Summary

Kurs 1867
Sicherheit im Internet 2

Inhaltsverzeichnis

1. Angriffe auf Rechner oder Netze

1.1 Einführung: siehe Skript.

1.2 Sniffing

Sniffer sind Systeme die den Datenverkehr auf einem Netz mitlesen. Dafür wird die Ethernet-Karte im *promiscuous mode* betrieben, so dass alle Pakete angenommen werden, auch wenn sie nicht an die Karte adressiert sind. Ein *Sniffer* Programm (z.B. *ethereal*) protokolliert die Pakete in einer Logdatei mit und unterstützt die Analyse der Pakete nach bekannten Protokolltypen. Damit können z.B. bei *ftp*-Anmeldungen Informationen über Benutzerkennung und Passwort (wird bei ftp im Klartext übertragen) aus den protokollierten Paketen herausgelesen werden.
Diese Angriffsart funktioniert nur, wenn der Angreifer Zugriff auf das Übertragungsmedium hat, also z.B. an einem Hub angeschlossen ist oder WLAN abhören kann. Durch die Verwendung eines Switches anstatt des Hub kann der Angriff im LAN verhindert werden.

1.3 Spoofing

Darunter versteht man das Vortäuschen einer falschen Identität (Maskierungsangriff). Dabei kann eine falsche IP-Adresse oder ein falscher DNS-Name vorgetäuscht werden.

1.3.1 IP Spoofing

Mit Administratorrechten kann problemlos die IP-Adresse eines Rechners geändert werden. Ein weitere Möglichkeit ist, dass ein Angreifer gefälschte ARP Pakete als Antwort auf einen ARP-Request absetzt oder dass der Angreifer sich als Router ausgibt. Der Angreifer muss dann aber die abgefangenen Pakete aber auch weiterleiten, da sonst wegen der verlorenen Pakete der Absender den Angriff bemerken könnte.

1.3.2 DNS Spoofing

Ein DNS-Server verwaltet zwei Datenbanken:

- ***forward zone*** für die Abbildung der Namen auf IP Adressen
- ***reverse zone*** für die Abbildung von IP Adressen auf Namen

Das Design von DNS-Servern verlangt keine Maßnahmen zur Konsistenzprüfung dieser Datenbanken und es findet auch i.d.R. keine sichere Authentisierung bei der Erstellung oder Veränderungen dieser Tabellen statt.
Probleme durch gefälschte oder inkonsistente Einträge:

1. Unter UNIX kann man vertrauenswürdige Rechner (über deren DNS-Namen) definieren und damit einen Verbindungsaufbau von diesen Rechnern ohne explizite Authentifizierung erlauben (*rlogin* und *rsh* nutzen dies). Hat ein Angreifer nun die ***reverse zone*** Datenbank manipuliert, so kann er einem Opferrechner den Namen eines vertrauenswürdigen Rechners vortäuschen, obwohl er seine eigene IP-Adresse unverfälscht lässt. Aufgrund der Manipulation der reverse zone löst der DNS aber die echte IP-Adresse des Angreifers mit dem Namen des als vertrauenswürdig definierten Rechners auf. Der Opferrechner prüft aber nur den DNS-Namen des Rechner und erlaubt die Verbindung.

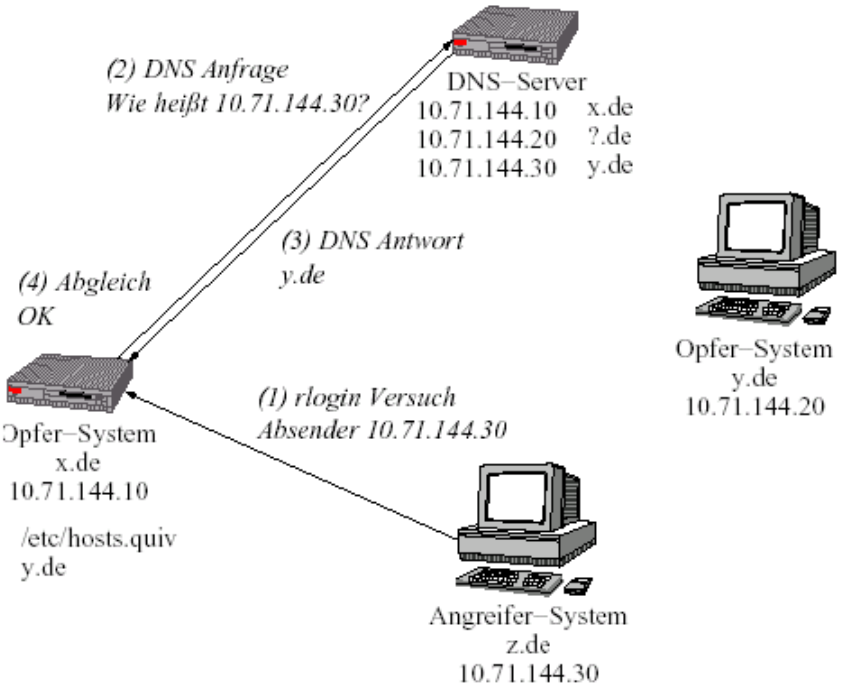

Abb. 1.1: Angriff auf reverse zone

2. Manipuliert ein Angreifer die ***forward zone*** Tabelle, so kann er fälschlicherweise das Ziel von Verbindungsaufbauwünschen werden. So könnte der Angreifer eine komplette Website auf dem eigenen Rechner nachbauen, z.B. einer Bank od. einer e-Commerce Firma. Dieser Angriff kann drei Schadenstypen zur Folge haben:
   a. ***Reputationsschaden:*** Verunglimpfung durch Änderung der Inhalte
   b. ***Umsatzschaden:*** e-Commerce Umsatz kann verhindert werden
   c. ***Vertraulichkeitsschaden:*** Bei Eingabe von Benutzerkennungen auf der Website können z.B. bei e-Banking PIN und TAN ermittelt werden

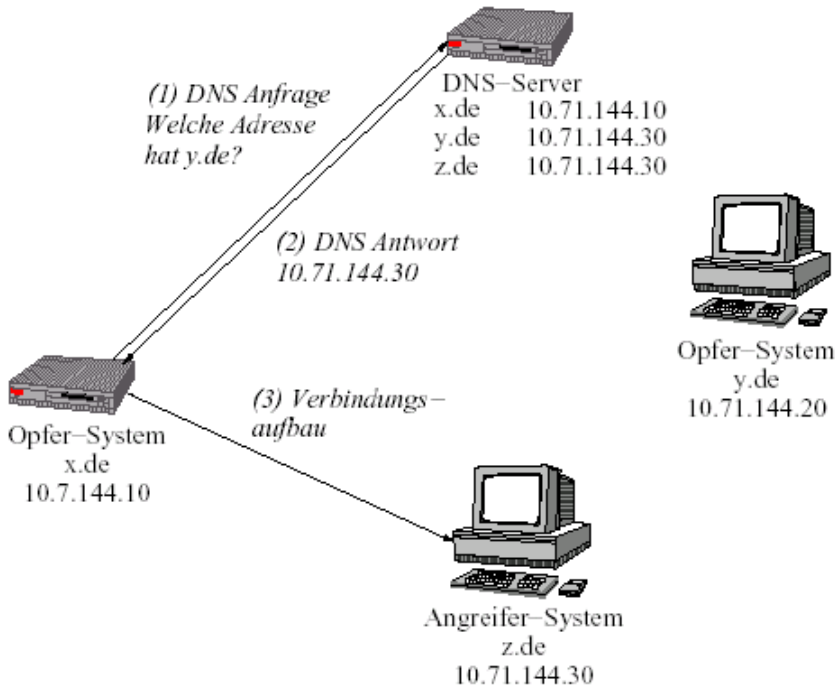

Abb. 1.2: Angriff auf forward zone

1.4 Wörterbuchangriffe

Dabei wird versucht, das Passwort eines legitimen Benutzers durch systematisches Probieren heraus zu finden. Dabei gibt es 2 Strategien:

1. Ausprobieren von bekannten Wörtern und ihren Abwandlungen bzw. Kombinationen. Diese Methode setzt darauf, dass Benutzer ein leicht merkbares Passwort wählen.
2. Ausprobieren aller möglichen Zeichenketten (*brute force*).Es werden alle möglichen Zeichenketten generiert und geprüft. Enthält ein Alphabet *x* Zeichen und ist das Passwort *n* Zeichen lang so ergeben sich x^n mögliche Kombinationen. Bei langen Passwörtern scheitert diese Methode am zu großen Zeitaufwand.

Bei beiden Methoden wird der Hash-Wert des zu prüfenden Wortes generiert und mit dem gespeicherten Hash-Code des zu knackenden Passwortes verglichen.
Möglichkeiten um an die Hash-Werte der Passwörter zu gelangen:

- *WindowsNT*:
  1. Auszug aus der Registry mit einem Hilfsprogramm erstellen. Dafür sind Admin-Rechte u. phys. Zugang notwendig. Ab SP3 und SYSKEY nicht mehr möglich.
  2. Auszug aus SAM-Datei. Datei ist während des Betriebs lesegeschützt, es geht also nur über den Umweg einer Systemsicherung durch das OS.
  3. Abhören des Netzverkehrs. Die Prüfung des Passworts wird bei einem Win-Client vom Domain Controller durchgeführt. Die Übertragung des Passworts kann dabei verschlüsselt oder unverschlüsselt erfolgen.
- *UNIX*: Hash-Werte bei alten Systemen aus `/etc/passwd` auslesen. Bei neueren Version nicht mehr möglich, da sie in der lesegeschützten `shadow`-Datei liegen.

1.5 Buffer Overflow Angriffe

Durch einen buffer overflow können verschiedene Fehler auftreten:

- ***Programmabsturz:*** der überschriebene Speicherbereich enthält keinen gültigen Programmcode mehr.
- ***Ausführen anderen Programmcodes:*** Ein Angreifer platziert durch einen buffer overflow seinen eigenen Programmcode im überschriebenen Speicherbereich.
- ***Modifikation von Daten:*** An der Stelle des Speicherüberlaufes können auch Daten stehen die so manipuliert werden.

Dabei wird der Speicheraufbau in prozeduralen Sprachen ausgenutzt:
Damit ein Angreifer eigenen Programmcode zur Ausführung bringen kann, muss er zwei Dinge erreichen:

1. Er muss den Programmcode in den Adressraum des Programms kopieren. Hierfür bietet sich der Speicher für lokale Variablen am *Stack* an.
2. Die Rücksprungadresse muss auf den Speicherbereich zeigen, in den der Angreifer seinen Code kopiert hat.

Dafür muss der Angreifer den Prozessortyp und den Maschinencode des anzugreifenden Systems kennen. Unter UNIX auf Intel-CPUs kann man z.B. den Code zum Starten einer Shell in 50 Bytes unterbringen!
Die einzige Gegenmaßnahme ist qualitativ besser Code mit weniger potentiellen *buffer overflow* Fehlern.

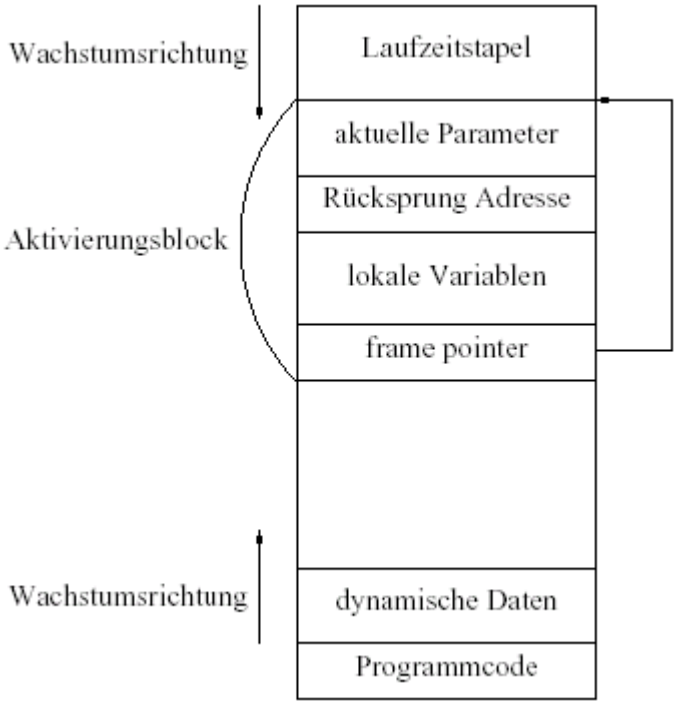

Abb. 1.3: Hauptspeicheraufbau in prozeduralen Sprachen

1.6 URL hacking

Bei diesem Angriffstyp wird eine falsche Identität bzw. eine falsche Absicht erzeugt. Z.B. werden kleine Fehler bei der Eingabe der URL ausgenutzt, indem eine URL mit einem leicht zu verwechselnden Namen registriert wird wie z.B. `wwwgmx.com` .
Um möglicht viele Besucher auf die eigene Seite zu locken kann auch dadurch erreicht werden, dass die Suchmaschinen manipuliert werden, indem bestimmte Schlüsselwörter, die für das Suchergebnis ausschlaggebend sind, auf der eigenen Seite entweder in Meta-Elementen oder im Text (auch unsichtbar) versteckt werden.

1.7 Spezielle Hacker Tools

Hilfsprogramme mit denen die Sicherheit eines Systems getestet werden können. Dabei gibt es 3 wesentliche Klassen:

- Host Scanner
- Network Scanner
- Intrusion Scanner -> siehe Abschnitt 3.4

1.7.1 Host Scanner

Untersucht die Konfiguration eines Rechners auf Schwachstellen:

Zugriffsrechte von Dateien:

Typisches Problem ist dabei ein Leserecht für zu viele Benutzer. (z.B. Passwortdatei)

Besitzer von Dateien:

- Probleme durch Zuordnung einer Datei an einen falschen Besitzer oder Gruppe.
- Ein gesetztes *SUID*-Bit bei ausführbaren Dateien bewirkt, dass die Datei mit den rechten des Besitzer der Datei ausgeführt wird, egal wer sie startet. Ein Angreifer könnte sein Programm mit dem *SUID*-Bit versehen und es dem `root` zuordnen. Damit

würde das Schadprogramm dann immer mit `root`-Rechten ausgeführt und könnte so maximalen Schaden anrichten.

Unbefugte Modifikationen:

Ein Host Scanner kann auch die Prüfung der Integrität der Daten durchführen, z.B. mit Hilfe digitaler Signaturen (Hash-Werte).

Inhalte von Konfigurationsdateien:

Bei der Konfiguration spezieller Systemsoftware werden oft Standard-Benutzer mit einem Standard-Passwort automatisch angelegt. Dies sollte natürlich sofort nach Installation geändert werden.

Versionen der Software:

Ein Host Scanner kann prüfen ob Programme mit bekannten Schwachstellen auf dem System installiert sind.

