploits, welche zuvor mit Hilfe des Anonymous-Logins das Erlangen von Zugangsrechten ermöglicht haben. Somit bestehen keine weiteren Informationen zum FTP-Service.

Schritt 4 - Analyse des SSH-Services: Die Analyse des SSH-Services bietet kaum Spielraum für einen direkten möglichen Angriff. Es können lediglich Informationen zu den verwendeten Algorithmen und Authentifizierungsmethoden entnommen werden. Zudem wird die SSH-Version 1 im System nicht verwendet.

Schritt 5 - Analyse des HTTP-Services: Die Analyse des HTTP-Services zeigt hinsichtlich der Apache-Version 2.2.22 zunächst ausschließlich mögliche Schwachstellen im Bereich der Systemverfügbarkeit auf. Weitere Informationen über den HTTP-Service wurden dabei bereits in Schritt 1 durch eine allgemeine Analyse der Webseite ermittelt.

3.2 Phase 2 - Einbruch in das System

Schritt 1 - Vorbereitung: Für den Einbruch in das System wird zunächst ein Remote-Shell-Zugriff angestrebt. Dieser soll mit Hilfe der analysierten „Local-File-Intrusion“ (LFI) Schwach-

stelle erreicht werden. Hierfür wird zunächst der PHP-Exploit „php_reverse_tcp“ verwendet und aus der vorhandenen php_reverse_tcp.php eine hacking.gif erstellt. Hierzu wird der Dateityp der PHP-Datei zu .gif abgeändert und dem Dateiinhalt wird in der ersten Zeile der Text *GIF99* hinzugefügt.

Schritt 2 - Einschleusen eines Exploits: Der Einbruch in das System erfolgt nun mit Hilfe der Informationen aus der Schwachstellenanalyse. Dabei werden die Authentifizierungsdaten (Benutzer: *frank* und Passwort *frank!!!*) für den Zugang zur Entwicklerseite (Link: *http://192.168.1.99/development/*) verwendet. Im Entwicklerbereich kann nun mit Hilfe des Uploaders (Link: *http://192.168.1.99/development/uploader*) die generierte hacking.gif Datei, welche den PHP-Exploit „php_reverse_tcp“ enthält, hochgeladen werden (siehe Abbildung 7). Diese ist nun unter dem Link *http://192.168.1.99/development/uploader/FRANKuploads* zu finden und erreichbar (siehe Abbildung 8).

