Aufgabe 1: Allgemeine Aussagen zur IT-Sicherheit

(1) Ein verteiltes System bietet insofern eine erhöhte Sicherheit, als dass das System aus mehreren Teilkomponenten besteht, eine Infizierung oder ein Sicherheitsbruch einer Teilkomponente also nicht notwendigerweise alle Teilkomponenten betrifft, was bei einem nicht-verteilten System der Fall wäre.
Ein Nachteil eines verteilten System gegenüber eines nicht-verteilten ist der erhöhte Wartungs-/Installationsaufwand, da mehrere Systeme offensichtlich einen größeren Ressourcenverbrauch bedeuten als einzelne, in sich geschlossene Systeme.

(2) DAU vor dem Bildschirm. Nutzung unsicherer Software/Systeme aufgrund von Komfortabilität jener Software/Systeme (sichere Systeme sind selten einfach und komfortabel benutzbar). Hohe Sicherheitsstandards verursachen häufig hohe Kosten, es existiert also ein Tradeoff zwischen Sicherheit und Machbarkeit/Finanzierbarkeit.

(3a) Da mehr Essen als normalerweise bestellt wird, ist der logische Schluss daraus, dass die Arbeitszeiten länger sind als sonst. Dies würde die Wahrscheinlichkeit von Fehlern durch Überarbeitung und/oder Übermüdung steigern.
Die Angreifbarkeit gegenüber Schadprogrammen oder Hacker aus dem Internet ist also erhöht.
Abgesehen davon könnte auch einer der zahlreichen Lieferanten Spionageabsichten haben. Beispielsweise könnte er versuchen mit einem WiFi-fähigen Gerät Funkverkehr abzuhören oder mit Schadsoftware ausgestattete USB-Sticks in Umlauf zu bringen.
In jedem Fall würde das Schutzziel der Vertraulichkeit also bedroht. Sollte ein Eindringen in das System per Schadsoftware gelingen, wäre ein verändernder Zugriff denkbar, sodass auch die Integrität und Verfügbarkeit als gefährdet einzustufen wäre.

(3b) Hier könnten alle drei Schutzziele beeinflusst werden. Bei einem öffentlichen, nicht verschlüsselten WLAN-Netz wäre es durchaus möglich, einen vorher manipulierten AP mit der gleichen SSID und entsprechender Schadsoftware auszustatten, auf die sich in der Nähe befindliche Clients dann automatisch verbinden.
Den über den AP laufenden Netzwerkverkehr könnte man dann loggen und später auswerten, damit wäre das Schutzziel der Vertraulichkeit verletzt.
Ebenfalls wäre eine Echtzeitmanipulation der Verbindung denkbar, indem man ankommende Daten in einen Cache lädt, diese verändern und erst dann zu dem Zielhost übermittelt.
Auch das Schutzziel der Verfügbarkeit ist gefährdet, da so einzelne Zielhost oder der komplette Datenverkehr blockiert werden könnten.

Aufgabe 2: Schutzziele

(1a) **Anonymität.** Bei der Anonymität können verschiedene Dienste vom Nutzer verwendet werden, ohne das dieser in irgendeiner Art und Weise seine Identität preisgibt. Selbst gegenüber dem Kommunikationspartner, beispielsweise beim Download einer öffentlichen Datei über eine verschlüsselte, nicht protokollierte Verbindung, bleibt die Identität der Nutzer unbekannt. Die Identifizierbarkeit des Nutzers darf also unter keinen Umständen gegeben sein. Eine Verschärfung

der Anonymität stellt die Unbeobachtbarkeit dar.
Unbeobachtbarkeit. Das Konzept der Unbeobachtbarkeit sichert dem Nutzer eines Dienstes oder einer Ressource zusätzlich zu, dass der Datenaustausch gegenüber Dritten unbeobachtbar bleibt, d.h. es kann nicht festgestellt werden, dass überhaupt kommuniziert wurde. Sowohl das Senden, als auch der eigentliche Datentransfer, sowie der Erhalt von Daten sind nicht feststellbar.
Pseudonymität. Die Pseudonymität ist ein schwächeres Konzept im Vergleich zur Anonymität. Der Nutzer ist in seiner Identität gegenüber unbefugten Dritten und dem eigentlichen Kommunikationspartner anonym, jedoch ist der Nutzer über sein Pseudonym bestimmten Personen/Systemen bekannt. EIn Beispiel wäre der Abschluss eines Handels über eine entsprechende Plattform, bei der das Pseudonym des Nutzers als Käufer dem Verkäufer gegenüber bekannt gemacht wird.

