it-sec

March 6, 2021

Contents

1 Stoff

1.1 Vorlesungen

1.1.1 1 Vorlesung (Einführung)

1. relevante Daten in der IT-Sicherheit
   (a) valuable data
      i. Vorderseite Was ist *valuable data*?
      ii. Rückseite Daten die mit monetärem Wert vebunden sind. Das sind Kreditkartendaten, Kryptowährungen, Amazon / Ebay Accounts, aber auch der Netflix-Account. Angreifer können damit Geld verdienen.
   (b) private data

i. Vorderseite Was ist 'private data'?
ii. Rückseite Gesundheit, Familie, Lebensplanung. Daten, die man nicht ins Internet stellen würde. Bergen die Gefahr der Erpressung. Diese Daten gehören zum persönlichen Schutzraum.

(c) dangerous data
i. Vorderseite Was ist 'dangerous data'?
ii. Rückseite Daten, die Einfluss auf physikalische Prozesse haben. Steuerung von Maschinen (Atomkraftwerken, Fahrzeugen). Ein Angriff auf diese Daten birgt Lebensgefahr.

2. Sichere Soft- und Hardware zu entwickeln ist schwierig Weil:
   (a) Komplexität der Systeme Je komplexer das System, desto schwieriger ist dessen Schutz.
   (b) Entstehung dieser Systeme Werden meist von Unternehmen entwickelt, die einer Vielzahl von Constraints unterliegen und daher bei Sicherheit sparen.
   (c) Bekannte Schwachstellen aus der Vergangenheit
      i. Vorderseite Nennen sie 3 bekannte Schachstellen aus der Vergangenheit
      ii. Rückseite
         A. Heartbleed Schwachstelle in OpenSSL. Ermöglichte das Mitlesen von Daten auf dem Transportweg.
         B. Shellshock Schachstelle in Bash, die 30 Jahre lang bestand.
         C. Spectre & Meltdown Hardware-Schwachstellen in Prozessoren. Spekulative Ausführung kann genutzt werden, um Einfluss auf Berechtigungen und den CPU-Cache zu nehmen.

3. Cybercrime Schaut auf Endnutzer, nicht mehr auf IT-Systeme / Server.
   (a) Skilled Attackers
      i. Vorderseite Was sind *skilled attackers*?
      ii. Rückseite **Professionelle** und gut ausgestattete Angreifer (meistens *state sponsored*), die gezielte Angriffe gegen Wirtschaft und Staat führen mit dem Ziel der Erpressung, Spionage, Sabotage.
      Beispiel: Stuxnet Worm. Angriff aus dem Jahre 2010 auf iranische Atomanlagen (vermutlich duch us-amerkanische und israelische Geheimdienste).

4. Sicherheitsziele und Bedrohungen Bergriffsklärung

   (a) Sicherheitsziele der IT Sicherheit

      i. Vorderseite Nennen und erklären Sie 3 grundlegende und 3 weitere Ziele der IT Sicherheit

      ii. Rückseite

         A. Grundlegende C.I.A.

         B. Vertraulichkeit (Confidentiality) Absicherung von Ressourcen gegen Enthüllung / Offenlegung. Abhören von Telefonie, Mitlesen der Emails. Gängigstes Gegenmittel ist die Verschlüsselung. Verstecken ist auch möglich, aber nicht effektiv.

         C. Integrität (Integrity) Absicherung von Ressourcen gegen unbefugte Manipulation. Z.B. Abänderung von Überweisungsdaten (Änderung des Empfängers), Abänderung von Dateien in meinem Dateisystem.
         Es muss nicht sein, dass Angriffe gegen die Integrität auch die Vertraulichkeit stören. Sie sind dann blinde Angriffe.
         Gegenmittel: Authorisierung prüfen, Prüfsummen und digitale Fingerprints. Am besten auf einem anderen Kanal schicken.

            Wer ist Mario und von welchem Zeug hat er 10kg besorgt? xD

         D. Verfügbarkeit (Availibility) Absicherung gegen die Störung der Ressourcen.
         Z.B.
         - Versucht jemand den Webserver zu crashen, damit eine Website nicht mehr erreichbar ist?
         - Versucht ein Angreifer meine Festplatte zu formatieren?

         Beliebt in Verbindung mit Erpressung. Das Abhandensein des Online-Shops führt zu Geldeinbußen.

         E. Absicherung
         - Einschränkung der Verfügbarkeit (Authorisierung).
         - Redundanz: Mehrere Server
         - Diversität: Verschiedene Server

         F. Weitere

         G. Authenticity (Authentizität) Der Urheber der Informationen ist, wer er vorgibt. Könnte als Teil der Integrität gezählt werden. Integrität des Absernders / Erstellers.

H. Accountability (Verantwortlichkeit) Das Verknüpfen von Handlungen und Nutzern. Wichtig für Forensik. Wer hat was gemacht? Wer ist schuld? Ist teilweise personalrechtlich verboten.

I. Privacy (Privatsphäre) Sicherheit und Kontrolle von persönlichen Daten. Hier gelten die gleichen Mechanismen wie oben (Vertraulichkeit, Integrität, Verfügbarkeit), aber die Daten sind personenbezogen. Eigener Zweig der IT-Sicherheit.

(b) Abgeleitete Begriffe

i. Threat

A. Vorderseite Definieren Sie *threat*.

B. Rückseite Verletzung eines Sicherheitsziels.

ii. Security

A. Vorderseite Definieren Sie *Security*.

B. Rückseite Absicherung gegen **absichtliche** Threats. Z.B. Die Bremse meines Autos soll nicht gehackt werden.

iii. Safety

A. Vorderseite Definieren Sie *Safety*.

B. Rückseite Absicherung gegen **versehentliche** Threats (Unfälle). Z.B. Die Bremse meines Autos soll funktionieren und nicht ausversehen kaputt gehen, oder falsch benutzt werden können.

(c) Angriffsklassen (threat classes)

(d) Unsere Definition von Angriffsklassen

i. Vorderseite Wie schauen wir auf Angriffsklassen?

ii. Rückseite Angriff: Alles, was **absichtlich** ein **Sicherheitsziel** zu verletzen versucht. Anstatt alle möglichen Angriffe aufzuzählen, sollten wir eher auf die Sicherheitsziele schauen.

(e) Threat Classes

i. Vorderseite Was sind *threat classes*? Nennen und definieren Sie 4 *threat classes*. (Muss laut Prof nicht auswendig gelernt werden).

ii. Rückseite *Threat classes* sind die Arten auf welche Sicherheitsziele verletzt werden können.

A. Disclosure Dinge werden an dritte weitergegeben. Unauthorized Access.

B. Deception Täuschung. Hat die Form falscher Daten. Absender des IP Pakets ist falsch.

C. Disruption Störung des Betriebs z.B. eines Webserver.

D. Usurpation Unauthorisierte Übernahme der Kontrolle. Beispiel: Selbstfahrendes Auto wird durch Hacker ferngesteuert.

5. Sicherheitsmechanismen und Policies

(a) Definition Sicherheitsmechanismen und Policies

i. Vorderseite Erklären Sie den Unterschied zwischen Sicherheitsmechanismen und -policies

ii. Rückseite Wir unterscheiden zwischen Mechanismen und Policies (Strategien, Gesetze).

- Policy ist ein Gesetz, das definiert, was erlaubt ist und was nicht.
- Der Mechanismus ist ein Werkzeug, welches die Policy umsetzt.

Es kann sein, dass etwas nicht erlaubt ist (Policy), aber der Mechanismus das Verbot nicht perfekt umsetzt.

Z.B. will man keinen Spam, aber die Spamfilter können das nicht perfekt umsetzten (lassen Spam durch oder filtern Ham raus).

(b) Arten von Mechanismen

i. Kreislauf der Sicherheitsmechanismen

A. Vorderseite Nennen und definieren Sie die 3 Phasen des niemals endenden Sicherheitskreislaufs.

B. Rückseite

C. Prevention (Prävention) Oberstes Ziel. Es soll keine Sicherheitsvorfälle geben. Damit beschäftigt man sich **vor** einer Verletzung der Sicherheitsziele. Z.B.:

- Man verwendet Verschlüsselung, damit niemand die Mails mitlesen kann.
- Man verwendet Authentifikation
- Man verwendet Ristriktion von Zugriffsrechten.

Bietet aber keinen 100%igen Schutz (falsche Benutzung, Sicherheitslücken). In anderen Fällen ist der Einsatz einiger Mechanismen nicht möglich,

- z.B. wenn man ein offenes System (z.B. Internetsuche) betreibt. Man soll es nutzen können, ohne sich vorher anzumelden.
- z.B. muss die Bremse im Fahrzeug ohne Passwort funktionieren.

Daher:

D. Detection (Erkennung) Wenn die Prävention versagt, soll trotzdem der Angriff erkannt werden. Reagiert wird **während** der Verletzung des Schutzziels. Funktioniert nur, wenn man die Angirffsart kennt:

- Man scannt Mails nach Bedrohungen, aber sie kommen über eine Website.

Z.B.:

- Virenscanner: Reagiert, wenn die Schadsoftware bereits im System ist.

Daher:

E. Analysis (Analyse) Ein Sicherheitsvorfall ist bereits aufgetreten. Wie ist das passiert? Dient nur der zukünftigen Absicherung.

F. Beispiel Wie beim Zahnarzt: Prävention: Zähneputzen Detection: Zahnarztbesuch zur Überprüfung Analysis: Stelle wird gefunden und entfernt. "Bitte putzen Sie besser, benutzen Zahnseide und nehmen eine andere Körperöffnung zur Essensaufnahme".

1.1.2 2 Vorlesung (Symmetrische Kryptographie)

1. Cryptography

   (a) Vorderseite Definieren Sie den Begriff *Cryptography*.

   (b) Rückseite Die Wissenschaft die *Confidentiality* und *Integrity* von Information zu schützen. Die Information wird nicht versteckt, also ist von außen sichtbar, dass Informationen vorhanden sind. Es muss davon ausgegangen werden, dass das Verschlüsselungsverfahren bekannt ist.

2. Cryptanalysis

   (a) Vorderseite Definieren Sie den Begriff *Cryptanalysis*.

   (b) Rückseite Die Wissenschaft der *Angriffe* gegen die Kryptographie.

3. Steganography

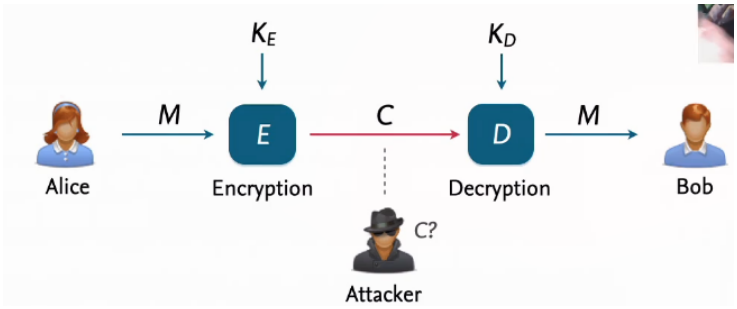
   (a) Vorderseite Definieren Sie den Begriff *Steganography*. In welcher Situation ist *Steganography* relevant?