Beispiele für Host Scanner:

COPS – Computerized Oracle and Password System:

(von Farmer u. Spafford) COPS ist eine Sammlung von Shell-Skripten oder Perl-Programmen, wobei jedes dieser Programme einen bestimmten Problembereich untersucht. Es werden dabei aber gefundene Probleme nicht gelöst sondern nur protokolliert. Das Hilfsprogramm CARP (COPS Analysis and Report Program) unterstützt den Benutzer bei der Analyse der Ergebnisse. Folgende Problembereiche werden von COPS überprüft:

- ***Zugriffsrechte*** auf Dateien, Verzeichnisse und Devices
- ***Inhalte, Formate und Sicherheit*** der Passwort- und Gruppendateien
- ***Programme und Dateien*** in `/etc/rc*`
- ***Existenz und Zugriffrechte*** von Programmen mit gesetztem SUID-Bit
- ***Schreibrechte*** in den Benutzer Home-Verzeichnissen
- ***Konfiguration*** des ftp-Dienstes (ist anonymous ftp erlaubt)
- ***CERT*** Hinweise und ob alte Programmversionen ausgetauscht werden sollen
- ***Kuang*** Expertensystem. Es enthält Regeln mit denen es testet, ob das System möglicherweise verletzlich ist.

TIGER:

Ist ein ähnliches System wie COPS und auch eine Sammlung von Shell-Skripten und C-Programmen. Hauptziel ist es herauszufinden, ob es einem Angreifer gelingen kann, root-Rechte zu erlangen. Dabei wird davon ausgegangen, dass der Angreifer recht leicht eine andere Benutzerkennung erlangen kann. Die Prüfungen decken folgende Problembereiche ab:

- cron Einträge
- mail aliases
- NFS
- inetd Einträge (steuern welcher Dienst gestartet werden soll, wenn ein IP-Paket auf einer bestimmten Portnummer empfangen wird)
- PATH Umgebungsvariablen
- .rhosts und .netrc Dateien.
- Zugriffsrechte in bestimmten Verzeichnissen auf bestimmte Dateien
- ungewöhnliche Dateien werden gesucht
- Integrität von wichtigen Programmen mit Hilfe digitaler Signaturen

- Pfadnamen aus Programmdateien werden geprüft. Werden aus einem Programm heraus andere Programme gestartet, so könnte ein Angreifer versuchen, anstatt des eigentlich vorgesehenen Programms sein Schadprogramm unter dem gleichen Pfadnamen zu installieren.

1.7.2 Network Scanner

Ein Network Scanner untersucht, welche Netzdienste auf einem Rechner installiert und freigeschaltet sind. Diese Netzdienste werden dann auf Konfigurationsprobleme hin untersucht.

Saint – Security Administrator's Integrated Network Tool:

Saint untersucht den Rechner von außen, womit es auch jedem Angreifer ermöglicht, fremde Rechner auf Schwachstellen zu untersuchen. Saint untersucht die angebotenen Netzwerkdienste wie finger, NFS, NIS, ftp usw. und kann damit Dienste erkennen, die wegen bekannter Schwachstellen od. Risiken aktualisiert oder gar nicht aktiviert sein sollten. Um Saint zu starten muss man root-Rechte besitzen.

Den Host- und Network-Scannern ist allen gemeinsam dass sie leicht erhältlich sind, einfach zu bedienen sind und somit auch von relativ unerfahrenen Angreifern eingesetzt werden können.

1.8 Angriffe auf Verschlüsselung

Mit *Kryptoanalyse*-Techniken ist es möglich, verschlüsselte Daten ohne Kenntnis des Schlüssels wieder zu entschlüsseln.

1.8.1 Klassifikation der Angriffe

Das Unterscheidungskriterium zwischen den verschiedenen Angriffstypen ist die Menge der Informationen, die dem Angreifer zur Verfügung stehen. *(Die Auflistung der Angriffstypen beginnt mit dem Angriff mit der geringsten Menge an zur Verfügung stehender Information)*:

Ciphertext only Angriffe:

Dem steht nur eine oder mehrere verschlüsselte Nachrichten zur Verfügung. Außerdem ist davon auszugehen, dass auch der Algorithmus selbst bekannt ist. Im wesentlichen bestehen diese Angriffe aus *brute force* Angriffen oder aus statistischen Angriffen, wo Methoden der Statistik oder Wahrscheinlichkeitsrechnung angewandt werden. Der Erfolg der statistischen Methoden hängt davon ab, dass genügend Geheimtext zur Verfügung steht.
Mit dem Demoprogramm *CrypTool* können verschieden Verschlüsselungsverfahren ausprobiert werden und auch verschiedene Analysen, wie z.B. Histogramme der Zeichen in Klar- und Geheimtext, durchgeführt werden.

Known Plaintext Angriffe:

Hierbei befindet sich der Angreifer im Besitz einiger bekannter Paare von Klartext und zugehörigem Geheimtext. *(Z.B. kennt der Angreifer bei Blockverschlüsselung den Inhalte einiger Blöcke, weil z.B. Teile der verschlüsselten Daten, wie z.B. Copyrights in Java Source, allg. bekannt sind.)* Damit können Klar- u. Geheimtext auf bestimmte Muster hin untersucht werden und so Rückschlüsse über den benutzen Schlüssel getroffen werden. -> siehe Abschnitt 1.8.3 *Lineare Kryptoanalyse*.

Chosen Plaintext Angriffe:

Der Angreifer hat die Möglichkeit, bestimmte Klartexte verschlüsseln zu lassen. Die können so gewählt werden, dass sie Muster enthalten, die nach der Verschlüsselung Rückschlüsse auf den verwendeten Schlüssel zulassen -> siehe Abschnitt 1.8.4 *Differentielle Kryptoanalyse*. Eine Erweiterung sind die ***Adaptive Chosen Plaintext*** Angriffe, wo auf den Ergebnissen der bisherigen Analysen immer wieder neue Klartexte zur Verschlüsselung ausgewählt werden können.

Weiter Angriffstypen:

- ***Chosen Ciphertext Angriff***: Angreifer kann verschieden Geheimtexte erstellen und die zugehörigen entschlüsselten Klartexte analysieren.
- ***Chosen Key Angriff***: Angreifer benutzt Informationen über den Schlüssel.
- ***Kryptoanalyse mit Gewalt***: Angreifer versucht von jemanden, der den Schlüssel kennt, die Herausgabe zu erzwingen.

1.8.2 Brute Force Angriffe:

Es werden systematisch alle möglichen Schlüssel getestet. Je größer der Schlüssel ist desto größer ist auch der potentielle Schlüsselraum der durchgetestet werden muss.
Schutzmaßnahmen:

- Keine Möglichkeit, einen Computer zur Generierung und zum Test der Schlüssel einzusetzen (z.B. Bankomat).
- Sperren des Zugangs nach einer gewissen Anzahl von Fehlversuchen.
- Verzögern der Überprüfung so dass mit jedem Fehlversuch die Überprüfung immer langsamer wird.
- Einsatz eines IDS.
- Vergrößerung des Schlüsselraums durch Verwendung längerer Schlüssel.

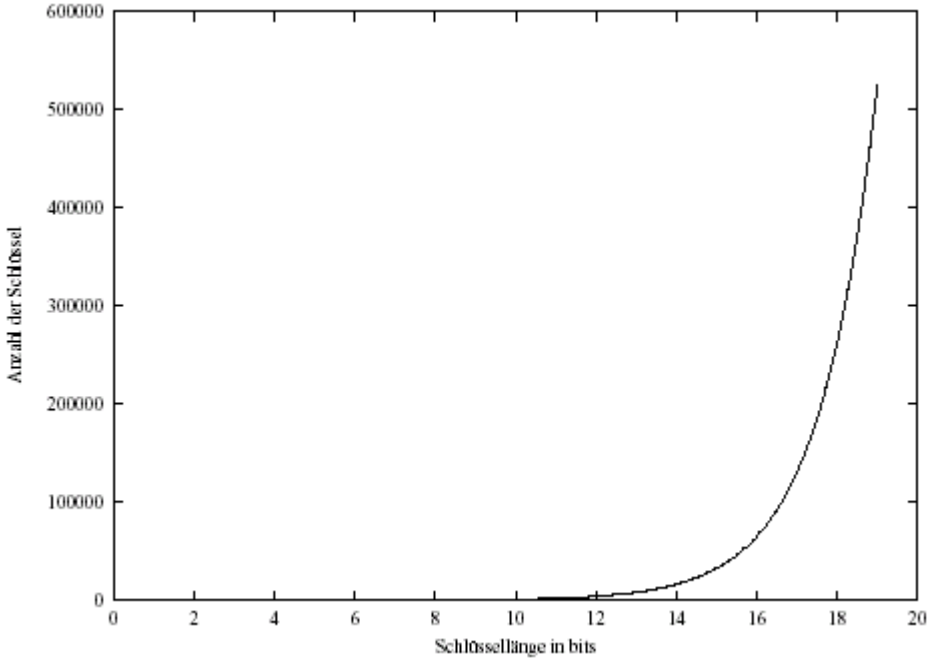

Abb. 1.4: Exponentielles Wachstum des Schlüsselraumes

1.8.3 Lineare Kryptoanalyse

(von Matsui) Es handelt sich dabei um einen *known plaintext* Angriff. Der Angriff konzentriert sich auf schwächen im DES Algorithmus *(siehe Kurs 1866, Abschnitt 2.3.3)*, im

konkreten auf ungünstig gewählte S-Boxen. Der Angreifer versucht, lineare Ausdrücke zu konstruieren, die über allen gegebenen Klartext/Geheimtext Paaren eine ungewöhnliche Verteilung bei der Auswertung ergeben.
Betrachtung der Vorgangsweise für eine einzelne Runde *(lässt sich auf mehrere Runden erweitern)*:

Sei R_i ein Bit aus dem Klartext und Y_j ein Bit aus dem Geheimtext. Es wird nun versucht, einen linearen Ausdruck der Form

$$R_{i1} \oplus R_{i2} \oplus \ldots \oplus R_{in} \oplus Y_{j1} \oplus Y_{j2} \oplus \ldots \oplus Y_{jm} = 0 \qquad \text{mit } \oplus \ldots XOR$$

zu konstruieren. Gesucht sind also Indizes $i_1 \ldots i_n$ und $j_1 \ldots j_m$, so dass für möglichst viele Klartext/Geheimtext Paare der obige Ausdruck wahr ist. Bei zufällig generierten Bits für R_i und Y_j sollte die Wahrscheinlichkeit dafür 0,5 sein. Findet man Indizes für die diese Wahrscheinlichkeit deutlich von 0,5 abweicht so lässt dies Rückschlüsse auf den Schlüssel zu. Zum Auffinden der entsprechenden Index-Kombinationen versucht man nun für jede S-Box *(ist die einzige nicht-lineare Komponente der Funktion F in einer Verschlüsselungsrunde von DES)* einen Ausdruck der obigen Form zu finden, für den die Abweichung von der erwarteten Wahrscheinlichkeit möglichst groß ist. Dazu wertet man alle Ein/Ausgabe-Kombinationen für die S-Boxen aus *(S-Boxen haben bei DES als Eingabe nur halbe Blockgröße -> Rechenaufwand vertretbar)*. Die gefundenen Ausdrücke der einzelnen Runden werden anschließend kombiniert, wobei man dabei Ausdrücke sucht, in denen sich die Zwischenergebnisse eliminieren *(dabei hilft XOR)*. So kann man möglicherweise Ausdrücke generieren, die nur mehr aus dem Klartext der ersten Runde u. dem Geheimtext der letzten Runde bestehen.

Neue Verschlüsselungsalgorithmen werden heute von Anfang an auf ihre Anfälligkeit gegen lineare Kryptoanalyse getestet.

1.8.4 Differentielle Kryptoanalyse

Ist ein *chosen plaintext* Angriff, bei dem einander möglichst ähnliche Klartexte ausgewählt werden, die sich nur an sehr wenigen Stellen unterscheiden. Nach der Verschlüsselung werden die Geheimtexte verglichen und untersucht, an welchen Stellen sich diese unterscheiden. Es wird wieder nach ungewöhnlichen Häufigkeiten gesucht.
Die Differenz zwischen 2 Klartextblöcken *R'* und *R''* wird bitweise mit einem XOR ermittelt, also $\Delta R = R' \oplus R''$. Die Wahrscheinlichkeit, dass bei 2 zufällig ausgewählten Blöcken *R'* und *R''* ein vorgegebener Differenzwert ΔR entsteht ist 2^{-n} bei *n*-Bits Länge der Blöcke. Man untersucht also Klartextpaare mit einer gegebenen Differenz ΔR so dass ein bestimmte Differenz ΔY der Geheimtextpaare auftritt. Dabei werden zuerst die einzelnen Verschlüsselungsrunden angegriffen und anschließend die Ergebnisse wieder kombiniert *(Ergebnis einer Runde ist immer Eingabe der nächsten Runde)*.

1.8.5 Faktorisierung großer Zahlen

Dabei geht es um die Zerlegung einer gegebenen Zahl in ihre Primfaktoren. Ein einfacher Algorithmus probiert nach einander die einzelnen Teiler aus und wenn kein Zahl kleiner der Quadratwurzel der zu prüfenden Teiler gefunden wird, so ist der Teiler prim. Die Komplexität dieses Algorithmus ist $O(\sqrt{n})$. Ist *n* eine sehr große Binärzahl mit *k* Stellen, also 2^k, so gilt $\sqrt{2^k} \approx 2^{k/2}$. Der Aufwand wächst also exponentiell in der Zahl der Stellen von *n*. Alle bekannten Algorithmen zur Faktorisierung arbeiten bestenfalls mit exponentiellen Aufwand *(was gut für die Sicherheit von RSA ist)*, einer der besten davon ist der

Quadratic Sieve Algorithmus:

Ist n die zu faktorisierende Zahl so sucht man 2 Zahlen x und y so dass gilt

$x^2 \equiv y^2 \bmod n$ (1) (x^2 ist kongruent zu y^2 *mod n*)

$x \not\equiv \pm y \bmod n$ (2)

Eigenschaft (1) bedeutet, dass *n* die Differenz (x^2-y^2) teilt und (2) bedeutet, das *n* weder (*x*-*y*) noch (*x*+*y*) teilt. Berechnet man nun den *ggT(n, x-y)* und *ggT(n, x+y)* so sind diese Zahlen möglicherweise die ersten nicht-trivialen Faktoren von *n*.

Es wird also zunächst nach Quadratzahlen gesucht die größer als *n* sind. Dann wird getestet, ob x^2-*n* wieder eine Quadratzahl ergibt. Wenn ja, dann ist $\sqrt{x^2 - n} = y$ das gesuchte *y*. Danach wird der wie oben angegebene *ggT* berechnet.

Für die Performance des Algorithmus ist es wichtig, wie schnell man diese beiden Zahlen *x* u. *y* findet. Dafür wird eine Menge potentieller Kandidaten x_i (**factor base**) erstellt, aus der dann diejenigen Werte mit den oben genannten Eigenschaften herausgefiltert (**sieve**) werden.

Die bisher besten bekannten Algorithmen haben folgende Komplexität:

Algorithmus	*Komplexität*
Quadratic Sieve	$O(e^{(1+o(1))\sqrt{\ln(n)\ln^2(n)}})$
Elliptic Curve	$O(e^{(1+o(1))\sqrt{\ln(p)\ln^2(p)}})$
Number Sieve Field	$O(e^{(1,92+o(1))\sqrt[3]{\ln(n)(\ln^2(n))^2}})$

o(1) bezeichnet eine Konstante die bei großen n sehr nah bei 0 liegt.

RSA Schlüssel mit 512 Bit sind nicht mehr sicher, derzeit wird mind. eine Länge von 1024 Bit empfohlen.