(1b) **Vertraulichkeit.** Die Vertraulichkeit gewährleistet die Geheimhaltung von Daten während ihres Übertragungsprozesses zwischen den Kommunikationspartnern, beispielsweise mittels Verschlüsselung. Den Inhalt der Kommunikationsdaten kennen also lediglich die kommunizierenden Parteien.
Verdecktheit. Bei der Verdecktheit werden die zu übertragenden Daten versteckt übermittelt, sodass die Existenz der verdeckt übertragenen Daten ausschließlich den kommunizierenden Parteien bekannt ist. Ein einfaches, analoges Beispiel wäre ein Brief, auf dem zusätzlich zu einer belanglosen Nachricht eine weitere, verdeckte Botschaft mit Geheimtinte geschrieben wurde.

(2a) **Integrität und Zurechenbarkeit.** Die Integrität von Daten beschreibt die Feststellbarkeit jeglicher Manipulation der Daten im Vergleich zu dem Originalzustand. Ist die Integrität von Daten zwischen Kommunikationspartnern gesichert, so kann jedwede Veränderung des Inhalts, also der Daten selbst, von den Kommunikationspartnern festgestellt werden und das unabhängig davon, ob die Daten vom Sender, vom Empfänger oder von einem Dritten verändert wurden.
Bei der Zurechenbarkeit hingegen wird nicht der Zustand einer Datei beschrieben, sondern die Sendung bzw. der Empfang der Daten. Hiermit kann das Verschicken oder das Erhalten einer Nachricht bewiesen werden.

(2b) **Verfügbarkeit und Erreichbarkeit.** Das Schutzziel der Verfügbarkeit beschreibt die Möglichkeit der Nutzung eines bestimmten Dienstes zu jedem Zeitpunkt, an dem der Nutzer den Dienst nutzen möchte. Die Verfügbarkeit stellt damit eine Inanspruchnahme eines Dienstes zu jedem gewünschten Zeitpunkt dar. Die Erreichbarkeit hingegen garantiert nicht die ordnungsgemäße Nutzbarkeit eines Dienstes, sondern lediglich die Erreichbarkeit des Dienstanbieters (Ressourcen, Klienten, Menschen, ...).

(3) **Techniken.** Anonymität kann gewährleistet werden, wenn keine zur Identifizierung dienlichen Nutzerdaten während der Nutzung eines Dienstes protokolliert werden.
Unbeobachtbarkeit ist umsetzbar, wenn weder Nutzer noch Betreiber oder Dritte technisch feststellen können, dass zur Nutzungszeit eines Dienstes überhaupt eine Nutzung stattfindet.
Pseudonymität wird erreicht, wenn keine persönlichen, der Identifizierung dienlichen Daten protokolliert werden, sondern den Nutzern lediglich Pseudonyme zugeteilt werden, dessen Aktionen protokolliert werden.
Vertraulichkeit bei einem Datenaustausch ist realisierbar, wenn nur die Kommunikationspartner Zugriff auf den Inhalt der Daten haben, die Daten müssen dementsprechend verschlüsselt werden

