GSS-Übungsblatt 1

Alexander Timmermann

1 Allgemeine Aussagen zur IT-Sicherheit

1. „Ein verteiltes System ist eine Menge voneinander abhängiger Computer, die dem Benutzer wie ein einzelnes, kohärentes System erscheinen."[1]

   Beispiele sind unter Anderem das Internet, ein internes Netzwerk wie z.B. das des Informatik-RZ oder ein verteiltes soziales Netzwerk wie Diaspora.

2. Durch den Charakter eines verteilten Systems entstehen natürlicherweise mehr Angriffspunkte als bei einem zentralisierten System, da viele Computer koexistieren und jeder davon ein Einfallstor sein kann. Weiterhin gibt es offene Übertragungswege, die als Angriffspunkt dienen können. In verteilten Systemen kann es oft auch mehrere Betreiber geben, unter denen es jedoch bei einem Angriff oft nur zu mangelhafter Kommunikation kommt.

   Bei verteilten Systemen ist aber nicht immer gegeben, dass man mit einem Angriff bzw. mit einem Eindringen in eine der Komponenten auch gleich Zugriff auf alle gewünschten Daten hat, da diese möglicherweise von einem anderen System verwaltet werden.

3. (a) **Mangelnde Kompetenz der Nutzer**: Oft sind gerade Nutzer, die nicht technisch versiert sind, nicht in der Lage, z.B. eine Phishing-Attacke als solche zu erkennen. Dadurch kann das Unternehmen kompromittiert werden.

   (b) **Komfort vs. Sicherheit**: Sichere Software bietet (leider) selten eine komfortable User Experience. Es ist jedoch mittlerweile auch in nicht-technischen Bereichen unerlässlich, Software einzusetzen. Diese muss dementsprechend intuitiv zu bedienen sein.

   (c) **Kostenfaktor vs. Sicherheit**: Industrielle Software, die strikte Sicherheitsstandards einhält, ist oft teuer. Viele Unternehmen erkennen den Wert solcher Software nicht an und greifen zu einer kostengünstigeren und oft unsichereren Lösung.

2 Schutzziele

1. (a) **Anonymität, Pseudonymität und Unbeobachtbarkeit**
   Die Begriffe beschreiben Abstufungen der Verschleierung der Identität eines Individuums und/oder seiner Handlungen. Anonymität ist dabei die stärkste Ausprägung, da hier vom Individuum nichts bekannt ist. Bei Pseudonymität ist wenigstens ein von der Person gewähltes Pseudonym bekannt, z.B. ein Username. Unbeobachtbarkeit hingegen bezeichnet nur den Schutz der Aktivitäten vor dem Einblick Dritter, d.h. dass z.B. das Mitlesen von Nachrichten ausgeschlossen wird. Ein Schutz der Identität ist hierbei nicht notwendigerweise eingeschlossen.

[1] Tanenbaum, van Steen: Verteilte Systeme. Pearson, 2. Auflage, 2008

(b) **Vertraulichkeit und Verdecktheit**
Diese Begriffe umfassen die Inhalte von Kommunikationsdaten. Bei Vertraulichkeit wird dieser Inhalt geschützt und kann nur von den Beteiligten eingesehen werden. Bei der Verdecktheit werden die Inhalte zusätzlich so übertragen, dass von Dritten nicht einmal die Existenz von vertraulichen Inhalten erkannt werden kann.

2. (a) **Integrität und Zurechenbarkeit**
Beide Begriffe umfassen das Senden und Empfangen von Daten. Während das Schutzziel der Integrität jedoch gewährleistet, dass Daten nach dem Senden weder vom Sender, noch von Dritten unbemerkt geändert wurden, weist die Zurechenbarkeit nur nach, ob bzw. dass Daten gesendet und empfangen wurden. Als Gegenteil dazu kann die *plausible deniability* (glaubwürdige Abstreitbarkeit) angesehen werden.

(b) **Verfügbarkeit und Erreichbarkeit**
Beide Begriffe beschreiben den ordnungsgemäßen Betrieb von verteilten Systemen. Während Verfügbarkeit jedoch die Bereitschaft zur Benutzung des Systems beinhaltet, bezeichnet die Erreichbarkeit nur, dass alle Komponenten ansprechbar sind, nicht ob sie auch korrekt zusammenarbeiten.

3. **Anonymität:** Als Betreiber einer Plattform kann man Anonymität gewährleisten, indem man keine Identifizierung fordert und auch keine Daten aufzeichnet, die zur Identifizierung genutzt werden könnten, z.B. Geodaten aus der IP-Adresse. Als Nutzer kann man sich hier nur teilweise mittels Software wie Tor oder VPNs schützen.

Pseudonymität: Pseudonymität ist gewährleistet, wenn man als Betreiber keine Nutzer mit ihrer Identität identifiziert, sondern mit einem Pseudonym. Persönliche Daten dürfen dazu nicht erfasst werden.