1.8.6 Quantencomputer

Basieren nicht auf einem deterministischen Rechnermodell (von Neumann-Architektur, Turing-Modell) sondern verfolgen einen anderen Ansatz (das Auslesen eines Quantenbits **Q-Bit** ist wegen der *Heisenbergschen Unschärferelation* nicht möglich, ohne das Q-Bit dabei zu verändern). Damit können Quantenalgorithmen wesentlich effizienter sein. *Shor* hat einen probabilistischen Algorithmus vorgestellt, der diskrete Logarithmen *(genutzt von ElGamal u. Diffie Hellmann, siehe Kurs 1866 Abschnitt 2.4.2)* und die Faktorisierung *(siehe RSA)* natürlicher Zahlen mit einem Quantencomputer mit *polynomiellen* Aufwand berechnet.

2. Benutzersicherheit

2.1 Einführung -> siehe Skript

2.2 eCommerce

2.2.1 Grundlagen des eCommerce

Klassifikation anhand der beteiligten Parteien:

- B2C – Business to Consumer: Klassischer eCommerce zwischen Firma u. Kunde. Probleme mit Sicherheit über Identität des Kunden, keine rechtsgültige Unterschrift.
- B2B – Business to Business: Handel zwischen Firmen über das Internet. Neben den selben Problemen wie beim B2C gibt es hier noch weitere große Probleme bezüglich der Integration der neuen Möglichkeiten in die bereits existierenden Programme und Systeme einer Firma (z.B. bestehende Bestellsysteme). Dafür sind meist Anpassungsarbeiten in folgenden Bereichen notwendig:
    - Abruf des Inhalts aus den existierenden Systemen (z.B. Lagerbestand) über das Internet.
    - Transaktionen (z.B. Bestellungen) sollen über das Internet eingegeben werden können und automatisch an bestehende Systeme übergeben werden.

Klassifikation der Güter:

Von der Art der Güter im eCommerce hängen einige wichtige Aspekte u. Problembereiche ab.

- Unterscheidung nach Lieferwegen:
    - Materielle Güter müssen physisch transportiert werden, es werden Lager o.ä. benötigt. Bestimmte Bezahlverfahren (z.B. Nachnahme) sind nur hier möglich. Spezielles Kundenverhalten.
    - Immaterielle Güter können direkt über das Internet in elektronischer Form „ausgeliefert“ werden (z.B. Tickets, Musik usw.).
- Unterscheidung nach Wertigkeit:
    - Hochpreisige Güter erfordern ein sicheres Bezahlverfahren und eine sichere Abwicklung der Transaktion. Eine Gebühr dafür wird meist akzeptiert.
    - Bei Gütern mittleren Werts können die Sicherheitsanforderungen und damit die Kosten eines Bezahlverfahrens niedriger sein, da das mögliche Schadensausmaß deutlich geringer ist.
    - Geringwertige Güter erfordern Bezahlverfahren die möglichst geringe Kosten verursachen. Die Sicherheit ist zweitrangig.

2.2.2 Anforderungen an Bezahlverfahren

Diese sind abhängig von der Position der Beteiligten. Dem Kunden ist wahrscheinlich mehr an der Einfachheit und der Anonymität gelegen, dem Händler ist wichtig dass er das Geld erhält und dass die Transaktionsgebühren niedrig sind.

Bequemlichkeit:

Das Verfahren soll ohne viel Aufwand für alle Beteiligten nutzbar sein. Als unbequem gilt, wenn

- die Installation spez. SW od. HW erforderlich ist,
- die Registrierung bei einer zentralen Stelle notwendig ist.

Akzeptanz:

Darunter versteht man die Verbreitung des Bezahlverfahrens. Als Kunde möchte man möglichst wenige Bezahlsysteme einsetzen.

Transaktionskosten:

Diese sollen natürlich so niedrig wie möglich sein.

Sicherheit:

- Fälschungssicherheit
- Sicherheit vor technischen Problemen (was passiert, wenn eine geladene elektronische Geldbörse (Chipcard) defekt ist)
- Betrugssicherheit
- Anonymität
- Beachtung gesetzlicher Regelungen

2.2.3 Nachnahme

Vorteile:

- Der Kunde muss erst dann bezahlen, wenn er die Ware tatsächlich erhält.
- Der Händler kann sicher sein, dass er entweder das Geld erhält oder die Ware zurück bekommt.

Nachteile:

- Relativ hohe Kosten für Nachnahmegebühren
- Der Kunde kann erst nach Aushändigung des Geldes das Paket auf richtigen Inhalt überprüfen.
- Relativ hoher Verwaltungsaufwand, da Buchungen normalerweise manuell erfolgen.

Nachnahme ist nur sehr eingeschränkt für immaterielle Güter einsetzbar.

2.2.4 Überweisung

Vorteile:

- Automatisierung möglich
- Relativ niedrigen Kosten
- Ausführungsfristen für die Überweisung sind geregelt u. hängen vom Verhältnis zw. Absender- und Empfängerinstitut ab.

Nachteile:

- Die Beteiligten müssen ein Konto unterhalten
- Beim eCommerce muss entweder der Anbieter od. der Zahler in Vorleistung treten. *(Meist ist es der Anbieter der Rechnung zusammen mit Ware verschickt.)*
- Die anfallenden Kosten sind unabhängig von der Höhe der Überweisung -> rel. hohe Kosten bei kleinen Beträgen.

2.2.5 Lastschrift

Einzugsermächtigung: Der Zahlende erteilt dem Zahlungsempfänger im vorab die Erlaubnis, Geld vom Konto des Zahlenden einzuziehen. Der Zahlende hat aber das Recht, eine Lastschrift ohne Angabe von Gründen zurück zu geben u. somit die Abbuchung von seinem Konto Rückgängig zu machen.

Abbuchungsauftrag: Der Zahlende erteilt seiner Bank den Auftrag, Lastschriften von bestimmten Dritten auszuführen.

Vorteile:

- Zahlungsempfänger löst die Zahlung aus und erhält dadurch sein Geld fristgerecht.
- Kostengünstiges Verfahren wegen elektronischer Erstellung der Lastschriften.

- Eingehende Rücklastschrift kann der Empfänger autom. in Mahnverfahren überleiten.
- Der Zahlende braucht sich nicht um Termine kümmern.

Nachteile:
- Missbräuchliche Belastung des Kontos des Zahlenden möglich.
- Die Zahlungsauslösung gibt keine Sicherheit, tatsächlich das Geld zu erhalten (z.B. wegen mangelnder Deckung des Zahlenden).
- Bei Erteilung der Einzugermächtigung kann sich Empfänger nicht sicher sein, ob das Konto des Zahlenden existiert.
- Lastschriftverfahren national begrenzt.

2.2.6 Kartenzahlungen

Kundenkarten, ec-Karten, Geldkarten:

Vorteile:
- Zahlungsempfänger kann seine Risiken minimieren, muss dafür aber höhere Kosten tragen.
- Der Bezahlvorgang ist einfach zu handhaben u. eine elektronische Verarbeitung ist einfach möglich.

Nachteile:
- Die Verfahren können im Internet nicht direkt eingesetzt werden, da hierfür ein Kartenlesegerät am PC notwendig wäre (ist nicht Standard).
- Bei kleinen Beträgen sind die Kosten verhältnismäßig hoch.

Kreditkarte:

Vorteile:
- Zahlungsempfänger hat die Garantie, dass das Geld ankommt, falls erforderliche Kartenprüfung durchgeführt wurde.
- Der Zahlende hat eine kurzfristigen Kredit erhalten.
- Der Zahlende kann auch im Ausland zahlen.
- Unter Umständen kann mit Kreditkarte zahlen ohne sie physisch vorzulegen (im Internet).

Nachteile:
- Durch das Disagio ist die Bezahlung mit Kreditkarte für den Empfänger recht teuer.
- Kreditkartenmissbrauch ist relativ einfach.
- Wird Karte nicht vorgelegt sondern nur die Kartennummer und Gültigkeitsdauer übertragen (Internet) so ist Missbrauch sehr einfach.

SET – Secure Electronic Transaction:

SET wurde eingeführt, um dem Betrug durch Mailorder ohne Vorlage der Kreditkarte entgegenzuwirken. Der Informationsaustausch erfolgt gesichert, d.h.

1. die Beteiligten werden authentisiert und
2. die Nachrichten werden verschlüsselt.

Durch das Konzept der dualen Signatur wird sichergestellt, dass das „need to know“ Prinzip weiter eingehalten wird. D.h. der Händler erhält keine Informationen über die Finanzen des Käufers und der Zahlungsabwickler kein Informationen über den Hintergrund der Zahlung.
Im SET Protokolle ist festgelegt, dass

- zur asymmetrischen Verschlüsselung DES,
- zur symmetrischen Verschlüsselung RAS und
- zur Berechnung der Hash-Werte SHA-1

verwendet werden muss. Dies ist ein Nachteil, da der Einsatz moderner Verschlüsselungsverfahren somit eine Protokolländerung notwendig machen. Die Gültigkeit

der Public Keys wird durch ein X.509 Zertifikat *(siehe Kurs 1866 Abschnitt 2.7.3)* bestätigt. Durch das Konzept der dualen Signatur und der Authentisierung aller Beteiligten ist SET ein sicheres Protokoll. Um SET zu nutzen, ist eine entsprechende SW am Kunden-PC nötig.

2.2.7 Digitales Geld

Alle diesbezüglichen Projekte in Deutschland wurden mittlerweile wegen der Kosten für den hohen technischen Aufwand (elektronische Fälschungssicherheit, hohe Verfügbarkeit der Systeme) und der mangelnden Nachfrage von Kunden wieder eingestellt.

2.2.8 Weiter Bezahlverfahren

Paybox:

Der Zahlende autorisiert sich seine Zahlungen mit Hilfe eines Mobiltelefons. Der Zahlende und der Zahlungsempfänger müssen bei Paybox registriert sein. Dabei wird Konto- und Mobilfunknummer der Teilnehmer erfasst.
Vorteil:

- Missbrauch ist nur möglich, wenn ein Angreifer in Besitz des Mobiltelefons und der Paybox –Geheimzahl gelangt.
- Der Zahlende muss nur seine Mobiltelefonnummer an den Zahlungsempfänger übermitteln.
- Das Verfahren kann flexibel eingesetzt werden (Internet, Parkautomaten, usw.).
- Man kann klassische Überweisungen durchführen ohne die Bankverbindung des Empfängers kennen zu müssen.

Nachteile:

- Die Kosten sind relativ hoch.
- Der Zahlungsempfänger hat das Zahlungsausfallrisiko zu tragen (z.B. mangelnde Deckung).
- Bei Zahlung von kleinen Beträgen sehr teuer.
- Zusätzlich zu den Paybox-Kosten fallen auch noch die üblichen Bankkosten an.

Paybox ist eine bequeme und sichere Möglichkeit Lastschriften zu erzeugen.

Stackbox:

Ein Java Applet von Stackbox verwaltet die Kontoinformationen des Benutzers. Weiters kann der Benutzer seine PINs und TANs in das Applet eingeben und somit kann das Applet selbständig zu den einzelnen Banken Verbindungen Aufbauen, sich authentisieren, Kontoinformationen einholen und Transaktionen durchführen. Außerdem kann das Applet eine Art Bezahlfunktion ausführen indem es dem Zahlungsempfänger bestätigt, dass der Zahlende eine Überweisung bei seiner Bank in Auftrag gegeben hat.

2.3 Schutz des privaten PCs

siehe Kurs 1866, Abschnitt 3.5

2.4 Biometrie

Biometrische Systeme sollen Menschen anhand ihrer physiologischen Eigenschaften oder ihrer Verhaltensmerkmale identifizieren bzw. ihre Identität verifizieren.

- ***Identifikation***: Aus einer gegebenen Menge von Objekten dasjenige zu finden, das mit den gegebenen Merkmalen ausgestattet ist. Also ein gegebenes Muster in einer großen Menge von Objekten wiederfinden – *one to many matching (1:n)*
- ***Verifikation***: Es wird überprüft, ob ein Objekt die gegebenen Merkmale besitzt. Also ein vergleich zwischen dem gegebenen Muster und einem einzelnen Objekt durchführen – *one to one matching (1:1)*

Bei herkömmliche Systeme basiert die Authentisierung auf den Merkmalen „Besitz" und „Wissen". Dies ist problematische, da beide Merkmale übertragbar sind. Biometrische Eigenschaften sind hingegen fest mit der Person verbunden und sind deshalb immer verfügbar und nicht ohne weiteres auf andere Personen übertragbar.

2.4.1 Übersicht biometrischer Merkmale

Neben den statischen Merkmalen wie Bild des Gesichts od. Fingerabdruck können auch dynamische Merkmale wie Stimme oder Unterschrift verwendet werden. Folgende Kriterien sind bei der Auswahl des Merkmals zu beachten:

- ***Eindeutigkeit:*** Merkmal sollte tatsächlich von Mensch zu Mensch eindeutig sein (siehe Zwillinge).
- ***Erfassbarkeit:*** Das Merkmal muss technisch möglichst einfach bei jeder Authentisierung neu erfassbar sein (z.B.: DNA dzt. ungeeignet).
- ***Fälschbarkeit***
- ***Möglichkeit der autom. Auswertung:*** Es muss effiziente Algorithmen geben, die die Merkmale vergleichen u. prüfen können.
- ***Zeitliche Konstanz des Merkmals:*** z.B. dynamische Merkmale wie Stimme, Unterschrift verändern sich mit der Zeit.
- ***Verfügbarkeit des Merkmals:*** Nicht alle Menschen verfügen über alle Merkmale.
- ***Benutzerakzeptanz:*** z.B. geringe Akzeptanz des Fingerabdrucks.

Folgende biometrische Merkmale können zur Authentisierung von Personen eingesetzt werden:

- ***Stimme:*** Analyse des Frequenzspektrums. Ist aber von Angreifern leicht kopierbar (Audioaufnahme der Stimme des legitimen Benutzers).
- ***Unterschrift:*** Zwei Varianten der Unterschrifterkennung
  - Erkennung des Bildes der Unterschrift. Leicht angreifbar durch Fotokopie der Unterschrift.
  - Erkennung des dynamischen Verhalten beim Unterschreiben. Zusätzlich zum Bild werden noch Druck u. Geschwindigkeit beim Unterschreiben erfasst.
- ***Fingerabdruck:*** hat ca. 35 Minutien (Kreuzungen, Verzweigungen od. Enden der Linien), 12 reichen oft schon für Erkennung aus. Es müssen also nur die Minutien gespeichert und verglichen werden. Nur schwer fälschbar, da Erfassungsgeräte auch noch prüfen, ob Finger von Blut durchströmt wird. Technisch kann die Erfassung entweder optisch (CCD-Sensor) od. über kapazitive Sensoren erfolgen. Erreichbare Auflösung ca. 500dpi.
- ***Iris:*** Die Muster des farbigen Teiles des Auges dienen als Merkmal – Vergleich geometrischer Muster. Moderne System funktionieren auch mit Brillen (auch Sonnenbrillen).
- ***Gesichtserkennung:*** Form und Position charakteristischer Merkmale des Gesichts werden ausgewertet. Die eingesetzten Algorithmen sind jedoch sehr rechenintensiv. Die Erfassung muss 3-dimensional erfolgen, Kopien (Fotos oder künstliche Köpfe) müssen als solche erkannt werden.

- ***Handgeometrie und Gefäßanalyse:*** Es wird die individuelle Knochenstruktur bzw. der Verlauf der Venen am Handrücken als Merkmal verwendet.