   (b) Rückseite Die Wissenschaft des *Versteckens* von Information. Wichtig, wenn Kommunikaiton *nicht observierbar* oder *abstreitbar* sein muss.

4. Cryptosystems

   (a) Vorderseite Skizzieren Sie den Weg von Nachrichten innerhalb eines *Cryptosystems*. Erläutern sie auch die jeweiligen Begriffe.

(b) Rückseite



M := *Plaintext* Nachricht C := *Cyphertext* Nachricht K_E := *Encryption Key* K_D := *Decryption Key*

5. More on Cryptosystems

   (a) Cipher

      i. Vorderseite Definieren Sie den Begriff *Cypher*.

      ii. Rückseite Funktionen für das *Ver-* und *Entschlüsseln* von Nachrichten. $E(M, K_E) = C$ $D(C, K_D) = M$

   (b) Keyspace

      i. Vorderseite Definieren Sie den Begriff *Keyspace* eines kryptographischen Schlüssels. Worin unterscheidet sich hierbei *symmetrische* und *public-key* Kryptographie?

      ii. Rückseite Die Menge aller möglichen Werte für K_E und K_D *Symmetrisch*: $K_E = K_D$ *public-key*: $K_E \neq K_D$

   (c) Confusion Property

      i. Vorderseite Was ist die *Confusion Property* einer kryptographischen *Cypher*?

      ii. Rückseite Sind *Cyphertext* und *Plaintext* vorhanden, muss der *Key* schwer ableitbar sein.

   (d) Diffusion Property

i. Vorderseite Was ist die *Diffusion Property* einer kryptographischen *Cypher*?

ii. Rückseite Es muss schwer sein den *Plaintext* aus dem *Cyphertext* abzuleiten.

(e) Prinzip von Kerckhoffs

i. Vorderseite Was ist das *Kerkhoffs'sche* Prinzip in der Verschlüsselung?

ii. Rückseite

- Es muss davon ausgegangen werden, dass die *Cypher* dem Angreifer bekannt ist
- Die Sicherheit darf nur von der Verschlüsselung abhängen (no obscurity). **Allein die Größe** des Schlüssels ist das Maß für Sicherheit.

(f) Warum ist Obscurity schlecht?

i. Vorderseite Ist an *Security through obscurity* sinnvoll?

ii. Rückseite Nein, denn nach einer langen Nutzung des Verfahrens (evtl. durch mehrere Personen) kann man nicht mehr abschätzen, wie weit sich das Geheimnis der Obskurität verbreitet hat.

6. Some Attack Types

(a) Brute-force attack

i. Vorderseite Was ist eine *Brute-force* Attacke auf ein Cryptosystem?

ii. Rückseite Alle möglichen Schlüssel des Keyspaces ausprobieren. Der Angriff wird durch die Größe des Keyspaces gebremst.

(b) Purchase-key attack

i. Vorderseite Was ist eine *Purchase-key* Attacke auf ein Cryptosystem?

ii. Rückseite Der Key wird durch Androhung von Gewalt, Diebstahl, Erpressung, Bestechung oder andere wenig subtile Methoden erlangt. Daher ist physikalische Sicherheit nicht zu vernachlässigen.

(c) Cyphertext-only attack

i. Vorderseite Was ist eine *Cyphertext-only* Attacke auf ein Cryptosystem?

ii. Rückseite Anwendung eines Verfahrens zur Ableitung des Schlüssels ausschließlich aus dem *Cyphertext*.

(d) Known-plaintext and chosen-plaintext attack

i. Vorderseite Was ist eine *Known-plaintext* bzw. *Chosen-plaintext* Attacke auf ein Cryptosystem?

ii. Rückseite Man hat Paare von unverschlüsselten Nachrichten und ihren verschlüsselten Versionen. Im *known* Fall hat man eine bekannte Anzahl dieser Paare, im *chosen* Fall kann man für beliebige Nachrichten verschlüsselte Texte generieren. Hier wird auch die Rekonstruktion des Schlüssels versucht.

7. Estimating Risk

(a) Ressourcen bei Angriff auf Cryptosystem

i. Vorderseite Nennen Sie die Ressourcen für einen Angriff auf ein Cryptosystem.

ii. Rückseite

- Computation Power
- Speicherplatz
- Zeit

- (Skill)

(b) Wann ist ein Verschlüsselungsverfahren *computationally secure*?

i. Vorderseite Wann ist ein Verschlüsselungsverfahren *computationally secure* und was heißt das?

ii. Rückseite

- Wenn der Trade-off zwischen den Kosten der Entschlüsselung und dem möglichen Profit negativ ist. Zu den Kosten der Entschlüsselung gehören z.B. die Kosten für die Rechenkapazität.
- Kryptographische Verfahren die mit den aktuell verfügbaren Ressourcen und den aktuell bekannten Angriffen nicht brechbar sind.

8. Ancient Ciphers

(a) Simple substitution ciphers

i. Defintion

A. Vorderseite Was ist eine simple *substitution cipher* und was ist seine Schwäche?

B. Rückseite

- Ein Buchstabe einer Nachricht wird nach einer festen Verschiebung auf einen anderen Buchstaben gemappt.
- Der Keyspace ist sehr klein.

ii. Caesar Cipher

A. Vorderseite Was ist die *Caesar* Cipher und wie groß ist ihr Keyspace?

B. Rückseite

- simple substitution cipher, Verschiebung des Alphabets um k Stellen. Verschiebung um 0 und um |Alphabet| ist die Indentitätsverschiebung (keine Verschiebung).
- Der Key ist die Verschiebung k, daher ist der Keyspace |Alphabet| - 2.

iii. ROT13

A. Vorderseite Was ist die *ROT13* Cipher und wie groß ist ihr Keyspace?

B. Rückseite

- simple substitution cipher, Spezialfall der Caesar Cipher, Verschiebung eines Buchstabens im Alphabet um 13 Stellen.
- Wird die Verschiebung als der Key angesehen, ist einer Verschiebung um 13 Stellen der Keys selbst, also ist der Keyspace = 1

(b) Monoalphabetic substitution ciphers

i. Vorderseite Was ist eine *monoalphabetic substitution cipher* mit Permutation und was ist seine Schwäche?

ii. Rückseite

- Zu dem Alphabet der Nachricht wird ein gleichlanges anderes Alphabet gewählt, z.B. durch Permutation. Die Buchstaben werden 1:1 gemappt.
- Frequenzanalyse: Man zählt Buchstaben nach einer bekannten Frequenz und nimmt ein Wörterbuch der vermuteten Sprache und versucht sie zu mappen...

(c) Vignère Cipher (Polyalphabetic substitution cipher)

i. Vorderseite Was ist die *Vignère Cipher* und was sind bekannte Angriffe auf ihn?

ii. Rückseite

- Ein Buchstabe wird auf einen Buchstaben des gleichen Alphabets gemappt, jedoch ist der Offset nicht mehr fest, sondern wird von einem *Running Key* festgelegt. Effektiv sind es mehrere einfache Substitutionschiffren in einender.
- Kasiski Test, Friedman Test: Die Verschiebnung hat eine Periode, die man herausfinden kann. Sie bestimmt die einzelnen monoalphabetischen Substitutionen, die dann zu ermitteln sind.

(d) XOR Cipher

i. Vorderseite Wie funktioniert die *XOR Cipher*?

ii. Rückseite

A. Nachricht und Key auf Binärcode mappen.

B. Key auf die Länge der Nachricht periodisieren.

C. E(M, K) = xor(M, K) = C D(C, K) = xor(C, K) = M

(e) One-Time Pad

i. Vorderseite Was ist die *One-time pad* Cipher und warum ist sie absolut sicher?

ii. Rückseite *XOR Cipher* mit Constraints:

A. Die Länge des Keys und der Nachricht ist gleich.

B. Die Bits des Keys sind echt zufällig.

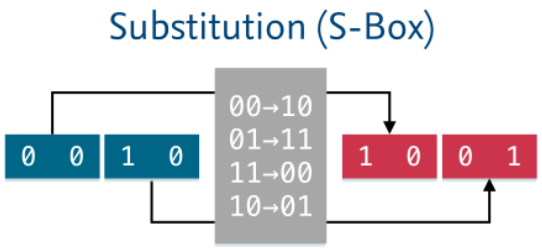
C. Der Key wird nur einmal benutzt und dann zerstört.

Funktioniert, weil:

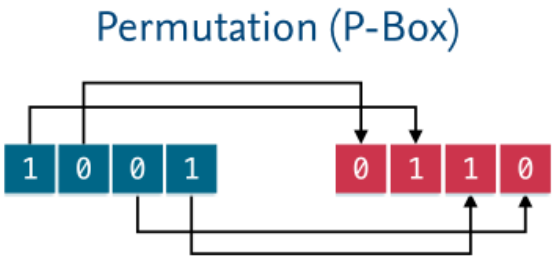
- XOR mit einem wirklich zufälligen Schlüssel wieder völlig zufällig ist.

- Wenn C = xor(M, K), dann ist jedes M gleich wahrscheinlich.

9. Gängige Chiffren
   (a) Grundlagen
   (b) Modern Ciphers
      i. S-Box
         A. Vorderseite Was ist eine *S-Box* im Kontext moderner, kryptographischer Ciphers? (VL 2, S.21)
         B. Rückseite Substitutionsbox. Ersetzt den Dateninhalt eines Blocks durch einen anderen.



      ii. P-Box
         A. Vorderseite Was ist eine *P-Box* im Kontext moderner, kryptographischer Ciphers? (VL 2, S.21)
         B. Rückseite Permutationsbox. Ersetzt den Dateninhalt eines Blocks durch einen anderen.



(c) Block Chiphers

i. Vorderseite Was ist eine *Block Cipher*? Nennen Sie drei gängige. Zu welchem Zweck werden sie präferiert eingesetzt?

ii. Rückseite

- *Block Ciphers* ver- bzw. entschlüsseln Datenblöcken fester Größe.
- AES, Serpent, Twofish
- Verschlüsselnung statisher Daten wie Dateien.

Nachrichten, die nicht in einen Block passen, müssen in Blöcke gestückelt und evtl. auf Blockgröße aufgefüllt werden (Achtung: nicht mit 0en).

(d) Strom Chiffren

i. Vorderseite Was ist eine *Stream Cipher*? Nennen Sie zwei gängige. Zu welchem Zweck werden sie präferiert eingesetzt?

ii. Rückseite

- *Stream Ciphers* arbeiten auf Bit bzw. Byte Ebene.
- Rabbit, Salsa20
- Verschlüsselung dynamischer Daten.

Ein *Cipher Stream* wird erzeugt indem mithilfe eines PRNG (*pseudo random number generator*) parallel zum *Message Stream* ein *Key Stream* erzeugt wird, die on-the-fly per XOR zusammengefügt werden.