um diesem Anspruch zu genügen.
Um Verdecktheit bei einer Kommunikation umzusetzen muss die eigentliche Nachricht versteckt werden, sodass die verdeckte Nachricht nicht feststellbar ist beim Datenaustausch. Bei analogen Medien wäre beispielsweise Geheimtinte denkbar, bei digitalen Medien das einbetten der Nachricht in ein Bild, in Metadaten, in scheinbar freien Speicherplatz eines Speichermediums, ...
Die Integrität attestiert den Originalzustand von Daten. Dies kann mit einer Hashfunktion gewährleistet werden. Bei Sendung der Daten wird mittels einer Hashfunktion ein Hashwert erstellt, wird später auch nur ein einziges Bit des Inhalts geändert, ändert sich ebenfalls der Hashwert womit die Integrität der Daten verletzt ist.
Die Zurechenbarkeit kann z.B. garantiert werden, wenn beim Senden oder Empfangen von Daten entsprechende Protokolle erstellt und automatisiert gesendet werden.
Die Verfügbarkeit eines Dienstes ist nur gewährleistbar, wenn die Verfügbarkeit regelmäßig überprüft wird. Bei der Verfügbarkeit von Onlinediensten auf Servern beispielsweise werden regelmäßige Pings und andere Überprüfungssignale gesendet und dessen ordnungsgemäße Rücksendung erwartet. Antwortet ein Dienst nicht, ist seine Nicht-Verfügbarkeit sehr wahrscheinlich und entsprechendes Technikpersonal kann benachrichtigt werden. Um trotz erwartbarer Fehlermargen im Jahresmittel eine hohe Verfügbarkeit zu erreichen, können verteilte Systeme genutzt werden. Beispielsweise ist es möglich, zwei Server parallel bereit zu stellen wobei einer als Ausfallsicherung fungiert, die einspringt, sobald der erste Server nicht mehr ordnungsgemäß arbeitet.
Erreichbarkeit kann ebenso wie die Verfügbarkeit durch regelmäßige Überprüfungen und die Parallelbereitstellung sowie Verteilung von Diensten gewährleistet werden.

Aufgabe 3: Angreifermodell

(1) Das Angreifermodell beschreibt die Eigenschaften eines potentiellen Angreifers im Kontext eines worst-case Szenarios, also mit maximalen Möglichkeiten in der spezifischen Situation. Die Ausprägung der Eigenschaften wird also als größtmöglich angenommen, jedoch in Abhängigkeit der maximalen Schutzmechanismen des Systems. Das Angreifermodell beschreibt also die Situation eines starken Angreifers, bei dem die implementierten Schutzmechanismen aber noch Wirkung zeigen. Dies dient zur Charakterisierung der möglichen Angriffsszenarien sowie der Beschreibung der Schutzmechanismen. Die Aufstellung eines Angreifermodells macht insofern Sinn, als dass damit die aktuelle Sicherheitsarchitektur analysiert werden kann. Nach der Aufstellung ist ersichtlich, welche Angriffsszenarien maximal abgewehrt werden könnten, was die Einschätzung der aktuellen Sicherheit erleichtert und i.d.R. Handlungsbedarf respektive Verbesserungsmöglichkeiten aufzeigt, denn für entsprechend entdeckte Defizite kann dann eine Lösung erarbeitet werden Mögliche Systemschwachstellen bzw. mögliche Angriffspunkte werden erkannt und können behoben werden. Zusätzlich dazu können potentielle Risikobereiche identifiziert werden, woduch das Bewusstsein zur besonderen Vorsicht im Umgang mit diesen Bereichen geschärft wird. In einem Angreifermodell werden zunächst die Rollen des Angreifers abgebildet, dies umschreibt das Verhaltensmuster des Angreifers. Mögliche vorstellbare Ausprägungen wären Benutzer eines Dienstes, Betreiber eines Dienstes, Wartungsarbeiter eines Dienstes, Außenstehender mit Zugriff auf das Netzwerk usw. Kombinationen aus mehreren Ausprägungen sind denkbar. Hinzu kommt die Verbreitung des Angriffs, dies umschreibt die angreifbaren Lokalitäten des Systems, dies sind alle Stellen an denen der Angreifer das System manipulieren oder ausspähen kann. Des Wei-