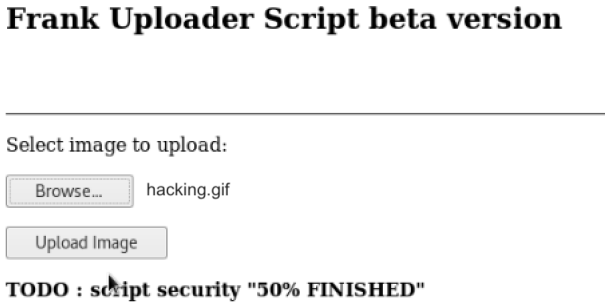

Abbildung 7: *Upload des Exploits für eine Remote-Shell Verbindung*

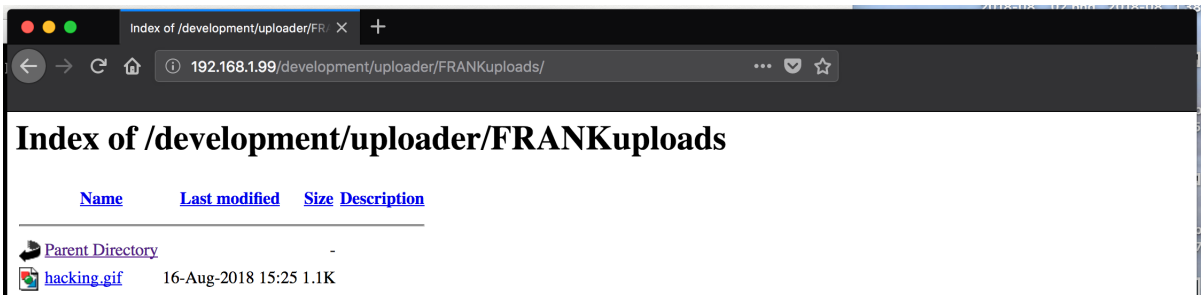

Abbildung 8: *Bestätigung des erfolgreichen Uploads*

Schritt 3 - Remote-Zugriff auf das Zielsystem: Nun kann mit Hilfe des Tools *netcat* auf dem Angreifer-Computer ein TCP-Listener, auf dem Ziel-Port des Remote-Shell-Exploits, platziert werden (siehe Abbildung 9). Da die Pfade der Webadresse den Verzeichnisstrukturen auf dem Server unter dem Verzeichnis */var/www* entsprechen, kann mit Hilfe der LFI-Schwachstelle, wie in Abbildung 10 dargestellt, der Exploit ausgeführt werden. Nach Ausführen des Exploits wird somit vom Opfer aus über den TCP-Listener eine Remote-Shell-Session gestartet. Das Ergebnis ist zunächst der Zugriff auf das System mit der Rolle „www-data“.

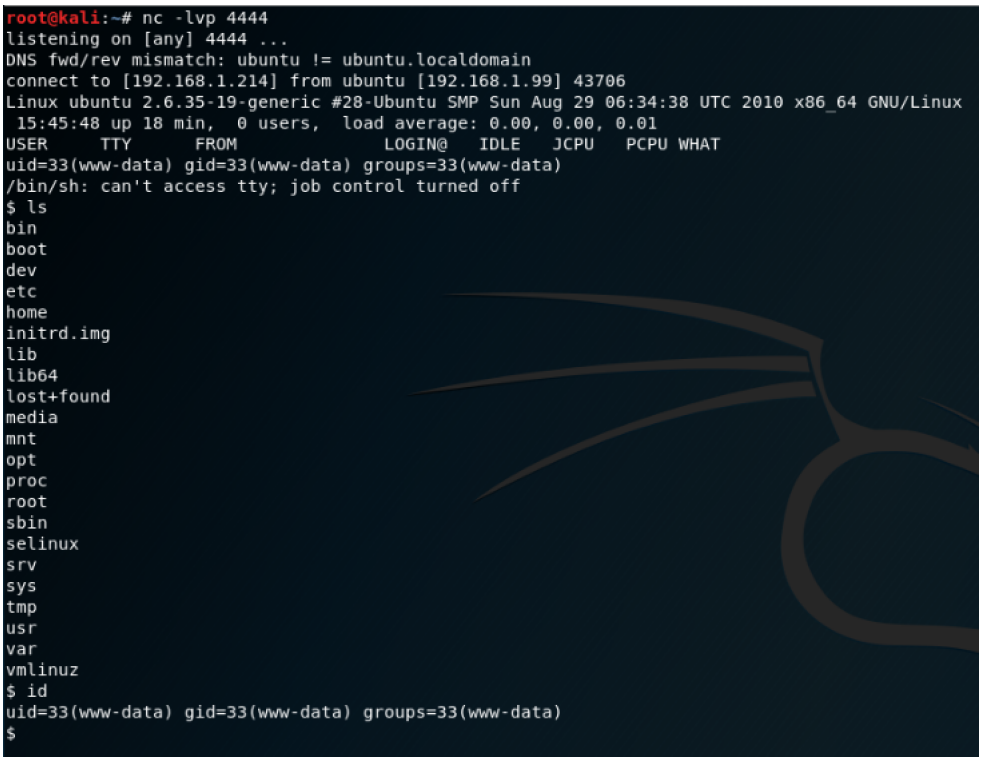

Abbildung 9: *Erstellen eines TCP-Listeners für Port* 4444 *zur Herstellung eines Remote-Shell-Zugriffes*

Abbildung 10: *Ausführen des Exploits zur Herstellung des Remote-Shell-Zugriffs*

Schritt 4 - Erlangen von administrativen Rechten: Mit Hilfe eines Kernel-Exploits[1] kann nun ein Root-Zugriff ermöglicht werden. Hierzu wird vom Angreifer zunächst ein Webserver mit dem Kernel-Exploit zur Verfügung gestellt. Vom Opfer wird nun der Exploit vom Server des Angreifers aus in das Verzeichnis */tmp* heruntergeladen (siehe Abbildung 11). Im Verzeichnis */tmp* kann meist jeder Benutzer Dateien ausführen und bearbeiten, so auch der aktuelle Benutzer der Remote-Shell-Session mit der Rolle „www-data“. Der Kernel-Exploit wird nun ausgeführt und es resultiert der Root-Zugriff (siehe Abbildung 12). Nun besteht die Möglichkeit der Exfiltration des Systems, sowie einer Veränderung der Inhalte des Systems. Es können ebenso jegliche administrativen Veränderungen durchgeführt werden.