10. Block Cipher Modes

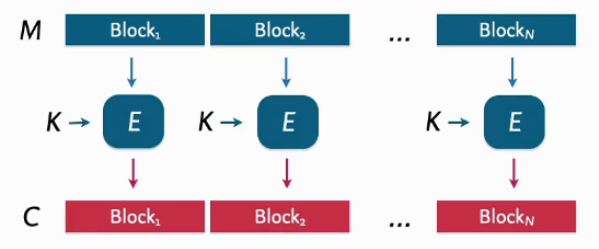
(a) Electronic Code Book (ECB)

i. Definition ECB

A. Vorderseite Was ist die *Electronic Code Book* Mode bei der Verschlüsselung mit Block Chiffren? Zeichnen Sie ein Diagramm.

B. Rückseite

- Naive Verschlüsselung der Blöcke: Man nimmt den Block und verschüsselt ihn ohne weiteres.



ii. Vor- und Nachteil bei ECB

A. Vorderseite Nennen sie den Vorteil der *ECB Mode* bei der Verschlüsselung von Blöcken. Was ist der Nachteil?

B. Rückseite

- Es ist einfach und effizient: Die Verschlüsselung der eines Blöcks passiert unabhängig von der eines anderen und daher ist es einfach sie zu implementieren und verteilt (auf mehreren Cores) auszuführen.
- Anfällig für known-plaintext und replay Attacks: Ein Block wird immer auf den gleichen verschlüsselten Block abgebildet. Überträgt man zwei mal die gleichen Daten, weiß man es sind die gleichen. Ein Angreifer kann so auch einen bekannten Block in die Übertragung injecten und so die Kommunikation stören.

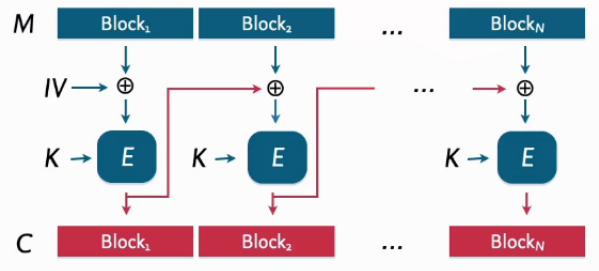
(b) Cipher-Block Chaining

i. Definition

A. Vorderseite Was ist die *Cipher-Block Chaining* Mode bei der Verschlüsselung mit Block Chiffren? Zeichnen Sie ein Diagramm.

B. Rückseite

- Vor der Verschlüsselung eines Blocks wird er mit bereits verschlüsselten Version des vorherigen Block geXORt. Der erste Block wird mit einem *Initialisierungsvektor IV* geXORt. Der IV ist eine zufälliger String und wird für jede Nachricht neu generiert und offen mitgesendet.



ii. Vor- und Nachteile bei CBC

A. Vorderseite Nennen sie die Vorteile der *CBC Mode* bei der Verschlüsselung von Blöcken. Wäre ein Nachteil?

B. Rückseite

- Resistent gegen known-plaintext und replay Angriffe.
- Verschlüsselung erfolgt sequenziell und kann nicht mehr parallelisiert werden.

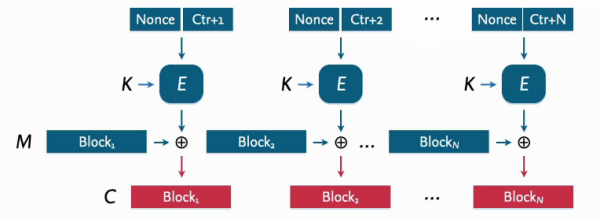
(c) Counter Mode (CTR)

i. Definition

A. Vorderseite Was ist die *Counter Mode* bei der Verschlüsselung mit Block Chiffren? Zeichnen Sie ein Diagramm.

B. Rückseite

- Die Verschlüsselung von Blöcken wird wie eine Stream Cipher gehandhabt. Man kann das Verfahren in zwei Teile teilen. 1. Erzeugung des *key streams*: Der *key stream* wird durch Verschlüsselung E einer offenen *nonce* (number used only once) zusammen mit einem *Counter* mithilfe des Schlüssel K erzeugt.
- Verschlüsselung der Blöcke: XOR Operation eines Key Blocks und des zu verschlüsselnden Blocks.



ii. Vor und- Nachteile bei CTR

A. Vorderseite Nennen sie die Vorteile der *CBC Mode* bei der Verschlüsselung von Blöcken.

B. Rückseite

- Die Keyblöcke können vorberechnet werden.
- Die Keyblöcke können *neu* berechnet werden. Dazu braucht man nur die offene Nonce, den offenen Counter und den geheimen Schlüssel. Sollte ein Block verloren gehen, kann er ohne Probleme neu berechnet werden.

1.1.3 3 Vorlesung (Public Key Cryptography)

1. Probleme der symmetrischen Verschlüsselung

   (a) Vorderseite Nennen Sie zwei Probleme der symmetrischen Verschlüsselung.

   (b) Rückseite

   - Symmetrische Kryptographie setzt ein *shared secret* (den Schlüssel) voraus. Der Austausch verschlüsselter Nachrichten ist sicher, aber der Schlüsselaustausch ist es nicht.
   - Gruppenkommunikation: Damit eine Gruppe unter einander sicher kommunizieren kann, braucht es ein *shared secret* pro Paar. Für n Gruppenmitglieder braucht man $\frac{n^2-n}{2}$ Schlüssel.

2. Idee der asymmetrischen Verschlüsselung

   (a) Vorderseite Was ist bei der *public key* Verschlüsselung *asymmetrisch*? Was erleichtert es? Hat es Nachteile?

   (b) Rückseite

   - Trennung des Schlüssels in einen zur Verschlüsselung und einen zur Entschlüsselung. Der Verschlüsselungsschlüssel kann öffentlich gemacht werden.
   - Weil der öffentliche Schlüssel öffentlich ist, ist das Problem des Schlüsselaustausches gelöst. Für Gruppenkommunikation braucht es nur ein Schlüsselpaar pro Person.
   - ...

3. Rivest-Shamir-Adleman (RSA)

   (a) Erfinder

   i. Vorderseite Nennen Sie die drei Erfinder von RSA

   ii. Rückseite

- Ron Rivest
- Adi Shamir
- Leonard Adleman

(b) Euler'sche Phi-Funktion

i. Vorderseite Wie ist die euler'sche Phi-Funktion definiert?

ii. Rückseite

- Für jede positive, natürliche Zahl n gibt sie die Anzahl der zu n teilerfremden natürlichen Zahlen kleiner n an.
- $\phi(n) = |\{a \in (N) | 1 \leq a \leq n \wedge ggT(a, n) = 1\}|$
- $\phi(n \cdot m) = \phi(n) \cdot \phi(m)$
- Für eine Primzahl p gilt: $\phi(p) = p - 1$

(c) Schlüsselerzeugung

i. Vorderseite Nennen Sie die nötigen Schritte zum Erzeugen eines RSA Schlüsselpaares. Gehen Sie darauf ein, warum der Schritt wichtig ist.

ii. Rückseite

A. Wähle zufällige **Primzahlen** p und q und berechne $n = p \cdot q$. Zur Sicherheit sollten p und q groß sein, weil die Sicherheit aus der Integer Faktorisierung kommt (schweres Problem).

B. Berechne Euler Funktion $\phi(n) = \phi(p) \cdot \phi(q) = (p - 1)(q - 1)$. Leicht, wenn p und q bekannt sind.

C. Wähle *öffentlichen* Schlüssel e mit $ggT(e, \phi(n)) = 1$. Nur so gibt es zu e ein Inverses (mod $\phi(n)$)

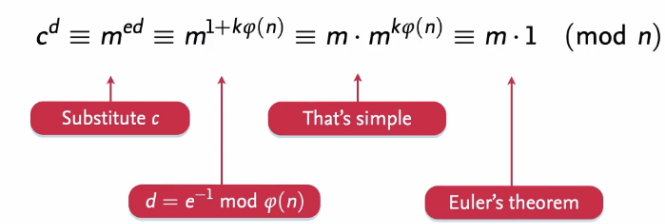
D. Berechne den *privaten* Schlüssel $d = e^{-1}$ mod $\phi(n)$.

(d) RSA Verschlüsselung

i. Vorderseite Sei ein RSA Schlüsselpaar e, d und eine Nachricht m gegeben. Wie erhält man den Ciphertext c? Wie entschlüsselt man c zurück zu m?

ii. Rückseite

- $c = m^e \mod n$
- $m = c^d \mod n$



(e) Angriff

4. Diffie-Hellman

(a) Erfinder

i. Vorderseite Wer hat das Diffe-Hellman Verfahren entickelt?

ii. Rückseite

- Die Idee kam von *Ralph Merkle*
- Entwickelt von *Whitfield Diffie* und *Marin Hellman.*

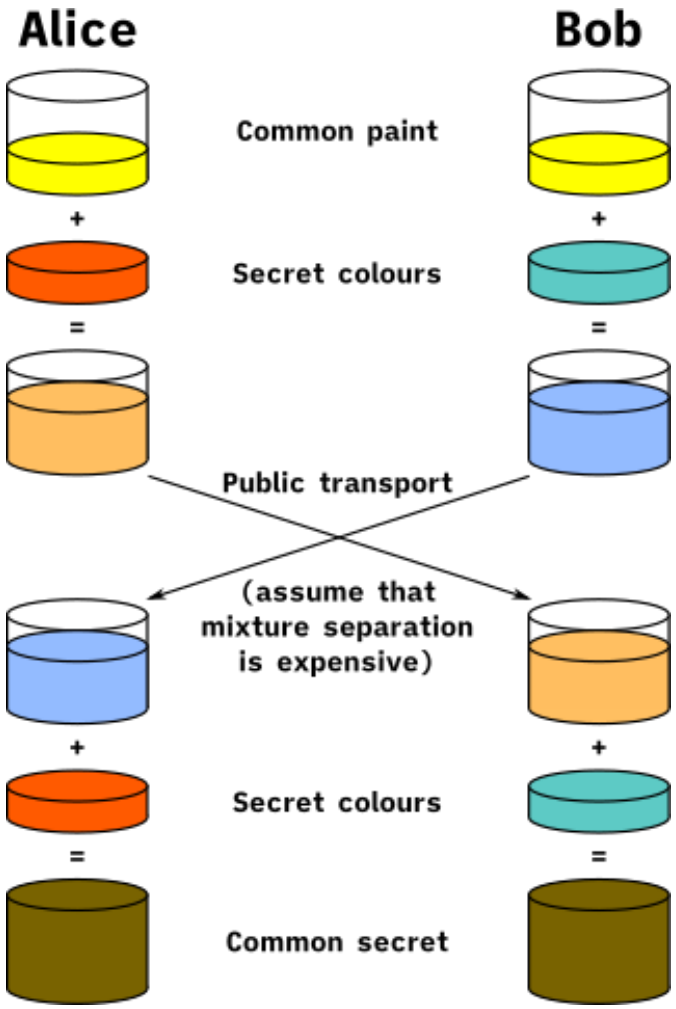
(b) Prinzip

i. Vorderseite Was ist der Zweck und das Prinzip von *Diffe-Hellman*?

ii. Rückseite

- Schlüsselaustausch: Etablierung eines *shared secret* über einen offenen Kanal. Kein Nachrichtenaustausch.