2.4.2 Ablauf bei biometrischen Verfahren

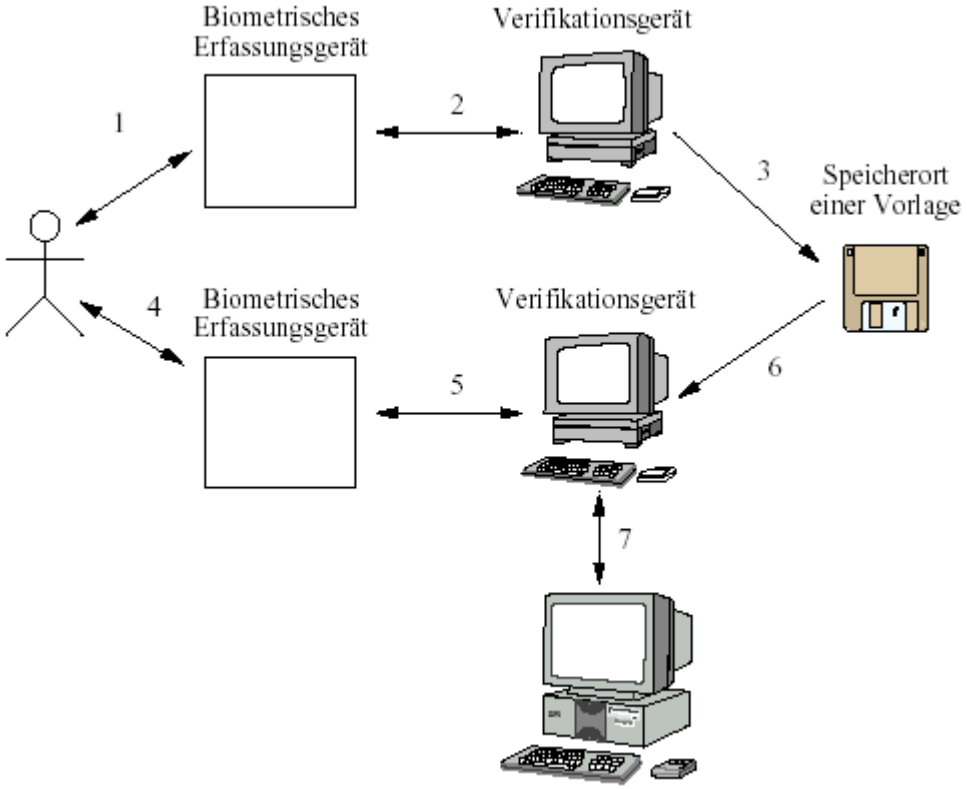

Abb. 2.1: Ablauf bei der Erfassung und Überprüfung biometrischer Merkmale

1. Erstmalige Erfassung mittels entsprechendem Eingabegerät.
2. Bearbeitung (z.B. Varianz bzw. Durchschnittswert der Testmuster berechnen), Transformation der Daten.
3. Speicherung der erfassten u. aufbereiteten Daten.
4. Erneute Erfassung für die Prüfung
5. wie 2.
6. Prüfung gegen den Referenzsatz.
7. Aktion auf Basis des Authentisierungsergebnisses

2.4.3 Sicherheit biometrischer Verfahren

Biometrische Verfahren sind immer mit einer gewissen Fehlerwahrscheinlichkeit behaftet, da bei der Erfassung biometrischer Daten immer Abweichungen auftreten. Das Verfahren wird also durch einen Parameter gesteuert, der angibt wie viel Abweichung der Daten tolerierbar ist. Bei den auftretenden Verfahren treten Fehler auf, wobei man folgende Typen unterscheidet:

- ***False Rejection Rate***: Gibt an, wie häufig ein legitimer Benutzer vom System abgewiesen wird. Grund ist meist ungenaue Erfassung des Merkmals.
- ***False Acceptance Rate***: Gibt an, wie häufig ein Angreifer fälschlicherweise vom System akzeptiert wird. Grund ist meist große Übereinstimmung der Merkmale zwischen Angreifer u. legitimer Benutzer.

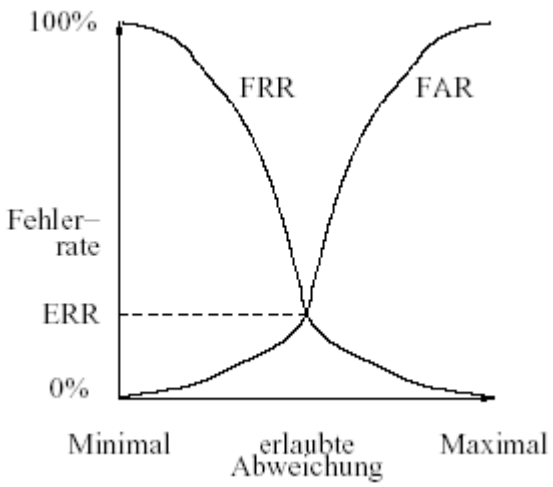

Abb. 2.2: Zuverlässigkeit biometrischer Verfahren

ERR – Equal Error Rate repräsentiert den Trade-Off zwischen hoher Sicherheit (FRR) und hohem Durchsatz (FAR) des Systems.

Zuverlässigkeitswerte biometrischer Verfahren nach einer Studie der National Physics Laboratories, in Teddington UK:

Biometrisches Verfahren	**FAR**	**FRR**
Iris-Scan	0,0001%	0,25%
Fingerabdruck	0,008% - 0,45%	2,5% - 11%
Stimmerkennung	0,03%	2%
Handgeometrie	0,7%	0,5%
Gesichtserkennung	0,45%	17%

Beim Fingerabdruck wurden optische u. kapazitive Sensoren eingesetzt, wobei die optischen Sensoren die schlechteren Werte lieferte.

3. Anbietersicherheit

3.1 Einführung

siehe Skript.

3.2 Virtual Private Networks – VPN

3.2.1 Motivation für VPNs

Bei Firmen od. Organisationen mit Mitarbeitern an verschiedenen Standorten sollen trotzdem alle Mitarbeiter die selben IT-Systeme und die selben Daten benutzen. Das selbe gilt für mobile Mitarbeiter (z.B. Service, Außendienst) oder externe Mitarbeiter (Freelancer). Aus Sicht dieser Mitarbeiter soll das Firmennetz wie ein großes lokales Netz aussehen, unabhängig vom eigenen Standort. Man spricht dabei von einem **Intranet** bzw. **site-to-site** Verbindung. Erfolgt die Verbindung über einen lokalen ISP so spricht man von einer **end-to-site** Verbindung.
Innerhalb des lokalen Netzes findet dabei der Datentransport unverschlüsselt statt, über die Verbindungsnetze aber verschlüsselt.

Abb. 3.1: Prinzip eines VPN

Wenn bei B2B Anwendungen der externe Partner Zugriff auf Teile des internen Netzes hat, spricht man auch vom **Extranet**.

Die **Anforderungen an ein VPN** überlagern sich deutlich mit den

- allgemeinen Schutzzielen für sichere IT-Systeme:
  - Vertraulichkeit
  - Authentizität
  - Integrität
  - Verfügbarkeit
- allgemeinen Anforderungen an IT-Systeme wie
  - einfache Bedienbarkeit
  - einfache Administration
  - Effizienz u. Skalierbarkeit
  - Kostengünstig

3.2.2 Prinzipien von VPNs

Die geeignetsten Protokollebenen für VPNs sind Ebene 2 u. 3. Anwendungs-Ebenen bieten zu wenig Sicherheit und die Bitübertragungsebene (Ebene 1) ist schon allein aus Kostengründen (eigenes Kabel verlegen) nicht interessant.

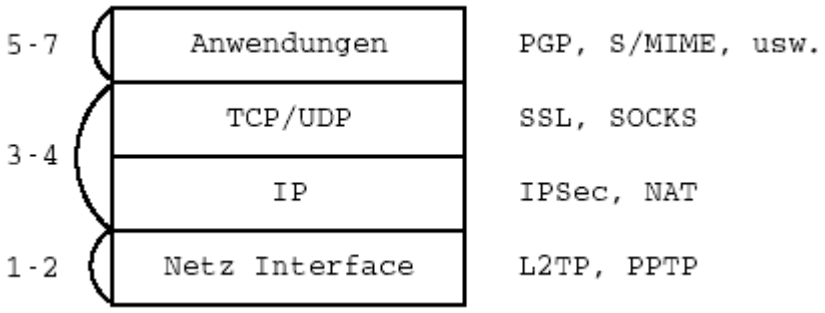

Abb. 3.2: Technologien zur Einrichtung eines VPN nach Ebenen.

Tunneling:

Dabei werden beliebige Datenpakete (gesamter Protokollrahmen) komplett als Nutzlast (*payload*) in einem anderen Datenpaket übertragen. Während des Transports hat das ursprüngliche Datenpaket keinen Einfluss mehr auf Routing od. andere Übertragungseigenschaften. Z.B. für das Versenden von IP-Paketen über eine ATM-Netz oder IP-Paket in einem IP-Paket verpacken (**encapsulation**):

Dabei kann die Nutzlast (also das ganze ursprüngliche IP-Paket) natürlich auch verschlüsselt werden.

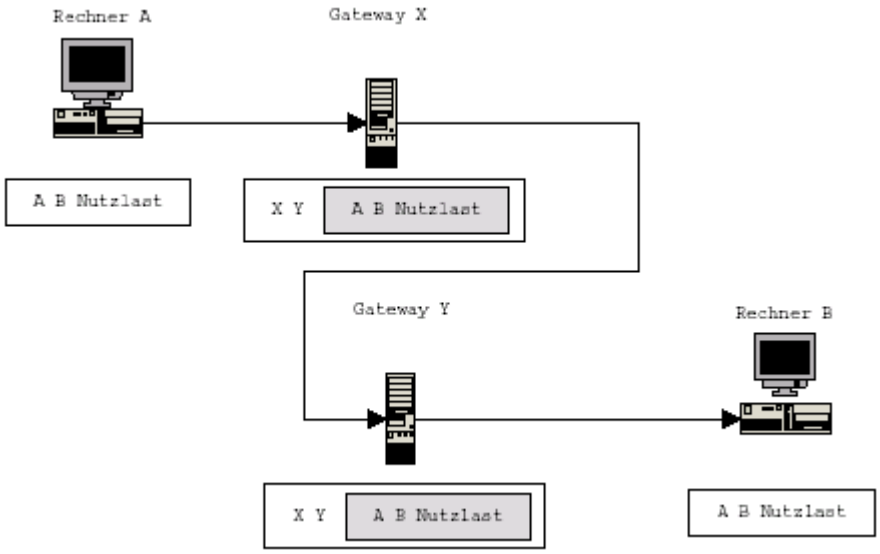

Abb. 3.3: Prinzip des Tunneling von IP-Paketen

Generic Routing Encapsulation – GRE:

Allgemeine Form eines GRE-Pakets:

Delivery-Header
GRE-Header
Payload

Der Delivery-Header enthält die Adresse des Pakets außerhalb des Tunnels *(ist das sicherheitstechnisch nicht problematisch wenn die Zieladresse praktisch sichtbar ist?)*, der GRE-Header enthält technische Informationen der Übertragung durch den Tunnel sowie Infos über den Typ der Nutzlast.

Layer 2 Technologien

Point to Point Protocol (PPP):
Damit können IP-Pakete über eine serielle Leitung übertragen werden. Oft von ISPs für den Internetzugang über Telefonnetz eingesetzt. Der Verbindungsaufbau erfolgt in 3 Phasen:

1. ***Link Control Protocol (LCP)*** Pakete werden ausgetauscht um eine PPP-Leitung aufzubauen.
2. Es werden verschieden Verbindungsparameter ausgehandelt.
3. Gegebenenfalls findet nun eine Authentifizierung statt. (*PAP* oder *CHAP*)

Bei der Authentifizierung mittels ***Password Authentication Protocol (PAP***) wird die Kombination aus Benutzername u. Password benutzt. Der Client überträgt dabei Benutzer-Kennung u. Password im Klartext an den Server.

Das ***Challenge Handshake Authentication Protocol (CHAP)*** benutzt stattdessen ein 3-Wege Challenge/Response Verfahren. Der Server sendet dabei dem Client eine Aufforderung zur Authentifizierung. Der Client antwortet mit seiner Benutzer-Kennung u. einer Challenge. Der Server bildet nun eine Konkatenation aus Benutzer-Kennung, Challenge und dem am Server gespeicherten Passwort für diese Benutzerkennung, bildet den Hash-Wert und schickt diesen zum Client. Der Client geht analog vor und vergleicht die beiden Hash-Werte.

Beide Authentisierungsprotokolle sind Bestandteil des *PPP* und bei beiden Protokollen ist es erforderlich, dass beide Seiten eine Passwort-Datenbank führen. Bei einem Angriff auf eine Datenbank wären alle Passwörter kompromittiert, deshalb arbeiten viele ISPs mit nur einer Benutzerkennung und einem allgemein bekannten Passwort.

Point to Point Tunneling Protocol (PPTP):
(von Microsoft) Als Erweiterung zum *PPP* und erlaubt das Verpacken von PPP-Paketen in IP-Paketen. Es wird dabei GRE angewandt.

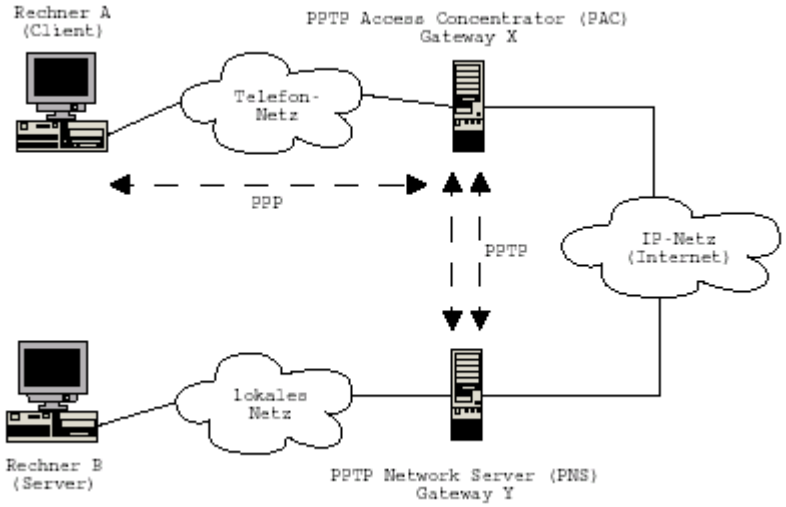

Abb. 3.4: Kommunikation über PPTP

Der Client wählt sich dabei bei einem ***PPTP Access Concentrator (PAC)*** ein, der die *PPP*-Pakete in IP-Pakete verpackt. Der ***PPTP Network Server (PNS)*** packt die PPP-Pakete dann wieder aus. Die Verbindung zwischen den Gateways besteht aus 2 Verbindungen, einem TCP-Kontrollkanal und dem eigentlichen GRE-Tunnel. Für die PPP-Verbindung wird ein Verschlüsselungsverfahren, z.B. ***Microsoft Point to Point Encryption (MPPE) Protocol*** mit RC4 Verschlüsselung, eingesetzt.