[1] Eine Beschreibung des verwendeten Kernel-Exploits für das Ziel-System ist hierbei unter https://github.com/lucyoa/kernel-exploits/tree/master/rds zu finden.

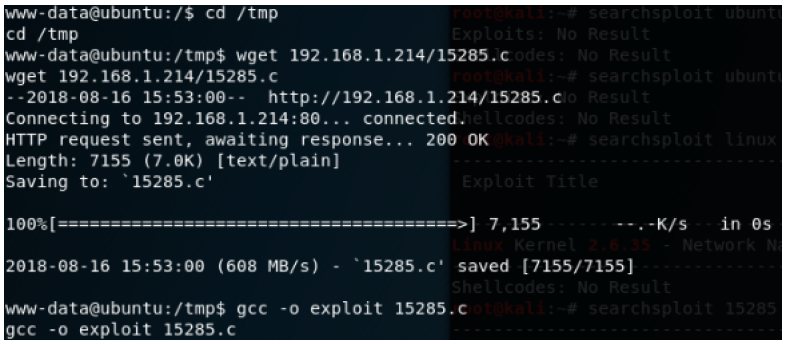

Abbildung 11: *Download des Kernel-Exploits auf den Ziel-Server*

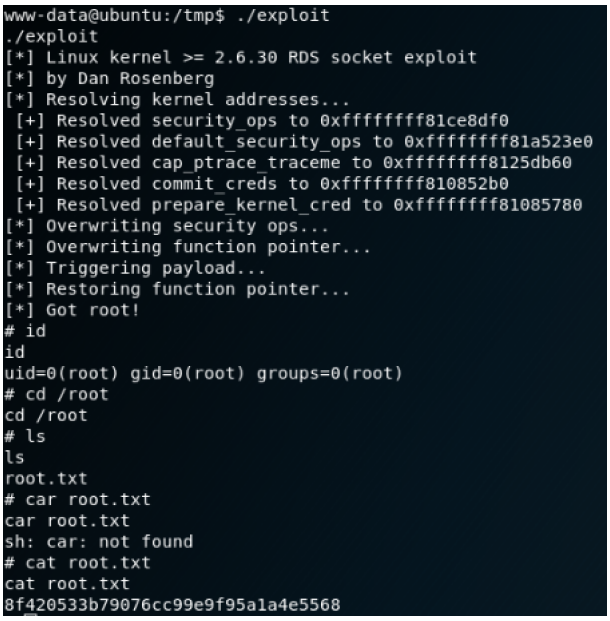

Abbildung 12: *Ausführen des Kernel-Exploits zum Erlangen von Root-Zugriffsrechten*

4 Fazit zur aktuellen Sicherheit des Systems

Zum aktuellen Stand der Sicherheit des Systems lässt sich somit zusammenfassend ein unsicherer Betrieb ermitteln. Die aktuellen Schwachpunkte ermöglichen, wie im Proof-of-Concept in Kapitel 3.2 gezeigt, Angreifern den Zugriff auf das System. Angreifer können dabei sogar administrativen Zugang erhalten und so dem Kunden durch Veränderung der Webseite oder durch Bereitstellen von illegalen Inhalten immensen Schaden zufügen. Dem Kunden wird mit Abschluss des Penetrationstests eine ausdrückliche Behebung aller erkannten Schwachstellen empfohlen. Um das aktuelle Sicherheitsrisiko zu beschränken, wird zu einer sofortigen Deaktivierung der API und des Datei-Uploads geraten.

Anhang

A Abbildungsverzeichnis

B Tabellenverzeichnis