- Beide Partner haben je eine geheime Zahl a bzw. b, die sie nicht direkt austauschen, sondern *vermischt* mit einer öffentlichen Zahl (das Entmischen ist schwer). Die Mischung wird ausgetauscht und die jeweils eingene geheime Zahl zur Mischung hinzugefügt. Beide Mischungen sollten daraufhin gleich sein.



(c) Funktionsweise

i. Vorderseite Wie funktioniert der Diffe-Hellman Schlüsselaustausch?

ii. Rückseite

A. Alice und Bob einigen sich öffentlich auf eine **Primzahl** n (Modulus) und einen **Generator** (Erzeuger) g aus dem Modulus n, so dass also $\langle g \rangle = \mathbb{Z}_n$.

B. Alice und Bob wählen je eine geheime, zufällige Zahl a bzw. b.

C. Sie vermischen jeweils ihre geheime Zahl mit $\mathbb{Z}_n$ und erzeugen A bzw. B: $A = g^a \bmod n$ bzw. $B = g^b \bmod n$.

D. Sie tauschen A und B aus.

E. Beide mischen ihre geheimen Zahlen hinzu und erhalten den gleichen Schlüssel k: $k = B^a \bmod n$ und $k = A^b \bmod n$.

Beweis:

$$k \equiv X^y \equiv g^{xy} \equiv g^{x \cdot y} \equiv g^{yx} \equiv Y^x \pmod{n}$$

Substitute X

Reverse direction

Logarithm rule

(d) Angriff

i. Ableitung des Schlüssels

A. Vorderseite Wie kann ein Dritter bei *Diffe-Hellman* der Schlüssel k berechnen?

B. Rückseite $k = A \cdot B \bmod n = g^x \cdot g^y \bmod n = g^{x \cdot y} \bmod n$. Jedoch muss dafür jedes mögliche $x * y$ aus $\mathbb{Z}_n$ ausprobiert werden.

ii. Entschlüsselung

A. Vorderseite Wie kann ein Dritter bei *Diffe-Hellman* die Verschlüsselung rückgängig machen?

B. Rückseite $a = log_g(A) \bmod n$. Allerdings ist das sehr schwer.

1.1.4 4 Vorlesung (Hybrid Cryptosystem)

1. Cryptographic Hash Functions Alernative Namen: message digests, fingerprints. Im Prinzip wie eine Hash-Datenstruktur, aber mit dem Zusatz, dass ie *cryptographisch* ist.

   (a) Prinzip

      i. Vorderseite Was ist das Prinzip einer *crypographischen Hash-funktion*? Was sind ihre Eigenschaften?

      ii. Rückseite

         A. Sie bildet Nachrichten einer beliebigen Menge auf (binäre) Zahlen einer festen Länge ab. Diese Zahl wird der *Fingerabdruck* genannt. Um effektiv zu sein, muss die Bildmenge sehr groß sein, sodass eine Kollision sehr unwahrscheinlich ist.

         B. Sei M die Nachricht, H die Hashfunktion, D der Fingerabdruck. Eigenschaften:

            - One-way property: Ist M gegeben, kann D = H(M) leicht bestimmt werden. Ist D gegeben, ist eine Rekonstruktion von M sehr schwer.
            - Schnell berechenbar
            - Die Änderung eines Bits in der Nachricht ändert den Fingerabdruck signifikant (um mehr als 50%).

   (b) Kollisionsresistenz

      i. Kollision

A. Vorderseite Was ist bei kryptographischen Hashfunktionen eine *Kollision*?

B. Rückseite Zwei verschiedene Nachrichten M haben den selben Fingerabdruck.

ii. Birthday Attack

A. Vorderseite Was ist eine *Birthday Attack* auf eine kryptographische Hashfunktion?

B. Rückseite Eine Sicherheitsüberprüfung der Integrität von Nachrichten mithilfe von Digests (und sogar MAC) kann umgangen werden, wenn man zu einem gegebenen Digest eine beliebige andere Nachricht finden kann.

iii. Resistenz

A. Vorderseite Was ist *schwache* Kollisionsresistenz? Was ist *starke*?

B. Rückseite

- Für eine *gegebene Nachricht* M ist es schwer eine Nachricht M' zu finden, die den gleichen Hash hat. Eine *Birthday* Attacke ist möglich.
- Für *alle möglichen Paare* von Nachrichten M und M' ist eine Kollision schwer zu finden. Eine *Birthday* Attacke ist nicht möglich.

2. Digital Signatures

(a) Problem der unauthentifizierten asymmetrischen Verschlüsselung

i. Vorderseite Welcher Gefahr setzt man sich aus, wenn man asymmetrisch verschlüsselt aber ohne Authentifizierung kommuniziert?

ii. Rückseite

- Man-in-the-middle Attacks (MITM): Wenn Kommunikaitonspartner nicht gegenseitig ihre Identität verifizieren, kann es sein dass ich eine Drittperson in die Mitte schaltet und für beide Seiten den Schlüsselaustausch simuliert und auf die Art die Kommunikation mitlesen kann. Gleiches gilt für den Austausch von öffentlichen Schlüsseln.

(b) Henne-Ei-Problem

i. Vorderseite Was ist das *Henne-Ei-Problem* beim Austausch von asymmetrischen Schlüsseln.

ii. Rückseite Für die Verifizierung der Identität des asymmetrischen Schlüssels braucht man einen Hash des Schlüssels, der auch erst unauthentifiziert übermittelt werden muss.

(c) Prinzip

i. Vorderseite Was ist das Prinzip bei *digitale Signaturen*?

ii. Rückseite Die Funktion der Schlüssel bei der asymmetrischen Verschlüsselung wird umgedreht: Ein Hash der Nachricht wird mit dem *privaten* Schlüssel verschlüsselt und kann mit dem *öffentlichen* Schlüssel entschlüsselt werden. So kann jeder, der den öffentlichen Schlüssel hat, sehen dass der Besitzer des privaten Schlüssels die Nachricht gesendet hat.

(d) in RSA

i. Reguläres Signieren

A. Vorderseite Wie wird eine digitale Signatur in RSA *klassischerweise* bewerkstelligt? Gibt es dabei Probleme? (VL 3)

B. Rückseite

- Der Vorgang ist ähnlich dem Verschlüsseln per RSA, nur wird dazu der private Exponent genommen.

- Weil nur der Besitzer des privaten Schlüssels signieren können soll.
- Die Signatur kann mit dem öffentlichen Schlüssel überprüft werden.
- Probleme:
  - Nachrichten müssen kleiner sein als der Modulus
  - Signieren kann viel Zeit in Anspruch nehmen, da ein großer Datensatz mit einem großen Exponenten verrechnet werden muss
  - Die Nachrichten und Signaturen von 0, 1, n-1 sind stets gleich
  - *Blinding Attack*

ii. Blinding Attack

A. Vorderseite Wie funktioniert die *Blinding Attack* auf RSA Signaturen?

B. Rückseite Sei M eine Nachricht, die nicht signiert werden würde, aber R^eM schon (MR^e geht auch). Signiert der Besitzer des *privaten* Schlüssels R^eM, kann auf folgende Art eine Signatur für M erzeugt werden:

- $(R^eM)^d \mod n = R^{e \cdot d}M^d \mod n = RM^d \mod n$
- Rausrechnen: $S/R \mod n = \frac{RM^d}{R} \mod n = M^d \mod n$

iii. Signatur des Hashes

A. Vorderseite Wie wird eine digitale Signatur in RSA mithilfe kryptographischer Hashfunktionen bewerkstelligt?

B. Rückseite Sei M eine Nachricht, H die Hashfunktion, d ein *privater* Schlüssel, e ein *öffentlicher* Schlüssel und s die Signatur.

- Unteschrift erstellen: Signieren mit d: $s = H(M)^d \bmod n$
- Unterschrift mit e verifizeiren: $H(M) = s^e \bmod n$

(e) Zertifizierungsstellen

i. Vorderseite Was ist die Funktion einer *Zertifizierungsstelle* in asymmetrischer Verschlüsselung?

ii. Rückseite Sie dient als *trusted third party*, die reale Identitäten und öffentliche Schlüssel in Verbindung bringt, also die Identität des Inhabers verifiziert. Dann setzt sie ihre digitale Signatur auf den Schlüssel. Findet man also die digitale Signatur einer Zertifizierungsstelle, der man vertraut, auf einem öffentlichen Schlüssel, kann man sich sicher sein, dass der Schlüssel der Person ist.

(f) Angriffe

i. Key-only Attack

A. Vorderseite Was ist ein *key-only attack* gegen Signaturen?

B. Rückseite Man kennt nur den öffentlichen Schlüssel und versucht damit eine Signatur zu produzieren.

ii. Known-message

A. Vorderseite Was ist ein *known-message attack* gegen Signaturen?

B. Rückseite Man kennt zu dem öffentlichen Schlüssel zusätzlich eine signierte Nachricht. Damit kennt man den Hash und aus welcher Nachricht er berechnet wurde. Jetzt kann versucht werden, die Nachricht zu verändern, ohne dass der Hash sich ändert.

iii. Chosen-message

A. Vorderseite Was ist ein *chosen-message attack* gegen Signaturen?

B. Rückseite Man kann Nachrichten nicht selbst signieren, kann aber das Opfer dazu bringen, Nachrichten für einen selbst zu signieren.

(g) Forgery

i. Vorderseite Was sind die 3 Stufen der *signature forgery*?

ii. Rückseite

A. Existential Forgery: Es existiert auf irgendeiner Nachricht eine gefälschte Signatur. Blöd, aber kein Problem.

B. Selective Forgery: Für gewissen Nachrichten können Signaturen gefälscht werden.

C. Universal Forgery: Es kann alles signiert werden.

3. Hybrid Cryptosystems

(a) Warum?

i. Vorderseite Warum sind *hybride Kryptosysteme* nötig?

ii. Rückseite

- Mit bessere Methoden der Technik und Algorithmik ermöglichen immer schnellere Berechnungen kryptographischer Schlüssel.
- Um dem entgegenzuwirken, werden Schlüssel immer länger, aber damit auch langsamer.
- Symmetrische Kryptographie ist dagegen widerstandsfähiger. In hybriden Kryptosystemen nimmt man die besten Teile beider Welten.

(b) Prinzip

i. Vorderseite Was ist das Prinzip bei *hybriden Kryptosystemen*? Was ist eine Schwäche

ii. Rückseite

- Asymmetrische Verschlüsselung für die Etablierung des symmetrischen Sitzungsschlüssels.
- Symmetrische Block- bzw. Stromcipher für den effizienten Austausch von Daten.
- Das ganze System ist nur so stark wie das schwächste Glied der Kette.