Layer 2 Tunneling Protocol (L2TP):
(von IETF) Ist eine Erweiterung des PPTP. Es kann anders als PPP über ein beliebiges paketvermittelndes Ende zu Ende Netz betrieben werden und ist nicht auf IP-Netze beschränkt. Außerdem wird nur eine Verbindung für Kontrollmeldungen und Daten benötigt. Die wesentlichen Unterschiede zwischen PPTP und L2TP:

	PPTP	***L2TP***
Standardisierung	„Firmenstandard“ von M$	IETF und RFC
Trägernetz	IP-Netz	beliebiges paketbasiertes Netz
Anzahl der Verbindungen	zwei (Kontrollkanal u. Datenkanal)	ein
Anzahl der Tunnel	max. einer zwischen PAC u. PNS	beliebig viele

L2TP wurde von M$ in Windows2000 implementiert und somit ebenfalls weit verbreitet und bietet sich für den Einsatz im professionellen Bereich an. Eine Authentisierung kann über CHAP erfolgen, vorausgesetzt die beiden Endpunkte des Tunnels (AC u. NS) kennen ein gemeinsames Geheimnis.

Layer 3 Technologien

IPSec:
Damit sollen auf IP-Ebene die Kommunikationswege zwischen Rechnern abgesichert werden. IPSec stützt sich dabei auf 2 spezielle Standards:

1. *IP Authentication Header (AH)*
2. *IP Encapsulating Security Payload (ESP)*

Beide definieren zusätzliche IP-Header. Mit Hilfe des *AH* werden die Schutzziele *Integrität* und *Authentizität* im wesentlichen durch anhängen eines verschlüsselten Hash-Wertes des Paketes erreicht.

Das *ESP* sichert zusätzlich die *Vertraulichkeit*, indem das IP-Paket oder Teile davon symmetrisch verschlüsselt werden. *ESP* bietet zusätzlich auch Mechanismen zum Schutz der *Integrität* und *Authentizität*.

Ein Security Gateway ist ein System (z.B. ein Router), das die Kommunikationsschnittstelle zwischen einem externen, nicht vertrauenswürdigen Netz und einem internen, vertrauenswürdigen System ist. Für die Rechner im internen Netz übernimmt das Security Gateway die Aufgabe, die angestrebten Schutzziele zu erreichen. Mit IPSec kann man also folgende Verbindungstypen absichern:

- Endgerät zu Endgerät
- Endgerät über Security Gateway in ein internes Netz (remote access)
- Internes Netz -> Security Gateway -> Security Gateway -> ein anderes internes Netz (klassisches VPN)

In IPSec sind 2 Übertragungsmodi vorgesehen:
Transportmodus: es wird nur die Nutzlast verändert, der Original Header des IP-Paketes bleibt unverändert. Durch Einfügung einer kryptographischen Prüfsumme des IP-Paketes in das Paket (AH) kann die Integrität geschützt werden, durch Verschlüsselung der Nutzlast (ESP) die Vertraulichkeit. AH und ESP können auch kombiniert werden, also z.B. zuerst einen AH einfügen und dann das so entstandene IP-Paket durch hinzufügen von ESP verschlüsseln.

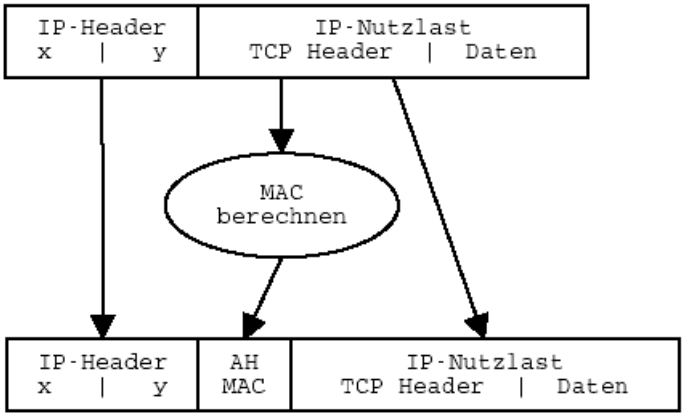

Abb. 3.5: Einfügen eines AH im Transport Modus

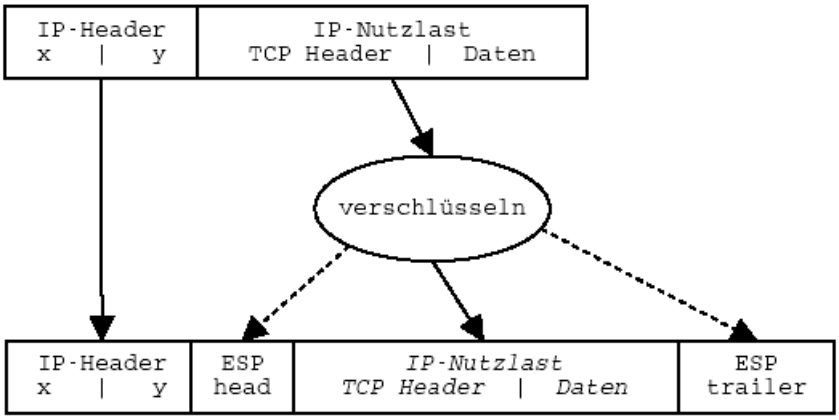

Abb. 3.6: Verschlüsseln mit ESP im Transport Modus

Tunnelmodus: Das komplette IP-Paket wird bearbeitet und als Nutzlast in ein neues IP-Paket verpackt.

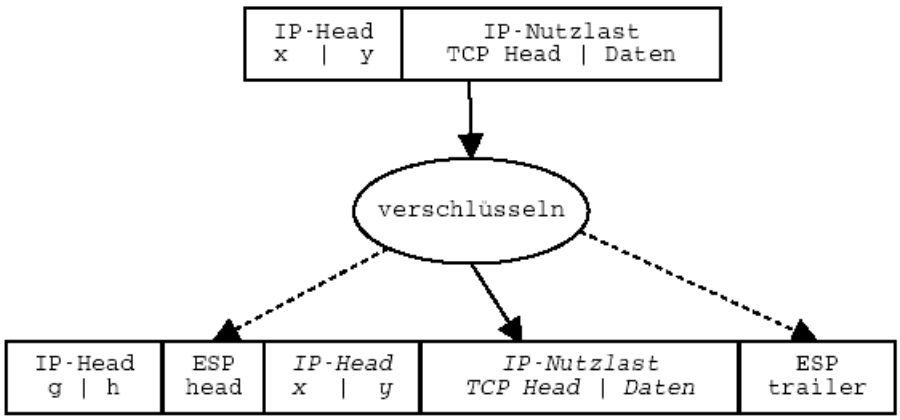

Abb. 3.7: Verpacken eines IP-Paketes mit ESP im Tunnelmodus

Die Darstellung setzt voraus dass IPSec nur auf den Gateways realisiert ist. Deshalb sind die IP-Adressen im neuen Kopf auch nicht mehr ident mit denen im Originalheader, sondern repräsentieren die Adressen der Gateways.

Schlüsselaustausch:

Da bei ESP ein symmetrisches Verschlüsselungsverfahren eingesetzt wird, somit müssen sich die Partner auf einen gemeinsamen Schlüssel, auf einen Verschlüsselungsalgorithmus und

Intervalle für den Schlüsselwechsel einigen. Ebenso ob Transport- od. Tunnelmodus eingesetzt wird.
Im Rahmen der ***Security Association (SA)*** werden diese Vereinbarungen für jeweils **eine Verbindungsrichtung** getroffen. Der Admin des Security Gateways muss also die Parameter für den *inbound* und *outbound* Datenverkehr spezifizieren. Gespeichert werden die Einstellungen in der ***Security Association Database (SAD)***.

Daneben wird eine ***Security Police Database (SPD)*** angelegt, in der festgelegt wird, für welche Verbindungen welche Einstellungen verwendet werden. Ist in der *SPD* für eine outbound Verbindung ein Schlüsseltausch gefordert, so erfolgt dieser nach dem ***Internet Key Exchange (IKE)*** [RFC2409] Verfahren. Dafür können im Rahmen von IPSec zwei Prinzipien verwendet werden:

- Manueller Schlüsseltausch: Die Schlüssel werden physisch z.B. mit Disketten getauscht. Nicht praktikabel bei großen VPNs.
- Automatischer Schlüsseltausch: erfolgt in 2 Phasen
  1. Zuerst tauschen die Systeme Informationen über Algorithmen und Methoden aus (z.B. *Diffie-Hellmann* Schlüsseltausch).
  2. Danach werden die Informationen zur Erzeugung der geheimen Schlüssel nach dem zuvor vereinbarten Verfahren ausgetauscht.

3.2.3 Aufbau von VPNs

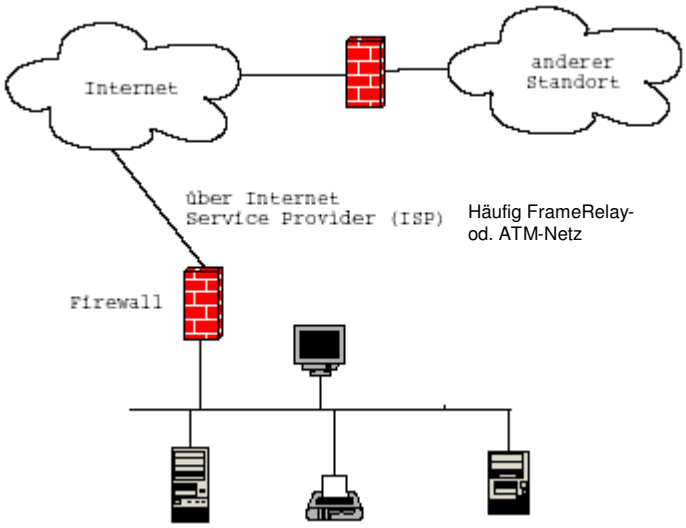

Abb. 3.8: Infrastruktur mit Einsatz eines VPN

In beiden Firewalls wird ein Security Gateway installiert, der die Realisierung des Datenverkehrs zwischen den beiden Standorten übernimmt. Die beiden Security Gateways bekommen die Adresse des jeweils anderen und die Security Policy vorgegeben, wo unter anderem die einzusetzende Technologie festegelegt wurde (z.B. *L2TP* mit *IPSec*). Außerdem legt man noch fest, dass der Datenverkehr verschlüsselt erfolgen soll und tauscht die Schlüssel aus. Unter Linux gibt es mit FreeSwan (www.freeswan.org) eine freie IPSec Implementierung.

Sollen nicht zwei komplette Standorte verbunden werden, sondern nur einzelnen Rechnern ein remote access ermöglicht werden, so kann dies einfacher realisiert werden. Es wird auf der Firewall ein *SSH*-Server als Security Gateway installiert und die passenden *SSH*-Schlüssel generiert. Der public key wird am Security Gateway installiert und die private keys werden passwortgeschützt auf den entfernten Rechnern installiert. Da *SSH* auch Tunneling unterstützt

ist somit nicht nur eine verschlüsselte Terminal-Verbindung möglich sondern auch der Transfer verschlüsselter Daten und auch das geschützte Abholen von Emails.

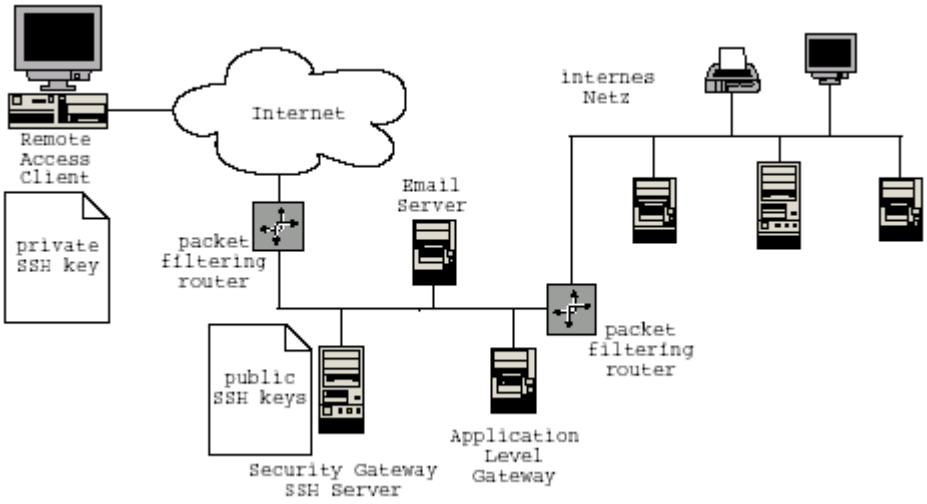

Abb. 3.9: VPN durch SSH

3.3. Intrusion Detection Systeme

3.3.1 Motivation für IDS

Da auch ein gut geschütztes Netz ständig Angriffen ausgesetzt ist müssen die Systeme ständig überwacht werden. Ein IDS unterstützt nun den Admin bei der Erkennung von Angriffen, es sollen also alle „*nicht normalen*“ Vorkommnisse erkannt werden. Außerdem soll ein IDS bekannte Angriffe erkennen und bei der Abwehr unterstützen.

3.3.2 Prinzipien von IDS

Abhängig vom Ort, wo die „Sensoren“ des IDS zur Registrierung eines Angriffs angebracht sind werden folgende IDS unterschieden:

- ***Host based Intrusion Detection System***: Der Sensor ist auf einem Rechner installiert und beobachtet das System u. die Ereignisse die eintreten, das Dateisystem, die Protokolldateien und andere wichtige Dateisysteme. Die Beobachtung erfolgt nicht ständig sondern in bestimmten Intervallen *(Rechner soll für normale Dienste nicht überlastet werden)*.
- ***Network based Intrusion Detection System***: Der Sensor ist direkt an das Übertragungsmedium des Netzes angeschlossen. Er beobachtet (in nahezu Echtzeit) u. analysiert den kompletten Datenverkehr im Netz. Der Sensor ist ein dediziertes System, z.B. Rechner mit Netzwerkkarte im *promiscuous mode*.
- ***Mischformen***: Z.B. ein IDS dessen Sensor auf einem host installiert ist und den Datenverkehr auf der Netzwerkkarte des hosts beobachtet.

Allgemeine Vor- und Nachteile der beiden Ansätze:

	Host based IDS	***Network based IDS***
Zeitpunkt	Ein Angriff wird erst erkannt, wenn sich dessen Auswirkungen im Systemzustand bereits widerspiegeln. Außerdem hat auch noch die Länge des Prüfungsintervalls auf den Erkennungszeitpunkt einen Einfluss.	Erkennt Angriffe normalerweise früher, da die meisten Angriffe über das Netz erfolgen. Ein Angriff kann erkannt werden, bevor er das Zielsystem erreicht und vom IDS eventuell auch schon beendet werden.

		Z.B. bei *DoS* Angriff werden vom IDS *TCP-Reset* Pakete an den Angreifer retourniert und die Verbindung beendet.
Angriffstypen	Ein Angriff auf die unteren Ebenen des TCP/IP-Stack kann nur durch Analyse der Paketheader erkannt werden. Bei Angriffen durch fragmentierte Paket würde ein Host based IDS ewig warten, bis das Paket vollständig ist um es analysieren zu können.	Angriffe, die nicht über das Netz erfolgen, können nicht erkannt werden.
Standort der Sensoren	Relativ einfach. Alle wichtigen und gefährdeten Rechner werden mit einem Sensor versehen.	Sensoren müssen so platziert werden, dass der gesamte Datenverkehr überwacht werden kann. Aus den eingesetzten Netzwerktechnologien ergeben sich dabei folgende Probleme: • ***Switched Networks***: Entweder besitzt der Switch ein spezielles Port, an dem der Sensor alle Pakete mitlesen kann oder es muss in jedes Subnet ein Sensor platziert werden. • ***Verschlüsselte Kommunikation***: Benutzt ein Angreifer eine SSL- od. SSH-Verbindung kann der Sensor zwar mitlesen aber nicht entschlüsseln und den Angriff erkennen.
Beweise eines Angriffs	Hier kann der Angreifer seine Spuren verwischen (Manipulation der Protokolldateien), womöglich sogar vor der nächsten Prüfung durch das IDS.	Hier besteht kaum eine Gefahr, dass der Angreifer seine Spuren verwischen kann. Das IDS zeichnet alle Pakete auf und diese Aufzeichnungen sind vom Angreifer nur sehr schwer manipulierbar.