teren wird das Verhalten des Angreifers beschrieben, welches passiv oder aktiv sein kann. Ein Ausspähen von Daten des Systems, also ein beobachtendes Verhalten, kann sowohl passiv als auch aktiv geschehen. Ein manipulatives Verhalten, also ein veränderndes Verhalten, kann nur aktiv passieren. Als letztes wird die Rechenkapazität des Angreifers genannt. Angenommen der Angreifer hätte unendliche Rechenkapazitäten zu Verfügung, dann wären herkömmliche Schutzmaßen nutzlos. Dies wird auch als informationstheoretischer Anfgriff bezeichnet. Die eher an der Realität orientierte Alternative ist bei dem Angreifer beschränkte Ressourcen zu vermuten, dies wäre ein komplexitätstheoretischer Angriff. Bei einem solchen Angriff hat der Angreifer endliche Rechenkapazitäten, sodass z.B. sehr komplexe Passwörter als sicher angenommen werden können, da die durchschnittliche Zeit zum Knacken der Passwörter bei ausreichender Komplexität schnell Hunderte oder Tausende von Jahren betragen kann.

Aufgabe 4: Passwortsicherheit

(1) Würden z.B. Passwörter von Kunden beispielsweise in einer Datenbank in Klartext abgelegt werden, so könnte zum einen der Administrator der Datenbank die Passwörter auslesen und zum anderen könnte ein potentieller Angreifer, der die in der Datenbank hinterlegten Daten kopiert, die Passwörter direkt auslesen.
Beide Szenarien können durch das hashen von Passwörtern vermieden werden.
Im einfachsten Fall wird ein Kennwort bei Erstspeicherung vom Server empfangen und direkt gehasht sowie beispielsweise in einer Datenbank (gehasht) gespeichert. Zukünftig, wenn der Nutzer sich ernaut authentifzieren möchte/muss, wird von seinem Passwort erneut der Hashwert gebildet und mit dem in der Datenbank gespeicherten Hash verglichen. Bei Übereinstimmung ist gewährleistet, dass das Passwort korrekt angegeben wurde.
Dieses Verfahren ist sicherer, weil so zum größten Teil der Zeit nur mit den Hashwerten des Passworts gearbeitet wird und da aus einem Hashwert nicht ohne beträchtlichen Aufwand (teilweise sogar nahezu unmöglich) der Originalwert errechnet werden kann, ist das Passwort sicher unter Verschluss.
Der verwendete Hash-Algorithmus bei dem Passwort von „leroy“ ist MD5 und das Passwort zu dem Hashwert „06e2b745f3124f7d670f78eabaa94809“ ist „hund“. Hashverfahren sind zwar irreversibel, man kann aber (beispielsweise über Rainbow-Tables) nach dem zugehörigem Urbild für ein entsprechendes Bild suchen.

(2) Nehmen wir die „Standardzeichen“ für so einen Alphanumerischen Zeichensatz an:
$A-Z$, $a-z$, $0-9$, so erhalten wir 62 mögliche Zeichen. Da bekannt ist, dass es sich maximal um ein achtstelliges Passwort handeln kann, ergeben sich so $62^8 = 218.340.105.584.896$ mögliche Passwortkombinationen.
Gegeben ein Passwort-Cracking-Tool mit einer Leistung von einer Million Passwörtern pro Sekunde würde die Entschlüsselungszeit eines solchen Passworts maximal gerundet $\frac{218.340.105.584.896}{1.000.000} = 218.340.105,58$ Sekunden dauern. Das entspricht etwa 60.650 Stunden, bzw. ca. 2.527 Tagen oder auch gerundet 6,92 Jahren.
Bei Ignorierung der Groß-/Kleinschreibung reduzieren sich die möglichen Zeichen auf 36, was bei einem maximal achtstelligem Passwort maximal $36^8 = 2.821.109.907.456$ Kombinationen ergeben.

Damit würde sich die Entschlüsselungszeit auf gerundet $\frac{2.821.109.907.456}{1.000.000} = 2.821.110$ Sekunden verringern. Das entspricht etwa 784 Stunden, bzw. ca. 33 Tagen.