(c) Forward Secrecy

i. Definition

A. Vorderseite Was ist *Forward Secrecy* in Kryptosystemen? Wie wird es implementiert?

B. Rückseite

- Erlangt ein Angreifer (in Zukunft) den Key, kann er damit den gesamten Nachrichtenverlauf entschlüsseln. ryptosysteme mit *Forward Secrecy* kalkulieren das mit ein und sorgen dafür, dass der Schaden bei einem Sicherheitsvorfall begrenzt bleibt.
- Für symmetrische Verfahren ist es leicht: Jede Sitzung sollte einen neuen Schlüssel benutzen.
- Bei asymmetrischen Verfahren ist es schwieriger: Sie sind groß und an Identitäten gekoppelt. Sie oft zu ändern ist inpraktikabel. Werden sie aber nur zur Etablierung der Sitzung verwendet (RSA signiertes Diffe-Hellman), ist *Forward Secrecy* garantiert.

ii. Diffe-Hellman + RSA

A. Vorderseite Wie funktioniert *Forward Secrecy* mit Diffie-Hellman und RSA? (VL 4, S.23)

B. Rückseite

- Der Session-Key wird für jede Session neu per Diffie-Hellman Key Exchange generiert ($\rightarrow$ jedes mal eine neuer Schlüssel)
- Der Diffie-Hellman Austausch wird mit RSA signiert, um Man-In-The-Middle-Attacks zu vermeiden.

(d) Public Key Infrastructure (PKI)

i. Arten

A. Vorderseite Welche zwei Arten von *Public Key Infrastructure* gibt es?

B. Rückseite

- Chain of trust: X.509 Standard, hierarchisch
- Web of trust: Pretty Good Privacy (PGP), dezentral

(e) Chain of Trust PKI

i. Wie funktioniert es?

A. Vorderseite Wie funktioniert eine *hierarchische* Public Key Infrastructure?

B. Rückseite

- Certificate Authorities (CA): Signieren mit einem wohlgehüteten privaten Schlüssel ein ausgestelltes Zertifikat (öffentlicher Schlüssel + Attribute).
- CAs sind *trusted third parties*.

- Mit dem ausgestellten Zertifikat können weitere öffentliche Schlüssel / Zertifikate signiert werden.

(f) Web of Trust PKI

i. Wie funktioniert es?

A. Vorderseite Wie funktioniert eine *dezentrale* Public Key Infrastructure?

B. Rückseite

- Es gibt keine zentrale Authorität.
- Jeder erstellt seinen Schüssel und lässt ihn von seinen Bekannten signieren und signiert wiederum ihre (über einen sicheren Kanal).
- Trifft man auf einen *unbekannten* Public Key, kann geguckt werden, ob er bekannte Signaturen enthält, die das Vertrauen in den Key erhöhen können.

4. SSL

(a) Was ist das?

i. Vorderseite Wofür steht SSL? Wofür steht TLS? Was ist der Zusammenhang?

ii. Rückseite

- Secure Sockets Layer
- Transport Layer Security
- Werden oft als Synonym verwendet. SSL war die erste Fassung eines Transportwegverschlüsselungsstandards. Nachdem einige Schwachstellen gefunden wurden, enschied man sich für ein auf SSL aufbauenden Standard names TLS, der jedoch nicht mit SSL rückwärtskompatibel ist.

(b) Zweck

i. Vorderseite Was ist der Zweck von TLS?

ii. Rückseite

- Implementiert eine Peer-to-Peer Verschlüsselung: Auf dem Weg einer Nachricht in einem Computer-Netzwerk sollen die Zwischensysteme die Nachricht nicht lesen können.
- Andere Prozesse, die auf dem Rechner laufen und keine Root-Rechte haben, sollen die Nachricht auch nicht lesen können.

(c) Auf welcher ISO OSI Schicht liegt TLS?

i. Vorderseite Auf welcher ISO OSI Schicht liegt TLS?

ii. Rückseite

- Zwischen Transportschicht (4) und Anwendungsschicht (5 bzw. 6).
- Bis einschließlich TCP (o.ä.) ist ist alles unverschlüsselt. Der Inhalt der TCP Segmente wird von der Anwendung für die Peer-Anwendung verschlüsselt.

(d) Wie ist TLS implementiert?

i. Vorderseite Wie ist TLS implementiert?

ii. Rückseite

- Hybrides Kryptosystem: Mindestens die Serverseite hat ein von einer CA signiertes X.509 Zertifikat, welches die Authentizität bestätigt.
- Mithilfe des Zertifikats (Signatur) wird asymmetrisch (Diffie-Hellman) ein symmetrischer Schlüssel für die Sitzung erzeugt (TLS Handshake).

- Mithilfe des Schlüssels werden Anwendungsdaten verschlüsselt ausgetauscht.

(e) Woher kommt das Vertrauen in TLS? Wie kann es angegriffen werden?

i. Vorderseite Woher kommt das Vertrauen in TLS? Wie kann es angegriffen werden?

ii. Rückseite

- Chain of Trust, dessen Ursprung eine *Certificate Authority* ist.
- Wird bei der CA eingebrochen und der private Schlüssel erlangt, können damit beliebig viele Serverzertifikate für beliebige Webseiten erstellt werden. So können dann MITM Attacken initiiert werden.

(f) Verteidigung gegen Chain-of-trust Angriffe

i. Vorderseite Was wären *Gegenmaßnahmen* für Angriffe gegen die Chain of Trust bei TLS? Haben sie Nachteile?

ii. Rückseite

- *Certificate Pinning*: Ein Client bringt das Zertifikat (oder den Fingerprint) des Servers direkt mit (quasi einkompiliert). So kann ihm kein gefälschtes Zertifikat untergeschoben werden, aber das Zertifikat kann auch nicht ausgetauscht werden ohne dass der Client angepasst werden muss. Weiterhin ist es ein Rückschritt zur Chain of Trust.
- *Certificate Transparency*: Ein öffentliches *append-only Log* in Form eines *Merkle-Hash-Trees* wird für ausgestellte Zertifikate angelegt. Jeder Eintrag im Baum führt den Hash seines Parents für den Eintrag mit. Wird oben im Baum etwas geändert, passen alle nachfolgenden Hashes nicht mehr. Auch neu ausgestellte, gefälschte Zertifikate müssen in das *append-only* Log eingefügt werden und sind so für immer drin. So können Manipulationen schnell erkannt werden. Die Überprüfung der Logs ist jedoch aufwändig.

5. PGP

   (a) Was ist?

      i. Vorderseite Wofür steht PGP? Was ist das? Was ist GPG?

      ii. Rückseite

         - Pretty Good Privacy
         - Hybrides Cryptosystem mit Web-of-trust Modell
         - GNU Privacy Guard: Eine freie PGP Implementierung.

1.1.5 5 Vorlesung (Authentication)

1. Auth

   (a) Was ist Auth?

      i. Vorderseite Was ist Authentifizierung?

      ii. Rückseite Die Verbindung einer *Identität* (Person oder anderer Computer) mit einem *Subjekt* (Nutzeraccount).

   (b) Faktoren der Authentifizierung

      i. Vorderseite Nennen Sie die drei *Faktoren* der Authentifizerung und geben sie je ein Beispiel.

      ii. Rückseite Die Bestätigung einer Identität erfolgt mithilfe von:

         - Knowledge Factors: Wissen, dass nur der Computer und die Person hat und niemand sonst. Z.B.: Passwort oder Pin-Code auf einer Geldkarte.
         - Ownership Factors: Unterscheidung von Personen anhand von Besitz: Z.B. Schlüssel, EC Karte, Hardware Token
         - Human Factors: Auch biometrische Faktoren genannt. Alles was den menschlichen Körper einzigartig macht. Iris-Muster, Fingerabdrücke, Stimme.

Authentifizierung kann über eine beliebige Kombination dieser Faktoren erfolgen.

(c) Location Faktor

i. Vorderseite Warum ist der *Location Factor* kein guter Faktor der Authentifizierung?

ii. Rückseite

- Authentifizierung erfolgt überwiegend über das Internet von überall auf der Welt. Authentifizierung über der Ort macht das ganze sehr unpraktisch.
- Der Ort selbst muss mithilfe weiterer Faktoren geschützt werden, wie einem Passwort oder Schlüssel. Daher ist der eigentliche Schutz der weitere Faktor.

(d) Auth Fakoren in der Betrachtung

i. Auth by Knowledge

A. Vorderseite Was sind die *Vor- und Nachteile* einer Authentifizerung über *Knowledge Factors*?

B. Rückseite

- Vorteile:
  - Simple Einrichtung ohne spezielle Hardware
  - Im Fall einer Offenlegung ist die Erstzung von Geheimnissen leicht.
- Nachteile:
  - Das menschliche Gedächtnis ist nicht immer verlässlich, dadurch tendieren Menschen dazu, einfache Passwörter zu wählen oder ein Passwort mehrfach zu verwenden.

- Die Geheimhaltung ist nicht immer garantiert (Post-it am Bildschirm oder *purchase key attacks*).

ii. Auth by Ownership

A. Vorderseite Was sind die *Vor- und Nachteile* einer Authentifizerung über *Ownership Factors*?

B. Rückseite

- Vorteile:
  - Handhabung ähnlich zu physikalischen Schlüsseln (Yubikey).
  - Replikation und Cloning sind i.d.R. schwer
- Nachteile:
  - Verlust oder Diebstahl des Hardware Tokens ist problematisch.
  - Brauch spezielle Hardware

iii. Auth by Biometry

A. Vorderseite Was sind die *Vor- und Nachteile* einer Authentifizerung über *Human Factors*?

B. Rückseite

- Vorteile:
  - Ideale Verbindung zwischen Subjekten und Identitäten.
  - Biometrische Eingenschaften sind durch die Natur allen gegeben und gehen nur schwer verloren.
- Nachteile:

- Die Ersetzung dieser Eigenschaften ist schwierig
- Sobald sie genutzt werden, müssen sie geschützt werden. Stimmen kann man aufnehmen, Fingerabrücke von einer Oberfläche entnehmen, eine Iris von einem hochauflösenden Foto entnehmen.
- Man kann sich nicht unter einem Pseudonym authentifizieren, weil die Kopplung nicht an *eine* Identität sondern an den Körper gebunden ist.

iv. Multi-Factor Auth

A. Vorderseite Was ist *Multi-Factor* Authentifizierung? Nennen sie drei Beispiele.

B. Rückseite

- Die Authentifizierung über mehrere Authentifizierungsfaktoren. Es erhöht die Sicherheit.
- Bezahlung per EC-Karte: Ownership Factor (Die Karte) + Knowledge Factor (die PIN).
- Bezahlung per Kreditkarte: Ownership Factor (Scan der Karte) + Unterschrift (Human Factor).
- Klausurenanmeldung: Knowledge Factor (GITZ Passwort) + Ownership Factor (TAN)

(e) Auth mit Passwörtern und deren Probleme

i. PW und Asym

A. Vorderseite Wie werden Passwörter in der *asymmetrischen* Kryptographie verwendet?

B. Rückseite Der private Schlüssel wird mithilfe eines *symmetrischen Schlüssels* verschlüsselt, dessen Grundlage ein Passwort sein kann.

ii. PW und Sym

A. Vorderseite Wie werden Passwörter in der *symmetrischen* Kryptographie verwendet?

B. Rückseite Aus einem Passwort wird per *Hash* oder *Key Derivation Function* ein 256 bit Schlüssel generiert. Das wird gemacht, damit jeder mögliche Schlüssel des Keyspaces gleich wahrscheinlich wird.

iii. Probleme

A. Vorderseite Nennen Sie drei Arten von Problemen von Passwörtern

B. Rückseite

- *Password snooping*:
  - Das Passwort kann mitgelesen werden (unverschlüsselte Netzwerkkommunikation).
  - Das Passwort kann per Malware vom Computer extrahiert werden (z.B. Keylogger).
- *Password guessing*:
  - *Dictionary Attacks*: Das Raten von Passwörter mithilfe von Wortlisten.
  - *Brute-force Attacks*: Alles durchprobieren.
- *Human deficiencies*: Menschen können das selbe schwache Passwort öfter verwenden.

iv. PW Storage

A. Cleartext

B. Vorderseite Wie sollen Passwörter auf keinen Fall gespeichert sein? Warum nicht?

C. Rückseite

- Im Klartext
- Gelingt es nicht das System zu sichern, sind alle Passwörter direkt offengelegt:
  - Ein Fehler in der Absicherung der User-Datenbank legt alle Passwörter offen
  - Gibt es ein hardcoded Passwort in einem Programm, kann es per Debugger herausgeholt werden.