Unbeobachtbarkeit: Unbeobachtbarkeit ist gegeben, wenn man auch als Betreiber keine Daten mitlesen kann. Dazu können Daten beispielsweise clientseitig verschlüsselt werden.

Vertraulichkeit: Vertraulichkeit kann z.B. mittels einer Verschlüsselung mit Public-Private-Key-Architektur umgesetzt werden. Am Beispiel eines Chat-Anbieters kann dann schon im Client verschlüsselt werden, die Übertragung geschieht dann geschützt vor dem Einblick Dritter.

Verdecktheit: Zur Verdecktheit müssen die Daten, die ausgetauscht werden sollen, so versteckt werden, dass sie im Normalfall nicht als vertrauliche Daten erkannt werden. Hierzu kann man sich beispielsweise der Steganografie bedienen und die Daten in einem Bild verstecken. Dies ist jedoch höchst selten praktikabel.

Integrität: Zur Sicherstellung der Integrität kann man beispielsweise Checksummen verwenden. Dazu wird ein Checksummen-Algorithmus benutzt und die resultierende Checksumme mit den Daten versandt. Wird nun etwas an den Daten verändert, ändert sich auch die Checksumme.

Zurechenbarkeit: Durch serverseitiges Loggen kann nachgewiesen werden, wann und ob Daten versandt wurden. Natürlich funktioniert dies nur so lange es überhaupt einen Server gibt (also kein P2P-Netzwerk) und der Server nicht kompromittiert ist.

Verfügbarkeit: Die Verfügbarkeit eines Systems kann immer nur nach bestem Wissen und bei regelmäßiger Überprüfung gewährleistet werden. Zum Zwecke der Überprüfung kann man jedoch eine Monitoring-Software einsetzen, die bei einem Ausfall alarmiert und ggf. bereits Schritte zur Behebung einleitet. Zur Sicherstellung eines möglichst ausfallfreien Betriebs kann man das System verteilen und in mehreren Aspekten redundant aufbauen (geografisch, funktional,...).

Erreichbarkeit: Ebenso wie Verfügbarkeit kann Erreichbarkeit nie garantiert werden. Durch Überwachungssysteme kann man einen Ausfall jedoch meist schnell feststellen und beheben. Zur Sicherstellung eines möglichst ausfallfreien Betriebs kann man das System verteilen und in mehreren Aspekten redundant aufbauen (geografisch, funktional,...).

3 Angreifermodell

1. Als Angreifermodell bezeichnet man ein Modell, über das man die Stärke und Wirksamkeit eines Schutzmechanismus definieren kann.

   „Das Angreifermodell definiert die maximal berücksichtigte Stärke eines Angreifers, gegen den ein Schutzmechanismus gerade noch wirkt."[2]

   Es kann benutzt werden, um einen existierenden Schutzmechanismus einzuordnen, und gibt ggf. auch Hinweise auf Verbesserungsmöglichkeiten.

   Der Angreifende wird hierbei in verschiedenen Kategorien modelliert:

   Rolle: Wie viel Informationen über und Zugriff auf das System steht dem Angreifenden zur Verfügung? Bspw. kann davon ausgegangen werden, dass ein Wartungsdienst wesentlich mehr Informationen und Zugriff besitzt als ein normaler Benutzer (vgl. Insider ↔ Outsider).

   Verbreitung: An welchen Stellen kann der Angreifende Informationen abgreifen oder verändern? Im kleinsten Fall hat ein Agitator nur eingeschränkte Benutzerrechte und kann nur wenige Informationen lesen oder verändern. In einem unwesentlich schlimmeren Fall hat ein Angreifer beispielsweise Zugriff auf allen Traffic an einem Internet Exchange Point (IXP) eines Landes (in Deutschland wäre das das DE-CIX in Frankfurt).

   Verhalten: Verhält sich ein Angreifer aktiv oder passiv? Sammelt er nur Daten, bspw. durch Mitschneiden von Netzwerktraffic, oder werden aktiv Daten zerstört?

   Rechenkapazität: Besonders bei kryptographischen Schutzmechanismus ist bedeutend, über wie viel Rechenkapazität ein Angreifer verfügt. Diese Schutzmechanismen beruhen auf Problemen, die sich mit „normaler", d.h. kommerziell erschwinglich verfügbarer Hardware, nicht lösen lassen. Verfügt ein Angreifer jedoch über eine große Rechenkapazität ist ein Schutz nicht notwendigerweise gewährleistet. Besonders wenn der Angreifer ein Staat bzw. eine Behörde ist (wie z.B. die *National Security Agency* der USA oder der Bundesnachrichtendienst) kann von einer enormen Rechenleistung ausgegangen werden.