3.3.3 Aufbau von IDS

Meist wird eine Kombination aus Network based und Host based IDS eingesetzt:

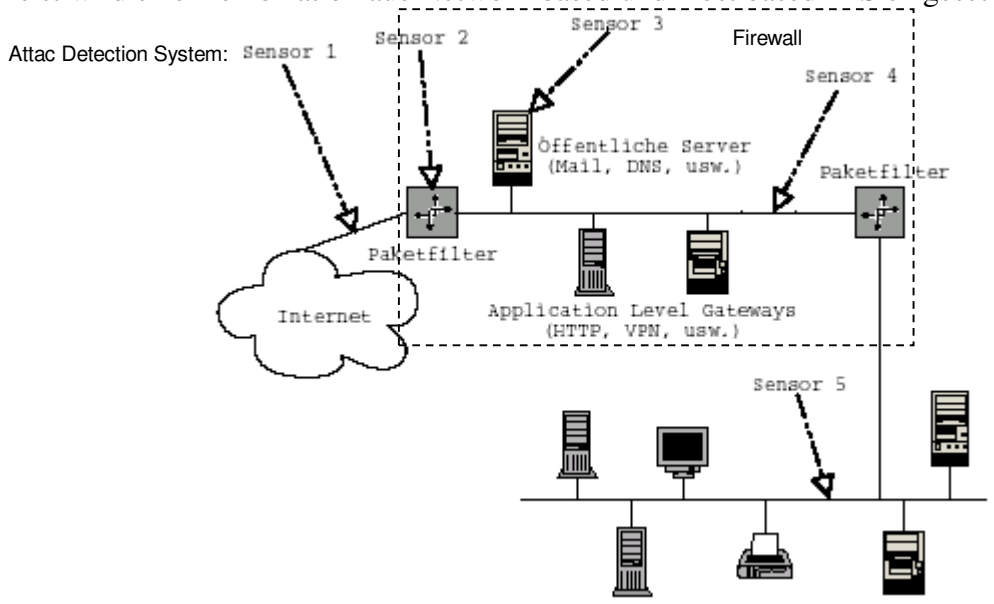

Abb. 3.10: Beispiel für die Platzierung der Sensoren

- **Sensor 1**: Dient als *Attac Detection System* und erlaubt es später erfolgreiche Angriffe besser zu analysieren. Aufgrund der Platzierung ist er besonders gefährdet für

Manipulationen u. muss deshalb besonders geschützt werden. – kein direkte Meldung an Admin.

- **Sensor 2**: Host based IDS der den Zustand des externen Routers überwacht.
- **Sensor 3**: Alle Rechner der DMZ sollten mit einem IDS ausgestattet werden, da diese öffentlich zugänglichen Systeme besonders gefährdet sind. -> direkt Meldung an Admin.
- **Sensor 4**: Network based Sensor überwacht Datenverkehr in der DMZ und von der DMZ ins interne Netz. Ist das DMZ durch ein *dual homed application level gateway* geteilt, so wird für jedes Teilnetz ein Sensor benötigt. Dieser Sensor erkennt als erster wenn ein Angriff den externen Paketfilter überwunden hat.
- **Sensor 5**: Überwacht das interne Netz auf Angriffe von innen. Dafür können auch noch zusätzliche Host based Sensoren eingesetzt werden.

Grenzen von Sensoren:

Ein IDS bzw. dessen Sensoren können nicht absolut zuverlässig sein und tatsächlich alles beobachten. Gründe dafür sind:

- Absturz der Software od. Ausfall des Rechners
- Beschränkter Speicherplatz für die Protokollierung
- Bei Netzen mit hoher Übertragungsbandbreite ist schon eine beträchtliche Rechenleistung erforderlich damit ein Network based IDS
  - das Muster der Daten erkennen,
  - den Rumpf der Pakete analysieren und
  - die Paketheader protokollieren

  kann, bevor das nächste Paket eintrifft.

3.3.4 Network based IDS

TCPdump und *Snort* sind 2 Network based IDS die auf Ebene der IP-Pakete ansetzen und folgende Teile eines IP-Paketes betrachten:

- **IP Header**
- **TCP Header**
- **UDP Header**
- **ICMP Messages**: Das *Internet Control Message Protocol* (wird auch von ping benutzt) wird von Angreifern benutzt, um Informationen über ein Netz zu sammeln um einen Angriff vorzubereiten.

Angriffssignaturen:

Verdächtige oder gefährliche Pakete erkennt man an folgenden Eigenschaften:

- **Ungewöhnliche Spezifikationsausnutzung**: Bei TCP/IP werden normalerweise erst nach dem Verbindungsaufbau Daten gesendet. Es können aber auch während des Handshakes bereits Daten im Paketrumpf mitgesendet werden, diese gelten dann nach [RFC 793] als erste Daten des Datenstroms. Nutzt ein Angreifer diese Möglichkeit aus und das IDS kontrolliert nur den Datenstrom nach dem Handshake so wird der Angriff nicht erkannt. Da kein Feld für die Länge des Paketrumpfes im Header existiert, muss diese aus den Headerdaten ausgerechnet werden.
- **Fragmentierte Pakete**: Damit wird häufig versucht einen gefährlichen Inhalt zu tarnen od. ein *DoS* Angriff versucht. Wenn nie das letzte Fragment gesendet wird, kann man ein System überlasten, da es Ressourcen für die Zwischenspeicherung der

Fragmente erservieren muss. Fragmentierte Pakete erkennt man am gesetzten *more fragments* Bit.

- **TCP-Pakete an Broadcast Adresse**: Da TCP ein verbindungsorientiertes Protokoll ist haben TCP-Pakete an eine Broadcast Adresse wenig Sinn. Der Angreifer erhofft sich dadurch ICMP Fehlermeldungen die ihm Aufschluss über die Netzstruktur geben.
- **Undefinierte Kombinationen von Statusbits**: Wenn gar keine Statusbits in einem TCP-Header gesetzt sind, so deutet dies auf einen Angriff hin, da dies bei normaler Benutzung von TCP nicht vorkommt.
  Weiters sind Pakete verdächtig, bei denen SYN- und FIN-Bit gleichzeitig gesetzt sind, da dies normalerweise auch nicht vorkommt *(SYN wird beim Handshake für den Verbindungsaufbau benötigt, FIN aber beim Beenden der Verbindung)*. Ebenso sind ACK-Pakete, in denen die acknowledge number auf Null gesetzt ist, verdächtig.
- **Pakete für Dienste, die nicht angeboten werden**: Z.B. Pakete an Port 21 obwohl ftp-Dienst nicht aktiviert ist. Solche Pakete deuten auf einen Portscan hin.
- **Bekannte Angriffe:** Es können noch komprimierte Systeme von früheren Angriffen im Netz aktiv sein von denen immer noch eine Angriffsgefahr ausgeht. Ebenso können Angreifer natürlich auch bekannte Security Scanner einsetzen, um andere Systeme auf Schwachstellen zu untersuchen.

TCPDump:

Damit können die Datenpakete am Netzwerkinterface mitprotokolliert werden, und zwar wahlweise auf *stdout* oder in eine Datei. Man kann beim Programmaufruf auch einen Filterausdruck angeben, welche Pakete protokolliert werden sollen. Die nachfolgende Analyse der Protokolldatei kann mit *TCPDump* selbst oder mit *Etherreal* (siehe auch Abschnitt 1.2) erfolgen.

Snort:

(von Martin Roesch) Ist frei verfügbar u. deshalb relativ weit verbreitet mit einer recht aktiven Community. Einsatzbereiche für *Snort* sind

1. *Speichern das Datenverkehrs*, vergleichbar zu *TCPDump*.
2. *Vergleichen des Datenverkehrs* mit den konfigurierten Regeln.
3. *Auslösen von Aktionen*, wenn passende Regeln erkannt wurden.

Softwaretechnisch ist *Snort* folgendermaßen aufgebaut:

packet decoder	detection engine	logging and alerting
libpcap packet sniffing library		

- **packet decoder**: nimmt die Bits des Paketes entgegen und versucht eine Struktur zu erkennen bzw. eine Struktur auf die Bits abzubilden. Er arbeitet auf dem *data link layer* (Schicht 2).
- **detection engine**: unterstützt die Auswertung der Regeln. Nachdem die Struktur der Pakete erkannt wurde, müssen die Bedingungen für gefährliche od. verdächtige Pakete ausgewertet werden.
- **logging and alerting**: ist für das Speichern der Pakete zuständig. Es kann in Klartext oder im kompakteren *TCPDump*-Format abgespeichert werden. Das alerting kann als Log in die System-log-Datei oder eine eigene Logdatei erfolgen oder als Nachricht auf die Admin-Konsole.

Bei jedem gelesenen Paket werden die Regeln geprüft und im Falle einer passenden Regel der Aktionsteil aus dem Regel-Kopf ausgeführt. Danach werden für dieses Paket keine weiteren Regeln mehr geprüft. Die Reihenfolge der Regeln ist also wichtig. Außerdem sortiert Snort

die Regeln nach dem Aktionstyp – zuerst werden Regeln mit der Aktion *alert*, dann mit *pass* und danach mit *log*-Aktionen geprüft.
Folgende Aktionen sind erlaubt:

- **Alert**: Admin erhält Warnhinweis u. Paket(-Kopf) wird in Protokolldatei gespeichert.
- **Log**: Das Paket wird nur protokolliert.
- **Pass**: Lässt das Paket passieren.
- **Activate**: Aktiviert eine andere, sonst deaktivierte, Regel.
- **Dynamic**: Diese Regel bleibt solange deativiert, bis sie durch *Activate* aktiviert wird.

Der Vorteil von Snort gegenüber TCPDump liegt darin, dass auch der Paketrumpf untersucht und in die Regeln einbezogen werden kann.

Die wichtigsten Regeloptionen:

- **Msg Option**: es kann eine Message zu einer Regel definiert werden, die dann im Falle der Regelausführung mit in die Protokolldatei geschrieben wird.
- **TTL Option**: Angaben zur *Time To Live* eines Paketes. Bei einem traceroute von einem UNIX System aus kann durch sehr niedrige TTL für die UDP-Pakete der Weg des Paketes rekonstruiert werden, da jeder Router beim Ablauf der TTL eine Nachricht an den Absender sendet.
- **Dsize Option**: Damit kann eine Regel auf die Paketgröße definiert werden. Z.B. `dsize>1024` bewirkt, dass die Regel angewandt wird, wenn der Paketrumpf>`1kB` ist.
- **Flags Option**: Es wird getestet ob bestimmte TCP-Flags gesetzt sind.
- **Content Option**: Es kann ein String angegeben werden, dessen Vorkommen im Paketrumpf überprüft wird.

Weiter Infos unter www.snort.org .

Anmerkungen zum Einsatz von TCPDump u. Snort:

- Immer Header aller Pakete in eine Datei protokollieren. Damit sind nachträgliche Analysen und Beweissicherung möglich.
- Regeln sollte nachgelesen werden können um die nachträgliche Analyse und Regelanpassung zu erleichtern.
- Eigene Festplatte für Protokolldaten verwenden. Bringt Vorteile bez. Performance u. Speicherplatz.

3.3.5 Host based IDS

Zu den bekanntesten Host based IDS gehört *Tripwire* das frei verfügbar ist u. den meisten Linux-Distributionen beiliegt. Es prüft die Integrität von Dateien u. Verzeichnissen in einem UNIX-Dateisystem. Es wird also die Auswirkung eines Angriffs erkannt.
In der Konfigurationsdatei `tw.config` stehen die Informationen darüber, welche Dateien und Verzeichnisse überwacht werden sollen und speziell welche Attribute der Objekte dabei überwacht werden sollen *(in [] stehen die Attribut-Bezeichner)*:

- Zugriffsrechte [p]
- *inode*-Nummer [i]
- Anzahl der hard links auf diese Datei od. Verzeichnis [n]
- Besitzer (*user/group*) [u und g]
- Größe (s)
- Datum der letzten Änderung [m]
- Datum des letzten Zugriffs [a]

- verschiedene Hash-Werte der Datei

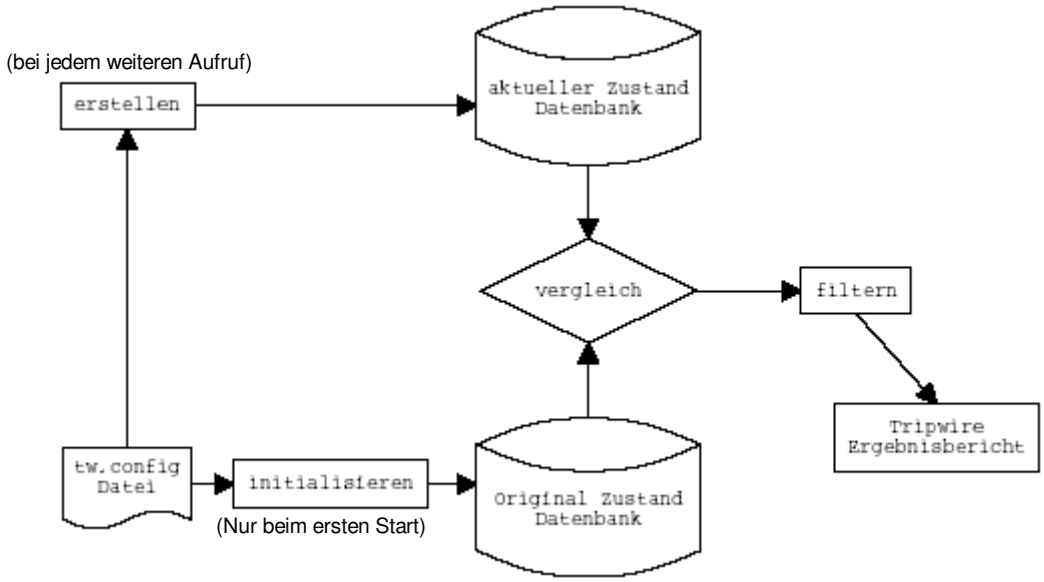

Abb. 3.11: Übersicht über die Funktionsweise von Tripwire

Tripwire kann in 4 verschiedenen Betriebsmodi laufen:

- **Database Initialisation Mode**: Es wird die erste (initiale) Version der Vergleichsdatenbank erstellt. Es werden dabei alle oben erklärten Attribute für alle zu überwachenden Objekte gelesen und in der Datenbank gespeichert.
- **Integrity Checking Mode**: Es werden die aktuellen Attribute gelesen, mit den gespeicherten Verglichen und Abweichungen angezeigt.
- **Database Update Mode**: Damit können die vom Admin absichtlich veränderten Objekte dem IDS bekannt gemacht werden. Es werden deren Attribute gelesen und in der Datenbank aktualisiert.
- **Interactive Database Update**: Es wird zuerst ein Integritäts-Check durchgeführt und danach kann der Admin für jedes als verändert erkanntes Objekt interaktiv angeben, ob der zugehörige Datenbankeintrag aktualisiert werden soll.