D. Hashed

E. Vorderseite Wie solle die Speicherung von Passwörtern realisiert werden. Was hat das für Vorteile?

F. Rückseite Hashed and salted: Zum Hash das Passwort zu finden ist sehr schwer, das Passwort den Hash sehr leicht. *Salt*: Ein offener, zufälliger String der an das Passwort gehängt wird.

G. Salt

H. Vorderseite Was ist das *Salt* bei gehashten Passwörtern? Was sind die Vorteile?

I. Rückseite

- Ein zufälliger, offen gespeicherter String für jedes Passwort. Es wird beim Erstellen das Hashen an das Passwort gehängt
- Hat man eine Login-Datenbank, muss man ein Passwort nur einmal Hashen und kann *in jedem* Account danach suchen. Hat aber jeder Account einen Salt, muss das Passort immer wieder neu gehasht werden. Man kann auch eine Liste von Hashes von beliebten Passwörtern vorbereiten (Pre-Computation of Hashes) und nach den Passwort-Hashes in den Account Daten suchen. Ein Salt verhindert das.

J. Key stretching

K. Vorderseite Was ist *key stretching* bei der Absicherung von Passwörtern?

L. Rückseite Ein Passwort wird nicht nur einmal, sondern mehrere (tausend) mal gehasht um die Berechnung etwas teurer zu machen. Führt dazu, dass ein Login länger braucht, aber das ist für Alice akzeptabel. Aber Mallory will ein Passwort raten und muss für jeden Versuch mehrere Male hashen.

2. Challenge Response

   (a) One Time Password

      i. Vorderseite Was sind die Vor- und Nachteile von One Time Passwords?

      ii. Rückseite

         - Vorteile:
            - *Password cracking* wird nutzlos
            - *Password snooping* wird nutzlos
            - Dem Altern von Passwörtern wird entgegengewirkt
         - Nachteile:
            - Es braucht ein System von Passwörtern

   (b) S/KEY Algorithmus

      i. Vorderseite Was ist das Prinzip hinter dem S/KEY *one-time password* Algorithmus?

      ii. Rückseite

         A. Der Nutzer wählt ein initialen Key K_1

B. Es werden K_n Passwörtern per *rekursivem* Hashing aus K_1 abgeleitet: $H(K_1) = K_2, H(K_2) = K_3, ..., H(K_{n-1}) = K_n$

C. Sie bilden die One-time Passwords: $P_1 = K_n, P_2 = K_{n-1}, ..., P_n = K_1$

D. Es ist schwer ein Passwort P_i aus P_{i-1} abzuleiten, weil Hashes rückwärts berechnet werden müssen.

E. Oft bekommter der Nutzer nicht den K_1, sondern eine Liste von P_1 bis P_n.

F. Diese Liste muss periodisch regeneriert werden

G. Sie muss sicher aufbewahrt werden.

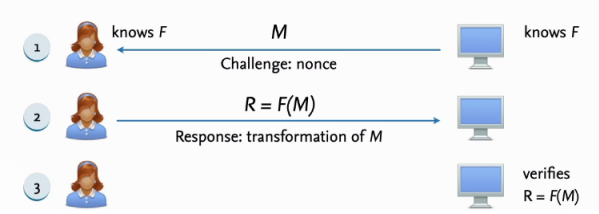
(c) Generisches Challange-Response System

i. Prinzip

A. Vorderseite Wie funktionieren *Challange-Response* Authentifizierungsalgorithmen prinzipiell? Zeichnen Sie ein Diagramm. (VL 5, S. 22).

B. Rückseite

- Das Geheimnis zwischen Nutzer und System ist eine geheime *Funktion F*.
- *Challange* der Kenntnis von F: Das System sendet eine *Nonce* an den Nutzer. Der Nutzer berechnet die *Response* R=F(M) und überträgt das an das System.
- Das System überprüft seinerseits R = F(M)



ii. Wahl der Challange

   A. Vorderseite Nennen Sie drei Dinge, die für die Wahl von M in *Challange-Response* Authentifizierung geeignet sind. (VL 5, S. 23)

   B. Rückseite

      - Uhrzeit
      - Zufällige Zahl
      - Counter

iii. Wahl der geheimen Funktion

   A. Vorderseite Nennen Sie zwei Beispiele für geheime Funktionen in *Challange-Response* Authentifizierung. (VL 5, S. 23)

   B. Rückseite

      - F = (M + P) wobei H eine Hash-Funktion ist und P ein geheimes Passwort.
      - F = $E_P(M)$ wobei E eine offene Verschlüsselungsfunktion ist und P ein geheimes Passwort

      Ist F kryptographisch start, kann das P nur sehr schwer abgeleitet werden.

iv. Beispiele

   A. Vorderseite Nennen Sie zwei Beispiele für *Challange-Response Authentication* Schemes. Was ist dabei die *Challange* und was die *Response*? (VL 5, S. 23)

   B. Rückseite

      - One-time passwords: Die *Challange* ist der Index eines Passworts auf einer Liste, die *Response* ist das einmalige Passwort.

- RSA SecurID: *Challenge*: aktuelle Zeit, *Response*: Authentifizierungscode.

(d) Access Control Was darf welcher Nutzer nachdem die Authentifizierung geschehen ist?

i. Was ist Access Control

A. Vorderseite Was ist *Access Control*? (VL 5, S. 26)

B. Rückseite Nachdem ein *Authentifizerungsschritt* die Identität eines Subjekts festgestellt hat, bestimmt *Access Control* in wie weit sie Systemdienste und -ressourcen in Anspruch nehmen dürfen. *Access Control* besteht aus:

- Einschränkung der Berechtigung eines Subjekts. Was darf das Subjekt machen?
- Verwaltung von Zugangsberechtigungen: Wie und wo wird das festgehalten?
- Enge Kopplung an Authentifizierung: Bevor eingeschränkt werden kann, muss die Identität festgestellt werden.

ii. Schwierigkeit

A. Vorderseite Woher rührt die Schwierigkeit der bei der Verwaltung von *Access Control*? (VL 5, S. 26/27)

B. Rückseite [Enthält eigene Mutmaßungen] Komplexe Systeme haben komplexe Access Control Modelle:

- Die *Identität* eines *Subjekts* ordnet ihm eine *Rolle* zu. Anhand der Rolle wird Zugriff eingeschränkt. Sollte eine Identität mehrere Rollen inne haben, wird das Konzept aufgeweicht.
- Es ist nicht immer absehbar, zu was sich grundlegende Fähigkeiten kombinieren lassen. Manchmal erwachsen ungeahnt große Fähigkeiten aus der Kombination kleiner Fähigkeiten.

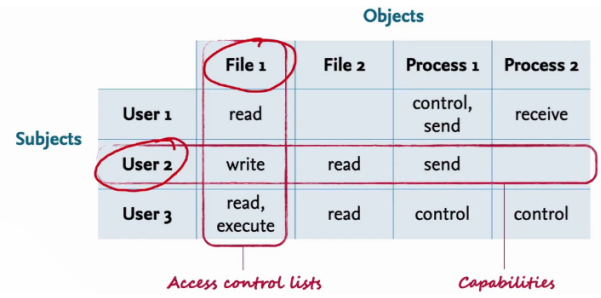
- Einige Objekte können auch als Akteure auftreten (z.B. im Fall von *Prozessen*), die wiederum eigene Fähigkeiten haben. Subjekte die solche Prozesse steuern können, bekommen implizit auch die Berechtigungen des Prozesses (bzw. den Teil, über den sie die Kontrolle haben).

iii. Access Control Matrix

A. Vorderseite Was ist eine *Access Control Matrix*? Zeichnen Sie ein Beispiel. Was repräsentieren die Zeilen der Matrix und was die Spalten bezogen auf den Benutzer? (VL 5, S. 27)

B. Rückseite

- Ein Mapping einer Kombination von Subjekt (Benutzer bzw. Akteure) und Objekt (Systemressource oder -dienst) auf *Aktionen*.
- Spalten repräsentieren *Access Control Lists*: Für jedes Objekt wird eine Liste von erlaubten Aktionen jedes Benutzers festgehalten.
- Zeilen repräsentieren *Capabilites*: Für jedes Subjekt wird eine Liste von erlaubten Aktionen bezüglich jedes Objekts festgehalten.



iv. Charakteristiken

A. Vorderseite Nennen Sie drei Charakteristiken von Access Control Modellen. (VL 5, S. 28)

B. Rückseite

- Definitionen von Objekten und Subjekten. Subjekte können Nutzer, Prozesse und Hosts sein.
- Repräsentation von Berechtigungen: *Access Control Lists*, *Capabilities*.
- Verwaltung von Berechtigungen: *discretionary*, *mandatory*, *role-based*.

v. ACLs

A. Vorderseite Nennen Sie die Vor- und Nachteile von *Access Control Lists*. Nennen Sie ein Beispiel für ein ACL basiertes System (VL 5, S. 29).

B. Rückseite Rechte werden an Objekte gebunden.

- Vorteile:
  - Gut in dezentralen Systemen: Sind die Objekte des Systems verteilt, ist es ein Vorteil die Zugriffskontrolle ebenfalls zu verteilen, denn ansonsten wäre das ein zentraler Mechanismus in einem dezentralen System.
- Nachteile:
  - Schwer zu ermitteln, welche Rechte ein Nutzer konkret hat, denn es müssen alle Objekte angeschaut werden.
- OpenBSD packet filter: Eine Firewall Implementierung bei der ein Netzwerkhost ein Subjekt ist und die Netzwerkpakete das Objekt. Die Firewall ist eine Liste an Regeln, welche (ein- und ausgehenden) Pakete passieren dürfen und welche verworfen werden.

vi. Capabilities

A. Vorderseite Nennen Sie die Vor- und Nachteile von *Capabilites* basierter Zugangsbeschränkung. Nennen Sie ein Beispiel für ein solches System (VL 5, S. 30).