(3) Bei herkömmlichen Angriffen wie z.B. dem reinen bruteforcen hängt das Knacken eines Passworts rein von der Zeitkomplexität ab. Bei Rainbow-Tables werden vorkalkulierte Tabellen von Klartextpasswörtern und dazugehörigen Hashwerten benutzt, um die Zeitkomplexität drastisch zu verringern. Allerdings müssen derartige Tabellen relativ groß sein, um gute Ergebnisse beim Passwortknacken zu ermöglichen. Deshalb spricht man bei Rainbow-Tables von einem Time-Memory-Trade-Off, denn größere Tabellen (größere Speicherplatzkomplexität) spart viel Berechnungszeit bei Angriffen, andererseits dürfen die Tabellen nicht zu groß sein, da sie sonst RAM und/oder das permanente Speichermedium zu stark belasten. Es muss also ein „günstiger" Tradeoff zwischen Zeit und Speicherplatz gefunden werden, der die Zeitkomplexität ausreichend reduziert, die Speicherplatzkomplexität aber ebenfalls in bewältigbaren Maßen hält.
Die Idee von Rainbow-Tables ist mit einem zufälligen Klartextpasswort anzufangen, davon den Hash-Wert zu berechnen, diesen mit einer beliebigen Reduktionsfunktion zurück zu rechnen zu einem Klartextpasswort, dieses erneut zu hashen usw.
Typischerweise wird dieser Vorgang z.B. 10.000x wiederholt. Start- und Endwerte werden gespeichert. Bei einem Angriff können diese erzeugten Ketten nun durchlaufen werden und nach einem Hashwert durchsucht werden.
Der Name Rainbow-Tables kommt daher, weil (um Kollisionen zu vermeiden), ein bewährtes Verfahren ist verschiedene Reduktionsfunktionen zu benutzen. Man kann sich also vorstellen, dass im ersten Schritt die rote Reduktionsfunktion genutzt wird, im zweiten die gelbe usw.

(4) Ein Salt macht die Generierung von Rainbow-Tables unwirtschaftlich, da die Zeit- und Platzkomplexität enorm steigt. Da das Passwort vor dem Hashen mit einem Salt (idealerweise zufällig berechnet) versehen wird, müsste man für jedes solche Passwort, das in einer Rainbow-Tabelle vorkommen soll, nun alle möglichen Salt-Kombinationen nutzen. Es müsste also für jeden möglichen Saltwert eine eigene Rainbow-Table erstellt werden.

(5) Das gesuchte Passwort ist „sonne". Zuerst wurde ein deutschsprachiges Wörterbuch heruntergeladen. Das Programm wurde in Python geschrieben. In einer for-Schleife wird jede Zeile des Wörterbuchs eingelesen. Zuerst wird geprüft, ob das Wort länger als 5 Zeichen ist. Ist diese Überprüfung positiv, wird die aktuelle Iteration sofort übersprungen. Danach wird geprüft, ob das Wort Großbuchstaben enthält, ist dies der Fall, wird die aktuelle Iteration sofort übersprungen. In allen anderen Fällen wird an den Wert des Salts das aktuelle Wort konkateniert, sodass man z.B. „xohth4dew5p8baum" für das Wort „baum" erhält. Dann wird von diesem Wert (Salt+Wort) der MD5-Hash berechnet und verglichen mit dem gegebenen Hashwert „199f066a0bac4140e792d1d4a434ae44". Die Schleife läuft so lange durch, bis entweder der aktuell errechnete Hash (Salt+Wort) mit dem gegebenen Hash übereinstimmt und damit das gesuchte Passwort gefunden wurde oder bis das Wörterbuch ein Mal komplett durchlaufen wurde.
Zum Ermitteln des Passworts benötigte das Programm 0.26s.
Wäre das Salt im Vorfeld nicht bekannt, so müsste man einen zufälligen Salt errechnen und für jeden solchen errechneten Salt dann ein Mal das komplette Wörterbuch durchlaufen. Das würde die Zeitkomplexität enorm erhöhen.