Anmerkungen zum Einsatz von Tripwire:

- Die initiale Erstellung der Vergleichsdatenbank kann je nach Dateisystemgröße einiges an Zeit beanspruchen und das System relativ stark auslasten.
- Aussagekraft der Ergebnisse hängt stark von der Vergleichsdatenbank ab, deshalb
  1. muss man bei der initialen Erstellung der Vergleichsdatenbank absolut sicher sein, dass das System in einem korrekten, nicht kompromittierten Zustand ist (z.B. direkt nach Systeminstallation)
  2. und dass die Datenbank so gespeichert ist, dass sie von einem Angreifer nicht manipulierbar ist (z.B. auf CDROM).
- Nach dem ersten Durchlauf sollte man anhand des Reports alle Objekte aus der Überprüfung herausnehmen, deren Änderung natürliche Ursachen hat (z.B. Cache-Dateien).

Weitere Infos unter www.tripwire.org .

3.3.6 Reaktion auf Alarm durch das IDS

Allgemein gilt, dass jede Reaktion auf einen Alarm mit Aufwand und Kosten verbunden ist. Deshalb ist es sinnvoll auf bestimmte Alarmsituationen das IDS automatische reagieren zu lassen.

Automatische Reaktionen:

3 Kategorien von automatischen Reaktionen:

1. **Genauer hinsehen was passiert**: Es werden zusätzliche Infos gesammelt indem man
   a. den Level der Protokollierung erhöht,
   b. nicht nur Paket-Header sondern komplette Pakete protokolliert,
   c. zusätzliche Filterregeln aktiviert,
   d. normalerweise deaktivierte Sensoren aktiviert.
2. **Verändern der eigenen Konfiguration**: Z.B.
   a. eine TCP-Verbindung durch senden von TCP-Reset Paketen beenden,
   b. Router, Firewall oder Rechner umkonfigurieren um Pakete des Angreifers zu blockieren -> ACHTUNG: Eventuell möchte Angreifer diese Umkonfiguration bewirken (er hat fremde IP-Adresse verwendet). Eine Prüfung durch den Admin zwingend notwendig.
3. **Etwas gegen den Angreifer unternehmen**: Den Angreifer einbremsen indem Ressourcen des Angreifers länger als nötig blockiert werden. Z.B. mit dem Programm tarpit. Bei Portscans reagiert tarpit indem es dem Angreifer eine gefälschte Antwort sendet. Will der Angreifer nun eine TCP-Verbindung aufbauen, so lässt tarpit den Verbindungsaufbau noch zu reagiert aber dann
   a. nicht mehr auf gesendete Datenpakete. Damit werden die Ressourcen des Angreifers so lange blockiert (er versucht Paket ständig zu wiederholen weil er von Paketverlust ausgeht), bis dieser aufgibt (timeout).
   b. oder indem es die Flusskontrolle unter TCP nutzt und dem Angreifer ständig einen vollen Empfangsbuffer meldet und so dessen Ressourcen blockiert.

Gegen Spammer gibt es eine mit 3. vergleichbare Technik, indem eine spez. Eigenschaft von SMTP genutzt wird. Es wird mit Fortsetzungszeilen die Übermittlung der Email vom Spammer blockiert (**Teergrube**).

Manuelle Reaktionen:

Sind nach einem erfolgreichen Angriff unumgänglich. Ein Experte muss die Art des Angriffs und das Schadensausmaß bestimmen. Danach muss der Schaden behoben und Vorkehrungen gegen zukünftige Wiederholungen des Angriffs getroffen werden.
Für eine planvolle und überlegte Vorgangsweise wird ein Prozess in 4 Schritten vorgeschlagen:

1. **Begrenzung (Containment)**:
   a. Das Problem entsprechend eingrenzen, so dass der Angriff nicht mehr fortgesetzt werden kann bzw. der Schaden sich nicht mehr vergrößern kann (z.B. Trennung der betroffenen Teile vom Netz. ACHTUNG: Schaden dabei nicht ungewollt vergrößern!)
   b. Komplette Sicherung des betroffenen Systems für spätere Analysen.
2. **Ausrottung (Eradication)**: Das Problem muss endgültig behoben werden. Dazu genau analysieren was passiert ist. Notfalls muss die gesamte Festplatte gelöscht werden um tatsächlich alle Manipulationen zu entfernen.
3. **Wiederherstellung (Recovery)**: Das System wird wieder betriebsbereit gemacht. Um nicht wieder die selben Schwachstellen zu haben, müssen zusätzlich zum Einspielen der letzten Sicherung Aktionen in folgenden bereichen erfolgen:

a. *Andere Systemkonfiguration wählen*, welche die erwähnten Schwachstellen nicht mehr beinhaltet (z.B. nicht benötigten Dienst sperren).
b. *Installation aktueller Software auf dem System*, wo die Schwachstellen behoben sind bzw. security patches einspielen.
c. *Änderung an ganz anderen Systemen* wie z.B. Router oder Firewall, die durch restriktivere Regeln einen solchen Angriff in Zukunft gar nicht mehr zulassen.

Bei der Rekonstruktion der Daten, die im Schritt *Begrenzung* gesichert wurden und dann der *Ausrottung* zum Opfer gefallen sind muss deren Korrektheit überprüft werden, da das System ja bereits kompromittiert war und dies natürlich auch die Daten betroffen haben könnte.

4. **Schlussfolgerungen (Lessons learned)**: Kritisches Überdenken was passiert ist. Ein schriftlichen Abschlussbericht erstellen mit Vorschlägen für zukünftige Verbesserungen.

4. Rahmenbedingungen und Software-Prozesse

4.1 Einführung

siehe Skript.

4.2 Gesetzliche Rahmenbedingungen

Das Internet ist weder für Anbieter von Diensten noch für den „normalen" User ein rechtsfreier Raum. Ale Beteiligten sind dabei an Gesetze und Verordnungen gebunden.

4.2.1 Allgemeine Vorschriften für Jedermann

unabhängig davon, welche Position oder Funktion ausgefüllt wird:

- Bürgerliches Gesetzbuch (BGB)
- Strafgesetzbuch (StGB)
- Gesetz zu allgemeinen Geschäftsbedingungen (AGB)
- Fernabsatzgesetz (FernAbsG)
- Verbraucherkreditgesetz (VerbrauKrG)
- Bundesdatenschutzgesetz (BDSG)
- Urhebergesetz (UrhG) (Achtung beim Einbinden fremder Grafiken und Texte über Links!)
- Signaturgesetz und Signaturverordnung (SigG und SigVO)

4.2.2 Vorschriften für ISPs

ISPs bieten den physischen Zugang zum Internet an. Sie sind vergleichbar mit den Telefongesellschaften und unterliegen deshalb auch ähnlichen Vorschriften und Gesetzen:

- Telekommunikationsgesetz (TKG)
- Telekommunikationsdatenschutzverordnung (TDSV)
- Telekommunikationsüberwachungsverordnung (TKÜV)
- Gesetz gegen unlauteren Wettbewerb (UWG)
- Handelsgesetzbuch (HGB)
- Gesetz zur Kontrolle und Transparenz in Unternehmen (KonTraG)

4.2.3 Vorschriften für Anbieter von Diensten

Dienste die neben dem reinen Zugang zum Netz angeboten werden, wie z.B. Information od. Unterhaltung, Mietsoftware od. Hostingangebote, usw.

- Teledienstgesetz (TDG)
- Teledienstdatenschutzgesetz (TDDSG)
- Mediendienste Staatsverordnung (MDStV)
- Produkthaftungsgesetz
- Handelsgesetzbuch (HGB)
- Abgabenordnung (AO)
- Jugendschutzgesetz (JuSchG)

4.3 Konstruktion sicherer Systeme

Da vollständige Sicherheit nicht erreichbar ist beschränkt man sich auf „hinreichend sichere" Systeme und versucht diese in einem rückgekoppelten, iterativen Entwicklungsprozess (analog zur SW-Entwicklung) zu erreichen. Typische Projektphasen sind dabei

- Analyse
- Design
- Implementierung
- Test
- Betrieb

4.3.1 Analyse

Spezifikation der funktionalen Anforderungen (requirement analysis), also was das System in welcher Betriebsumgebung für den Benutzer leisten soll. Ein Anforderungskatalog enthält also folgende Punkte:

- Funktionale (Leistung) Beschreibung
- Beschreibung der Daten bzw. Informationsstrukturen, mit denen das System arbeiten soll.
- Bedrohungs- und Risikoanalyse

Bedrohungsanalyse:

Welche Bereiche sind gefährdet und was verursacht die Gefährdung wie z.B.

- benutzerbedingte,
- technische oder
- organisatorische

Ursachen. Man beginnt häufig bei bekannten Bedrohungen vergleichbarer Systeme und weiteren für das eigenene System vorstellbaren Bedrohungen und leitet dann aus den gemeinsame Eigenschaften der einzelnen Bedrohungen die allgemeinen Gefährdungsbereiche ab (*Generalisierung*). Die Gefährdungsbereiche können in folgende Klassen unterteilt werden:

- **Externe Angriffe**
- **Interne Angriffe**
- **DoS**: Nicht erlaubte Benutzung von Ressourcen, so dass sie für die erlaubte Nutzung blockiert sind.
- **Abstreiten**: Wer hat Ressourcen benutzt und ist für die resultierenden Folgen verantwortlich.
- **Rechtemissbrauch**: wenn ein erlaubter User seine zugeordneten Rechte missbraucht.

Eine weitere Klassifizierung ist nach

- den bedrohten Schutzzielen,
- dem Auslöser der Bedrohung (Programmierer, interner User, usw.) oder
- der Ursache der Bedrohung (s.o.)

möglich.

Zur Beurteilung der Bedrohung gehört nun auch die Bestimmung der Eintrittswahrscheinlichkeit und des potentiellen Schadensausmaßes. Dies erfolgt in der

Risikoanalyse:

Dabei werden z.B. die Bedrohungen anhand der Kriterien Eintrittswahrscheinlichkeit und Schadensumfang in einer *probability impact* Matrix (siehe Kurs 1866, Abschnitt 4.5.1) eingetragen.
Für große Risiken werden Maßnahmen zur Risikominimierung erarbeitet, also zur Senkung der Eintrittswahrscheinlichkeit (Sicherheitsmaßnahmen) und/oder zur Schadensbegrenzung (z.B. Limite bei Online-Überweisungen).

Bei der Risikoanalyse ist insbesondere die Automatisierungsfähigkeit von Angriffen mittels Computer zu beachten (z.B. Fälschen von e-Money).

4.3.2 Design

Meist hierarchische Vorgehensweise, d.h. man beginnt mit einem Grobentwurf (Architektur) der dann in weiteren Schritten verfeinert wird. Mit der Systemarchitektur wird festgelegt, aus welchen Komponenten das System bestehen soll, was sie machen und wie sie miteinander agieren. Außerdem sollen folgende Sicherheitsanforderungen durch die Architektur erfüllbar sein:

- **Identifikation und Authentisierung**: Alle Systembenutzer (Personen u. Programme) müssen eindeutig identifizierbar sein. Art der Benutzerkennung u. des Authentisierungsmechanismus festlegen.
- **Rechtevergabe**: Unterstützung der Vergabe von Rechten und die Überprüfung der Einhaltung während des Betriebes.
- **Protokollierung**: Die wichtigsten Vorgänge im System müssen auch später noch nachvollziehbar sein.

E-Commerce Architektur:

In den meisten Fällen wird eine 3-Schichten-Architektur eingesetzt:

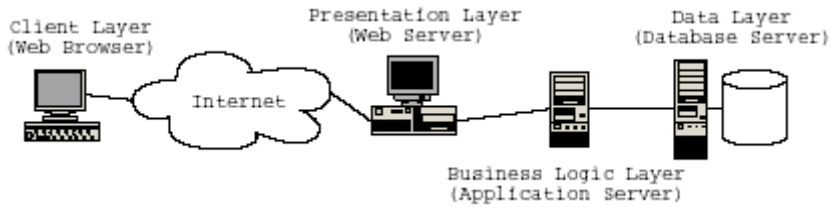

Abb. 4.1: 3-Schichten-Architektur im eCommerce. Client wird nicht mitgezählt.

- **Presentation Layer**: Wird durch einen Web Server realisiert und ist praktisch die Benutzerschnittstelle nach außen.
- **Business Logic Layer** (Application Server): Stellt die dynamischen Inhalte bzw. die Benutzerinteraktionen zur Verfügung, wie z.B. Auswählen der Produkte, Bezahlverfahren usw.
- **Data Layer**: Die Verwaltung der Benutzerdaten sowie die Daten über die Geschäftsobjekte werden in einem Datenbank-System verwaltet.

Die 3-Schichten-Architektur hat den großen Vorteil, dass sie leicht skalierbar ist, indem man bei Performance Problemen einen weiteren Server des betroffenen Systems hinzufügt.

Sicherheitsaspekte der E-Commerce Architektur:

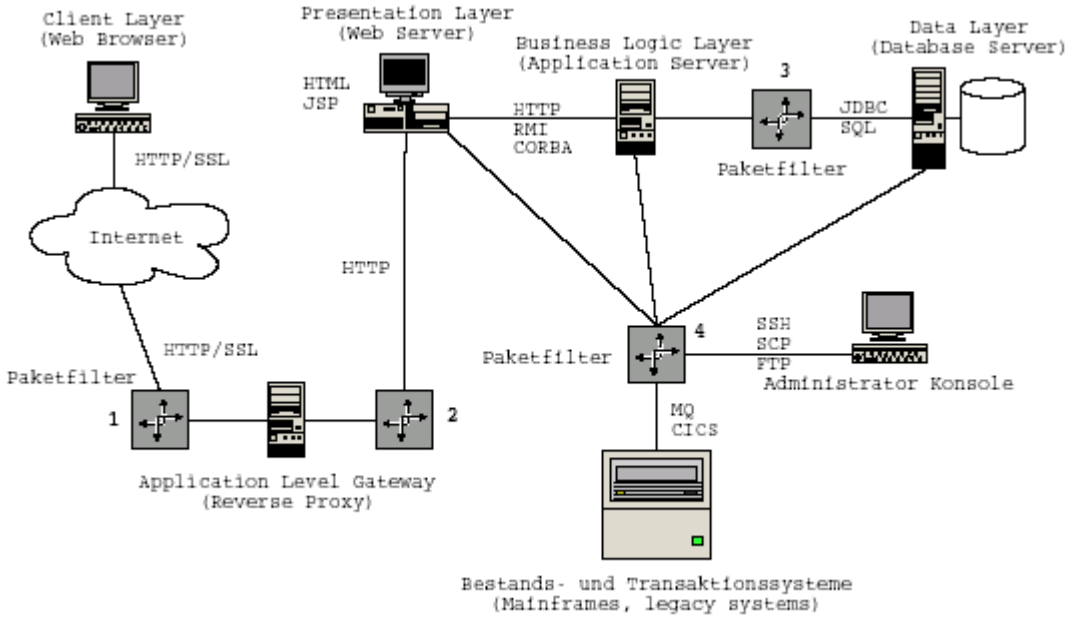

Abb. 4.2: Detaillierte Architektur der E-Commerce Anwendung.