B. Rückseite Rechte werden an Subjekte gebunden.

- Vorteile:
  - Das Auflisten der Berechtigungen eines Subjekts ist leicht, weil sie sich an einem Ort befinden.
- Nachteile:
  - Feingranuläre Zugangskontrolle ist für Dateien schwer, denn es braucht für jeden Nutzer eine Übersicht aller Dateien mit den Zugangsberechtigungen. Zugang zu Funktionen zu verwalten ist hingegen leichter, wenn es nicht so viele davon gibt.
  - In dezentralen Systemen schwer verwaltbar
- Linux Capabilities: Die Fähigkeiten eines Nutzers werden aufgewertet, indem ihm einige Möglichkeiten, die eingetlich nur dem Super User zur Verfügung stehen, gegeben werden. Konkret sind es z.B. die Fähigkeit das System neuzustarten oder die Fähigkeit ein Kernelmodul zu laden.

vii. Management of permissions

A. Discretionary Access Control (DAC)

B. Vorderseite Was ist *Discretionary Access Control*? Wo wird das häufig eingesetzt? (VL 5, S.31)

C. Rückseite

- Der Besitzer eines Objekts kontrolliert die Zugansberechtigungen. Es ist praktisch, aber schwierig, wenn das Objekt den Besitzer wechseln soll, weil der neue Besitzer die Rechte erst prüfen muss.
- Unix Dateisysteme

D. Mandatory Access Control (MAC)

E. Vorderseite Was ist *Mandatory Access Control*? In welchem Kontext wird das oft verwendet?

F. Rückseite

- Zugangskontrolle wird von einer übergeordneten Instanz verwaltet. Das beschneidet die Möglichkeiten des Nutzers die Zugangsberechtigung zu seinen eigenen Objekten zu ändebrn.
- High security environments wie Geheimdiensten. Das System wird sehr sicher aber unflexibel.

G. Role-based Access Control (RBAC)

H. Vorderseite Was ist *Role-based Access Control*?

I. Rückseite

- Rechte werden mit Rollen verknüpft. Ein Nutzer hat stets die Rechte aller Gruppen, denen er zugehörig ist.

1.1.6 6 Vorlesung (Netzwerkangriffe)

1. Layer Communication Model

   (a) Negiative Auswirkung der Vernetzung von Computern auf ihre Sicherheit

      i. Vorderseite Wie hat die Vernetzung von IT-Systemen die IT Sicherheit negativ beeinflusst? (VL 6, S. 4)

ii. Rückseite

- Physical access → Network Access: Location based auth factor fällt weg.
- Dozens of users → Thousands of hosts: Unterscheidung zwischen User und Host verschwimmt. Mehrere Geräte können einem Nutzer zugeordnet sein und sich parallel in mehrere verteile Dienste eingloggen.
- Central resources → Distributed Resources: Mails müssen z.B. nicht mehr auf dem selben Rechner liegen wo der Mailserver läuft. Jedes einzelne System und die Schnittstellen haben eigene Angriffsflächen
- Easy accountability → Hard Accountability: Erschwert die Suche nach Schuldigen

Die Systeme wurden in den 60er konzipiert, die Protokolle in den 70ern. Beides wurde nur nach und nach an die aktuellen Gegebenheiten angepasst.

(b) ISO OSI Modell

i. Vorderseite Nennen sie die Layer des ISO OSI Modells (VL 6, S. 5)

ii. Rückseite 7 - Application 6 - Presentation 5 - Session 4 - Transport 3 - Network 2 - Data Link 1 - Pysical

(c) TCP/IP Modell

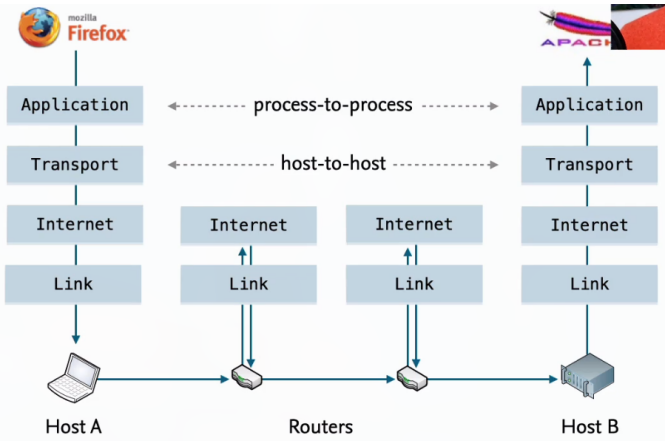
i. Vorderseite Nennen sie die Layer des TCP/IP Modells, gehen Sie kurz auf seine jeweilige Funktion ein und nennen sie je zwei Beispielprotokolle (VL 6, S. 6).

ii. Rückseite Application: Schnittstelle von vernetzten Anwendungen. HTTP, FTP Transport: Kommunikation verschiedener Anwendungen hinter den IP Adressen: TCP, UDP Internet: Adressierung und Transfer von Daten zwischen Hosts und Gateways: IP, ICMP Link: Die physische Verbindung von Geräten und die Beschaffenheit der Kommunikation. PPP, ARP

(d) TCP/IP Data Flow

i. Vorderseite Zeichnen Sie das TCP/IP Data Flow Diagramm (VL 6, S. 7)

ii. Rückseite



2. Classic Network Attacks

(a) Ursachen von Schwachstellen

i. Vorderseite Sind die vier Angriffsfläche von Netzwerkangriffen? Nennen Sie Beispiele (VL 6, S.12)

ii. Rückseite

- Schwächen im Netzwerkprotokolldesign
  - Beim Entwurf von TCP/IP hat niemand geahnt, dass es ein weltumspannendes Protokoll wird.
- Schwachstellen in den Protokollimplementierungen
  - Windows, Linux, *BSD usw. haben alle eine eigene Implementierung des TCP/IP Stacks, die sich unterscheiden und zu Problemen führen können.

- Spezifikationen sind manchmal mehrdeutig

- Fehlerhafte Konfiguration der Netzwerkdienste
  - Webserver setzt den Root Folder auf seine Konfigurationsdatei

- Fehlerhafter Betrieb der Netzwerkdienste
  - Kommunikation startet mit HTTPs hat aber Links auf HTTP Basis, die Verschlüsselung nicht nutzt.

(b) Drei Klassiker

i. Vorderseite Nennen und erklären Sie drei verschiedene klassische Netzwerkangriffe (VL 6, S.13)

ii. Rückseite

- Spoofing: Verfälschen von Daten wie die der Identität (Absender oder Empfänger).
- Hijacking: Eine Verbindung oder Sitzung wird übernommen. Als MITM übernimmt man einfach die Kommunikation komplett.
- Flooding: Fluten von Diensten mit einer enormen Menge von Paketen um seine Verfügbarkeit anzugreifen.

(c) Sniffing

i. Vorderseite Was ist Network Sniffing? (VL 6, S.14)

ii. Rückseite

- Abhören von Netzwerkpaketen.
- Prinzipiell keine Attacke, denn Admins machen das auch.
- Benötigt wird der physische Zugang zum Übertragungsmedium.

- Angriff ist passiv, verletzt die Vertraulichkeit.

(d) ARP Spoofing

i. Vorderseite Was ist ARP Spoofing? Was ist ARP (in einem Satz)? (VL 6, S.16)

ii. Rückseite

- Das *Address Resolution Protocol* ist ein mappt logische Adressen (IP) auf Geräteadressen (MAC)
- Dem Switch (Link Layer) wird per ARP Reply mitgeteilt (manchmal kontinuierlich → ARP Flooding), dass sich eine bestimmte IP, dessen Pakete man abhören möchte, sich unter seiner eigenen MAC befindet. So leitet man IP Pakete zu sich um.
- Basis für MITM Attacks: Greift die Vertraulichkeit und Integrität an. Leitet man die Pakete nicht weiter, dann auch die Verfügbarkeit

(e) Smurf Attack

i. Vorderseite Was ist ein *Smurf Attack*? Was setzt ihn voraus? (VL 6, S.18)

ii. Rückseite

- Im Prinzip ein *Amplified Flooding* Angriff, bei dem man sich zu nutze macht, dass eine kleine Datenmenge, dessen Absenderadresse gespooft wurde und die man an einen geeigneten bzw. misskonfigurierten Netzwerkdienst rausschickt, der dem gespooften Absender daraufhin mit einer sehr großen Datenmenge antwortet.
- Ein Biespiel funktioniert per Ping an eine IP-Subnetz-Broadcastadresse mit gespoofter Quell-IP-Adresse.
- Reduziert die Verfügbarkeit der Netzwerkbandbreite.

3. Network Defenses

(a) Präventive Sicherheit Wie verhindere ich Angriffe? :ANKI$_{\text{NOTETYPE}}$: Einfach

   i. Vorderseite Was sind die drei Konzepte der *präventiven Sicherheit* im Netzwerk? (VL 6, S.23)

   ii. Rückseite

   - Kryptographie: Verschlüsselung und Verfikiation der Daten
   - Authentifikation: (gegenseitige) Bestätigung der Indentität
   - Zugriffskontrolle: Beschränkung und Kontrolle der möglichen Kommunikation im Netzwerk

(b) Reaktive Sicherheit Was mache ich, wenn Angriffe passiert sind? :ANKI$_{\text{NOTETYPE}}$: Einfach

   i. Vorderseite Was sind die drei Konzepte der *reaktiven Sicherheit* im Netzwerk? (VL 6, S.23)

   ii. Rückseite

   - Vulnerability assessment: Die Suche nach Schwachstellen
   - Intrusion detection: Die Suche nach laufenden Angriffen
   - Computerforensik: Analyse bereits geschehener Angriffe, die Suche nach Angreifern

(c) IPSec

   i. Vorderseite Was ist *IPSec*? Nennen Sie drei Haupteigenschaften. (VL 6, S.25)

   ii. Rückseite Erweiterung des IP Protokolls um Sicherheitsfeatures zum Schutz der Kommunikation im Internet Layer:

   - IKE: *Internet Key Exchange* (Diffe-Hellman)

- ESP: *Encapsulating Security Payload*: Verschlüssenlung des IP Payloads
- AH: *Authentication Header*: Implementierung eines Challenge-Response Mechanismus.