2. TODO

4 Angriffsformen

1. Wenn wir einem der Lieferdienste Spionageabsichten unterstellen, könnte das Schutzziel der Vertraulichkeit kompromittiert werden. Beispielsweise könnte ein feindlicher Agent einen Datenträger mit einem Trojaner einschleusen. Sollten dann auch Sabotageabsichten hinzukommen, könnte z.B. auch Schadcode eingeschleust werden, der Daten zerstört und damit den ordnungsgemäßen Betrieb bzw. das Schutzziel der Verfügbarkeit stört. Werden von diesem Schadcode zum Zwecke der Sabotage Daten aktiv verändert ist auch das Schutzziel der Integrität betroffen.

[2] Folien zur 1. Vorlesung, Seite 26

2. Auch hier ist eine Brechung aller drei Schutzziele denkbar. Bei einem normalen WPA-Netzwerk gibt es keinerlei Authentifizierung der APs, man kann also (solange man das Passwort des ursprünglichen Netzes kennt) einfach ein Netzwerk mit der gleichen SSID und dem gleichen Passwort erstellen, und solange der eigene AP der leistungsstärkere ist, werden sich alle Clients in Reichweite verbinden. Danach kann man Traffic mitschneiden (Vertraulichkeit), teilweise manipulieren (Integrität) oder auch bestimmten Traffic blockieren (Verfügbarkeit).

5 Passwortsicherheit

1. TODO

2. Wenn Passwörter im Klartext gespeichert werden, sind sie für alle ersichtlich, die Zugriff auf diese Datenbank haben, sei dieser Zugriff legitim oder illegitim. Bei einem massiven Leak würden Passwörter dabei ungeschützt verloren gehen. Über die Hälfte aller Internetnutzer benutzt weniger als 5 Passwörter in ihrem gesamten Online-Leben[3], weshalb mit hoher Wahrscheinlichkeit auch weitere Accounts bei unbeteiligten Services kompromittiert würden.

   Bei Benutzung einer kryptografischen Hashfunktion wird eine Hashfunktion benutzt, die sich nicht umkehren lässt, d.h. aus dem berechneten Hashwert lässt sich der Ausgangstext nicht wiederherstellen. Benutzt man dieses System zur Kennwortspeicherung, so wird beim Setzen des Passworts der Hashwert berechnet und in einer Datenbank gespeichert. Versucht der Benutzer sich nun mit einem Passwort anzumelden, wird davon ebenfalls der Hashwert berechnet. Sind beide Hashwerte gleich hat der User das korrekte Passwort eingegeben.

3. TODO

4. Durch das Hinzufügen eines Salts wird der Hashwert in unvorhersehbarer Weise verändert. Man müsste also die Rainbow-Table für einen spezifischen Salt neu berechnen. Wird nun für jedes Passwort ein zufälliger Salt gewählt, wird die Rainbow-Table praktisch nutzlos.

5. Als Wortliste wird das deutsche Wörterbuch von *GNU Aspell* benutzt, das (ggf. nach Installation) unter `/usr/share/dict/ngerman` zu finden ist. Das Programm wurde in Ruby geschrieben und ist sehr kurz:

```ruby
require 'digest'

start = Time.now

# Wörterbuch laden
dictfile = open('/usr/share/dict/ngerman')
dict = dictfile.read.split("\n")

# Wörter mit >5 Buchstaben herausfiltern
dict.reject! { |word| word.length > 5 }

# Die vorgegebenen Werte
salt = 'xohth4dew5p8'
hashval = '199f066a0bac4140e792d1d4a434ae44'
```

[3] TeleSign Consumer Account Security Report, 2015, https://www.telesign.com/wp-content/uploads/2015/06/TeleSign-Consumer-Account-Security-Report-2015-FINAL.pdf

```ruby

# Wort für Wort durch das Wörterbuch iterieren
dict.each do |word|
    # diese Iteration überspringen wenn der Hash nicht passt
    next unless Digest::MD5.hexdigest("#{salt}#{word}") == hashval
    # Wenn der Hash passt: Zeit stoppen, Ergebnis ausgeben
    stop = Time.now
    puts "Password is \"#{word}\""
    puts "Took #{stop - start} seconds"
end
```

Durch die Kommentare im Code sollte die Funktionsweise klar sein. Führen wir es aus, so erhalten wir als Ergebnis:

```
Password is ‘‘sonne’’
Took 0.102220636 seconds
```

Das Programm hat das Passwort also schon nach etwa $0,1$s geknackt. Wenn wir die Zeit erst nach Einlesen Wörterbuchs starten, so dauert es sogar nur $0,009$s.

Wenn uns der Salt nicht bekannt wäre, so müssten wir für jedes Wort auch jeden Salt ausprobieren. Die Komplexität steigt damit stark an. Wenn zusätzlich auch keine Constraints vom Salt bekannt sind (z.B. max. 12 Stellen, alphanumerisch, usw.), ist es praktisch unmöglich, den Hash zu knacken.