Web Server (Presentation Layer): Der Web Server ist in Abb. 4.2 über eine mehrstufige Firewall mit einem Proxy-Server in der DMZ mit dem Internet verbunden. Diese Art der Internetanbindung hat folgende Konsequenzen:

- Paketfilter 1 lässt nur Pakete von Adressen außerhalb (des DMZ und des internen Netzes) an den Proxy durch.
- Paketfilter 2 lässt nur Pakete vom Proxy an Port 80 des Web Servers durch. Da man bereits im „privaten" Bereich der Infrastruktur ist, kann die Kommunikation bereits unverschlüsselt erfolgen.
- Der DNS-Name des Web Servers muss mit der IP-Adresse des Proxy hinterlegt sein.
- Soll der Client den Server authentisieren, so muss das Zertifikat auf den Proxy ausgestellt und auch dort hinterlegt sein.
- Da alle Pakete über die DMZ laufen, ist die Performance der Paketfilter u. des Proxy absolut wichtig.

Application Server (Business Logic Layer): Die Kommunikation zw. Web Server u. Application Server erfolgt direkt über das LAN, erst beim Zugriff auf Daten wird wieder ein Paketfilter eingesetzt, der nur Pakete an die Datenbank-Ports des Database Servers durchlässt. Damit kann verhindert werden, dass z.B. ein Angreifer vom Web Server aus eine telnet-Verbindung zum Database Server aufbaut. SQL Befehle vom Application Server aus können damit aber nicht verhindert werden.

Backend System / Administration: Die Kommunikation für Transaktionen die am Backend System ausgeführt werden müssen wird durch Paketfilter 4 überwacht. Ebenso der Zugriff durch den Administrator oder für das Content Management System.

Weitere Sicherheitsaspekte:

- Einrichtung von Benutzer und Gruppen mit geeigneten Rechten auf allen Rechnern. Dies gilt nicht nur für das OS sondern auch für sensible Anwendungen wie z.B. Datenbankzugriff.
- ***Directory Services*** für die konsistente und einheitliche Verwaltung der Daten einsetzen. Mit ***LDAP (Lightweight Directory Access Protocol***) steht ein einheitliches

Protokoll für den Zugriff auf Directory Services zur Verfügung. Die Paketfilter müssen diese *LDAP* Zugriffe auf den Directory Server zulassen.
- *IDS* zur Überwachung des Netzwerkverkehrs einsetzen.
- Protokollierung der wichtigsten Ereignisse (v.a. Buchführung) auf einem zentralen *Log Server*. -> anpassen der Filterregeln der Paketfilter. Eventuell Protokollierung über ein paralleles Netz wegen der hohen Netzlast.
- Eigener *Zeit Server* zur Synchronisierung der System-Uhren der beteiligten Rechner z.B. über ***NTP (Network Time Protocol)***, damit die Logs mit korrekten Uhrzeiten versehen sind.

4.3.3 Implementierung

Prozesse:

Häufig wendet man bei der Entwicklung komplexer Systeme eine evolutionäre Vorgangsweise an. Dabei wird eine Serie von Prototypen (Versionen) entwickelt, die dann letztlich direkt in die Endversion des Systems konvergieren. Für jede Version wird der volle Entwicklungszyklus durchlaufen, die ersten Versionen erreichen jedoch i.d.R. nicht die Phase Betrieb u. Wartung sondern dienen lediglich zur Evaluierung und als Basis für die Verfeinerung der weiteren Versionen. Es wird also z.B. eine erste Version entwickelt, in der nur die wichtigsten Grundfunktionen enthalten sind. Mit der Anforderungsspezifikation der 2. Version kann man dann schon beginnen, wenn die 1. Version in der Design-Phase ist und die Erkenntnisse der Analyse-Phase der 1. Version dabei nutzen.

Umgebungen:

Folgende Umgebungen werden für die Implementierung benötigt:
- **Produktionsumgebung**: In dieser Umgebung läuft der Betrieb des Systems. Sie ist an das Internet und an das Produktions-Back-End angeschlossen u. besteht im wesentlichen aus den Servern der 3-Schichten-Architektur.
- **Testumgebung**: Um beim Test der Folgeversion nicht den Betrieb der vorhergehenden Version zu stören ist dafür eine eigene Testumgebung notwendig, welche der Produktionsumgebung möglichst ähnlich sein sollte. Sie ist allerdings nicht an das Internet angeschlossen und kann auch nur auf Test-Backend-Systeme zugreifen.
- **Entwicklungsumgebung**: Ist mit typischen Entwicklertools und einer eingeschränkten Testumgebung *(möglichst auf der Entwicklermaschine wo dann auch die entsprechenden Server-Dienste laufen)* ausgestattet, sodass die ersten Tests während der Entwicklung nicht die eigentliche Test- bzw. Produktionsumgebung der Vorgängerversionen stört. Zusätzlich werden noch System für die Versionskontrolle u. Konfigurationsmanagement benötigt.

Die Aufteilung in mehrere Umgebungen ist aus Gründen der Sicherheit absolut erforderlich!

Fehler:

Viele Sicherheitsprobleme haben ihre Ursache in Programmfehler (z.B. buffer overflow exploits). Da ein Beweis der Korrektheit eines Programms in der Praxis zu Aufwändig ist und man dabei auch prüfen müsste ob der Beweis selbst auch fehlerfrei ist, wird versucht, durch bestimmte Methoden eine möglichst geringe Anzahl an verbleibenden Fehlern in einer Software zu erreichen:
- **Testen**: Es wird versucht, mit verschiedenen Teststrategien möglichst viele Fehler in einem Programm zu finden. Anschließend werden die gefundenen Fehler behoben und erneut getestet bis keine Fehler mehr gefunden werden.

- **Warnungen einschalten**: Alle Warnungen des Compilers einschalten.
- **Selbst überwachende Software**: Integration der Überprüfung von kritischen Werten und Zuständen in der Software.
- **Debug-/Release-Version**: Mit z.B. Preprozessoranweisungen in C spezielle Ausgaben und Prüfungen einfügen die nur durchgeführt werden, wenn das Programm im Debug-Mode compiliert wurde. Im Release-Mode sind diese Tests nicht enthalten und damit wird auch die Abarbeitungsgeschwindigkeit nicht beeinflusst. ACHTUNG: Timingprobleme können so eventuell aber erst in der Release-Version auftreten.
- **assert Anweisungen**: In C kann mit den assert Makro eine Fehlermeldung generiert werden, wenn eine zu prüfende Bedingung nicht erfüllt wird. Dieses Makro wird vom Compiler aber nur im *Debug-Mode* expandiert und ist damit im *Release-Mode* unwirksam. Mit *assert* können Überprüfungen von z.B. Vor- u. Nachbedingungen, Schleifeninvarianten, Datenkonsistenz, usw. durchgeführt werden. *assert* darf aber nicht unter folgenden Situationen eingesetzt werden:
  - Damit dürfen aber nur logische Programmierfehler abgefangen werden *(z.B: NULL-Pointer, Indexüberschreitung bei Feldern)*, nicht Fehlersituationen, die im Betrieb wirklich auftreten können.
  - Bei öffentlichen Funktionen u. Methoden (z.B. aus Library) muss die Prüfung von z.B. Parameterwerten auch im *Release-Mode* stattfinden *(also nicht mit assert!)*, da eine böswillige Verwendung von Library-Funktionen für einen Angriff immer mit in Betracht gezogen werden muss.

4.3.4 Test

Zum Auffinden von Fehlern in Programmen werden hauptsächlich folgende Teststrategien eingesetzt:

- **unit test**: Es wird geprüft, ob eine Komponente gemäß Spezifikation arbeitet. In der Java Umgebung gibt es mit *JUnit* ein spezielles *unit test* System das bei der Erstellung und automatischen Ausführung von *unit tests* unterstützt. Die Idee dabei ist, dass bereits nach der Definition der Schnittstelle einer Methode aber vor der Implementierung des Methodenrumpfes bereits ein Test *(in Form eines Java Programms)* für die Methode entworfen wird. Damit kann der Test später automatisch ausgeführt werden. -> www.junit.org .
- **integration test**: Es wird geprüft, ob alle Komponenten gemäß Spezifikation zusammenarbeiten.
- **performance test, load test**: Erfüllt das System die Anforderungen an Ausführungsgeschwindigkeiten und Speicherplatzbedarf?
- **user acceptance test**: Kommen die Benutzer mit dem Programm zurecht?
- **regression test**: Bei Änderungen am System oder Programm werden alle bisherigen Tests und die durch die Änderungen oder Erweiterungen neu hinzugekommenen Testfälle am Prüfling ausgeführt um somit sicher zu gehen, dass nach der Änderung die „alten“ Funktionen noch so arbeiten wie geplant.

Spezielle Tests zur Sicherheit eines Systems sind

- **Penetrationstests**: Es wird versucht die Sicherheit eines Systems zu überprüfen indem verschiedene Angriffe auf das System durchgeführt werden. Sie sollten nicht von den Herstellern des Systems durchgeführt werden, da diese die Entstehungsgeschichte des Systems kennen und somit nicht unvoreingenommen die Tests durchführen kennen. Den Testern sollten alle Informationen über das System zur Verfügung gestellt werden, da sie dann bessere Erfolgsaussichten haben, versteckte Schwachstellen des Systems zu finden.

- **Audits und Zertifizierungen**: (Externe) Experten Analysieren Systeme und Abläufe und verfassen dazu eine Beurteilung.
    - Dabei wird meist nicht das System selbst sondern der *Prozess* der Systemerstellung betrachtet (z.B. ISO9000). Die Mindestanforderungen, die an einen solchen Prozess gestellt werden sind
        - Dokumentation des Prozesses
        - Überwachung und Einhaltung der Prozessschritte
        - Dokumentation der Anforderungen an das System
        - Dokumentation der Entwurfsergebnisse

      Die Einhaltung der Mindestanforderungen garantieren aber noch lange nicht qualitativ gute Systeme!
    - Daneben gibt es aber auch Zertifizierungen für die *Systeme* selbst, die nach bestimmten Kriterienkatalogen erfolgen wie z.B.
        - *TCSEC – Trusted Computer System Evaluation Criteria (USA 1985)*, womit die Sicherheit von Systemen gemessen und in verschiedene Sicherheitsklassen eingestuft werden kann.
        - *ITSEC – Information Technology Security Evaluation Criteria (EU 1995)*, welche eine Auflistung von Funktionsklassen enthält, von denen jede bestimmte Sicherheitsgrundfunktionen beschreibt. Daneben werden auch Mechanismen zur Umsetzung der Funktionen betrachtet, die in 3 Stufen von niedrig bis hoch bewertet werden. Insgesamt gibt es 7 Evaluationsstufen, wobei eine höhere Stufe mehr Vertrauen in die Korrektheit der Funktionen bedeutet.
        - *Common Criteria*, ein Projekt zur Vereinheitlichung der verschiedenen Zertifizierungskriterien zu einem weltweiten Standard.

      Alle Zertifizierungen beziehen sich aber immer nur auf ein konkretes Produkt in einer konkreten Konfiguration und Umgebung. Wird daran etwas geändert, so muss die Zertifizierung wiederholt werden.

4.3.5 Betrieb

Um einen sicheren Betrieb zu gewährleisten sind folgende Punkte wichtig:

- Ordnungsgemäßer Ablauf des normalen Betriebs
- Beheben von kleineren Problemen durch den Admin
- Erkennen von ungewöhnlichen Aktionen (IDS)
- Abwehr von Angriffen aller Art (Firewall)
- Kontrollierte Aktualisierungen des Systems (updates)

Daraus ergeben sich 2 Aspekte:

Technische Aspekte:

- wie meldet sich der Admin sicher am System an (z.B. ssh)
- der Admin sollte sich nicht als Administrator anmelden sondern als „normaler“ user und nur für die einzelnen Operationen mit Administratorrechten arbeiten (z.B. mittels `sudo` unter UNIX). -> einfachere Protokollierung, welcher Admin was getan hat.
- Für einen sicheren 24-Stunden Betrieb sollte die Produktionsumgebung entsprechend redundant (Stromversorgung, Internet, usw.) ausgestattet sein (z.B. Rechenzentrum).
- Physischer Zugang zu den Systemen sollte gesichert sein.
- Rest siehe Kurs 1866 Abschnitt 4.2, 4.4

Organisatorische Aspekte:

- Einzelne Verantwortlichkeiten sollten klar geregelt und dokumentiert sein, und zwar im geschäftlichen wie im technischen Bereich. Z.B. ist der Geschäftsbereich zuständig für die Beauftragung/Abnahme von Änderungen/Erweiterungen und für die Risikoanalyse, während der technische Bereich die Änderungen/Erweiterungen vorschlägt und die technischen Risiken abklärt.
- Regelmäßige und korrekte Erstellung von Backups
- Genau definierter Prozess für Änderungen oder Erweiterungen am System. Damit soll erreicht werden, dass
    - die Verfügbarkeit des Systems,
    - die Integrität der Daten und
    - die Vertraulichkeit der Daten

  weiter gewährleistet bleibt. Eine Änderung sollte also wie eine Neuentwicklung betrachtet werden. Abweichungen von diesem Prozess sollten nur in Notfällen möglich sein (z.B. DoS Angriff).

Anhang A – Prüfungsfragen Kurs 1867

Zusammengestellt aus Gedächtnisprotokollen von Studienkollegen aus dem Zeitraum Mai 2003 bis Oktober 2004. Die Gedächtnisprotokolle sind auf den Seiten des Fachschaftsrat erhältlich.

1. Wie kann ich denn nun vertrauliche Daten abhören? *[1x Keller]*

2. Und wenn ich keinen Zugang zu einem DNS-Server habe? *[1x Keller]*

3. Angriffe auf Verschlüsselung? *[1x Kern-Isberner]*

4. Wieso und wie kann man den RSA brechen? *[1x Kern-Isberner]*

5. Was sind Replay Angriffe und wie kann ich sie verhindern? -> *siehe Kurs 1866, Verschlüsselungsmodi [1x Keller]*

6. Wie noch außer mit CBC? *(z.B. mit PIN u. TAN) [1x Keller]*

7. Wie kann ich Zugriff auf einen entfernten Rechner erhalten? *[1x Kern-Isberner]*

8. Was sind VPN und wie funktionieren sie? *[1x Kern-Isberner]*

9. Warum sind Remote-Verbindungen mit telnet gefährlich? *[1x Kern-Isberner]*

10. Wie erkenne ich einen Angriff? *[1x Keller]*

11. Eine Firma möchte ihr Netzwerk ans Internet anschließen. Was muss gemacht werden damit Angriffe so früh als möglich erkannt werden? *[1x Keller]*

12. Was sind IDS? *[1x Kern-Isberner]*

13. Welche Typen von IDS gibt es? *[2x Keller] [1x Kern-Isberner]*

14. Wie funktionieren sie? Wie wird was erkannt? *[1x Keller] [1x Kern-Isberner]*

15. Welcher der beiden IDS-Typen ist besser dazu geeignet, Evidenz anzulegen, also was ist besser als forensischer Nachweis bei einem späteren Schadensprozess geeignet? *[1x Keller]*

16. Wie kann ein Hacker ermittelt werden? *[1x Keller]*

17. Welcher Gesetze sind für Nutzer und Anbieter im Internet relevant? *[1x Keller]*

18. Detaillierte Architektur einer eCommerce Anwendung? *[1x Kern-Isberner]*

19. Was macht man, wenn man eine Attacke feststeht? *[1x Kern-Isberner]*