(d) Access Control in Networks Welches Paket darf an welchen Host im Netzwerk gehen? → *Firewall*, genauer *packet filter*.

i. Umsetzung

A. Vorderseite Wie wird Access Control in Netwerken umgesetzt? (VL 6, S.26)

B. Rückseite

- Definition von Netzwerk-Objekten auf allen Ebenen der TCP/IP Modells: Subnetze, Hosts, Ports
- Filterung von Paketen auf allen Ebenen:
  - Link Layer: MAC Filter
  - Internet + Transport Layer: Packet Filter
  - Application Layer: Proxy and application-layer gateway

ii. Firewalls

A. Vorderseite Wie ist eine *Firewall* definiert? Was sind seine Aufgaben? (VL 6, S.27)

B. Rückseite Ein Host, der Zugang zu einem Netzwerk vermittelt. Zu seinem Aufgaben gehört:

- Inspektion von ein- und ausgehenden Paketen
- Zugangskontrolle auf verschienden Schichten des Netwerks
- Semantische Analyse von Protokollen: z.B. Unterscheidung verschiedener TCP Pakte, Zustänader einer TCP Verbindung

- Aufteilung des Netzwerks in Subnetze

iii. DMZ

A. Vorderseite Was ist ein Vorteil der Aufteilung eines Netzwerks in ein lokales Netz und eine *demilitarized zone*? (VL 6, S.27)

B. Rückseite Alles, was von außen erreichbar sein muss, kommt in die DMZ, und der Rest in das lokale Netz. Es können verschiedene Regeln für die Netze gelten. Beispielweise kann die HTTP(s) Ports in der DMZ für die Außenwelt öffnen, aber für das lokale Netz verbieten. Ist ein Server im DMZ kompromittiert, so ist der Zugang ins lokale Netz nicht garantiert.

iv. Desktop Firewalls

A. Vorderseite Was ist eine Desktop-Firewall? Was ist ein Vor- und ein Nachteil davon? (VL 6, S.29)

B. Rückseite

- Ein im Desktop-Betriebssystem eingebauter Paketfilter. Seine Fähigkeit als Firewall ist beschränkt, weil die Regeln geändert werden können, sobald der Rechner infiziert ist. Andererseits kann er feingranular dazu genutzt werden, besondere Regeln für den Desktop-Rechner einzuführen. Z.B. kann verhindert werden, dass ein bestimmtes Programm mit dem Internet kommunizert.

1.1.7 7 Vorlesung (Web Security)

1. Server-side attacks

(a) code injection

i. Vorderseite Was ist der Ursprung jeder *code injection*? Wie verteidigt man sich dagegen? (VL 7, S.18+)

ii. Rückseite

- Code und Daten werden nicht richtig getrennt. Es werden Strings vom Benutzer entegengenommen und wie Code ausgeführt.
- Verteidigung:
  - *string sanitization*: Man überprüft den entgegengenommenen String auf reservierte Symbole in der jeweiligen Programmiersprachensyntax, besonders Symbole, die Strings schließen oder Befehle terminieren.
  - Aus dem String die Parameter für eine vorbereitete Funktion entnehmen und sie ausführen. Z.B. *prepared sql statements*.

2. Web sessions

   (a) Was sind Web Sessions?

      i. Vorderseite Was ist eine *Web Session* und wofür braucht man die? (VL 7, S.22)

      ii. Rückseite

      - Das Web funktioniert über HTTP.
      - Diese Protokoll ist im Prinzip zustandslos, d.h. jedes Neuladen der Website ist wie ein komplett neuer Aufruf.
      - Viele Webanwendungen müssen aber einen Überblick darüber haben, was der Nutzer bisher getan hat, wie z.B. sich eingeloggt.
      - Daher muss eine Web Applikation ihre eigene Sitzungsverwaltung über der HTTP Ebene implementieren.

   (b) Sessionmanagement

      i. Vorderseite Nennen Sie drei Möglichkeiten, *Web Sitzungen* zu verwalten. Erklären Sie diese. Welche Probleme können auftreten? (VL 7, S.23)

ii. Rückseite

- *URL Rewriting*: Session IDs werden an die in der URL gehängt. Kann die Session-ID versehentlich an eine andere Website verraten, wenn per Link auf sie gewechselt wird (HTTP Referrer Header).
- *Form-based Sessions IDs*: Die Session ID wird in den Body der HTTP Nachricht verlagert. Dort ist sie Teil eines versteckten Formularfelds. Die HTTP-Anfrage muss jetzt per POST Methode passieren (sonst gibt es keinen Body). Verlust der Session bei Betätigung des Rückbuttons des Browsers, weil nach der vorherigen URL ohne POST Request gefragt wird.
- *Cookies*: Eine Erweiterung von HTTP. Header zum Setzen von Strings, die sich der Webbrowser für den nächsten Anruf merkt und mitsendet. Kann zum Tracken von Nutzern verwendet werden.

(c) Session IDs

i. Vorderseite Nennen Sie drei Dinge, die für sichere Sitzungen und Sitzungs-IDs gelten müssen. (VL 7, S.26)

ii. Rückseite

- Sie sollten schwer zu erraten sein (Lang und zufällig)
- Sie sollten nur verschlüsselt übertragen werden
- Sie sollten nach einer Zeit ablaufen

(d) Schutz von Cookies

i. Vorderseite Wie werden Cookies vor *client-side Angriffen* geschützt? (VL 7, S.)

ii. Rückseite

(e) Gleichheit nach Same-origin Policy

i. Vorderseite Wann haben zwei Ressourcen nach der *same-origin policy* den selben Ursprung? (VL 7, S.30)

ii. Rückseite

- Das Protokoll ihrer URI ist gleich
- Der Host in der URI ist gleich
- Der Port in der URI ist gleich

Pfade, Query Strings, Fragmente, Login-Daten können abweichen.

(f) Wofür ist die same-origin Policy?

i. Vorderseite Wer schützt wen wovor mithilfe der *same-origin* Policy? (VL 7, S.30)

ii. Rückseite

- Browser und Anbieter eines Webdienstes schützen ihre Nutzer vor bösartigen Websites, die JavaScript beinhalten.
- Durch die Fähigkeiten von JavaScript wäre es ohne Same-origin Policy möglich, auf Authentifizierungscookies dritter Webdienste zuzugreifen und HTTP Requests jeder Art an sie zu senden.
- Durch die Same-origin Policy beschränken sich diese Fähigkeiten und der einhergehende Zugriff auf Ressourcen der eigenen Seite.

3. Client-side attacks

(a) Cross-site Scripting Name

i. Vorderseite Wie kommt es zu dem Namen *Cross-site Scripting*? (VL 7, S.31)

ii. Rückseite Es wird versucht, fremden JavaScript Code in die aufgerufene Website des Opfers einzuschleusen und damit im gleichen Origin zu landen.

(b) Cross-site Scripting

i. Vorderseite Was ermöglicht *Cross-site Scripting*? (VL 7, S.34)

ii. Rückseite Das Anzeigen von ungenügend validierten Daten. Das kann über folgende Dinge geschehen:

- Seiteninhalt wird dynamisch aus URI Parametern erstellt → Script in den Link
- Seiteninhalt wird unverschlüsselt per JavaScript extern nachgeladen → per MITM JavaScript in die Daten injizieren
- Mediendateien wie SVGs können JavaScript beinhalten.

(c) *Reflected* Cross-site Scripting

i. Vorderseite Was ist *reflected* Cross-site Scripting? (VL 7, S.37)

ii. Rückseite XSS wird per Link an das Opfer transportiert. Damit ist der Angriff kurzlebig. Der Link muss nicht immer offensichtlich sein, sondern sich unter dem HTML einer Email verbergen, oder per URL-Shortener verändert werden.

(d) *Stored* Cross-site Scripting

i. Vorderseite Wie funktiniert *stored* Cross-site Scripting? (VL 7, S.38)

ii. Rückseite

- JavaScript wird über eine Datenbank in die Website eingeschleust.
- Werden bei der gewöhnlichen Erstellung einer Ressource die Eingabedaten nicht validiert, kann man dort JavaScript injizieren, welches dann in die Datenbank kommt.

- Ein Opfer braucht nur dem Link zu dieser Ressource zu folgen, damit bei der dynamischen Erstellung der Seite das JavaScript aus der Datenbank geladen wird.
- Dieser Angriff ist persistent, und daher heißt es *stored*.

(e) Verteidigung gegen Cross-site Scripting

i. Vorderseite Wie kann man sich gegen *Cross-site Scripting* verteidigen? (VL 7, S.41)

ii. Rückseite

- Escaping und Bereinigung von GET und POST Parametern
- Validierung von extern (nicht im Websitecode) gespeicherten Daten (in der Datenbank)
- Prüfen anderer Quellen, wie Cookies oder Referrer

(f) Schwierigeiten bei der Input-Validierung

i. Vorderseite Warum ist die Validierung von Eingabedaten zum Schutz gegen *Cross-site Scripting* schwer? (VL 7, S.41)

ii. Rückseite

- Oft ist es schwierig zu beurteilen, ob Eingabedaten XSS zum Ziel haben (z.B. weil Nutzer sich in den Kommentaren über HTML unterhalten, oder gar am HTML auf ihrer Page basteln dürfen).
- In dem Fall ist es leichter, die Daten bei der Ausgabe zu escapen. (Whitelisting ist besser als Blacklisting).

1.1.8 8 Vorlesung (Vulnerabilites and Exploits)

1. Basics of software security

(a) Definition Vulnerability

i. Vorderseite Wie ist *Vulnerability* definiert? (VL 8, S.3)

ii. Rückseite

- Ausnutzbare Schwachstelle im System oder Software
- Muss eine Basis für eine Attacke gegen die Verfügbarkeit, Integrität oder Vertraulichkeit bilden
- Nicht jeder Bug führt auch zu einer Verwundbarkeit

(b) Quellen von Vulerabilities

i. Vorderseite Nennen Sie die vier häufigesten *Quellen* von Vulnerabilites. Nennen Sie je ein Beispiel (VL 8, S.3)

ii. Rückseite

- Fehler im Design:
  - z.B. in Library Code: Inkonsistene Abstraktionen und Schnittstellen
- Fehler in der Implementierung:
  - Vermischen von Code und Daten
- Fehlkonfiguration:
  - Schwache Verschlüsselung wird gewählt
- Fehlerhafter Betrieb:
  - Bei dezentraler Architektur: unsichere Abspeicherung auf einem anderen System

(c) Schwierigkeiten bei der Entwicklung sicherer Software

i. Vorderseite Warum ist die Entwicklung sicherer Software extra schwierig? (VL 8, S.4)

ii. Rückseite Sicherheit muss in das Design, die Entwicklung und den Betrieb integriert sein. Das erfordert mehr Expertise und Zeit seitens der Entwickler und erhöht so die Kosten während der Entwicklung.

(d) Definition Exploit

i. Vorderseite Definieren Sie *Exploit*? (VL 8, S.5)

ii. Rückseite Ein *Exploit* ist ein Programm, welches eine Schwachstelle in einem System oder Software ausnutzt, um Schutzziele der der IT-Sicherheit zu gefährden (oder zumindest um die Gefährdung zu demonstrieren).

(e) Phasen eines Exploits

i. Vorderseite Nennen Sie drei typische *Phasen* eines Exploits. (VL 8, S.5)

ii. Rückseite

A. Injektion von Code als Daten

B. Manipulation von Kontrollfluss (Injizierter Code wird ausgeführt bzw. aus per ROP bösartige Funktionalität zusammenbasteln)

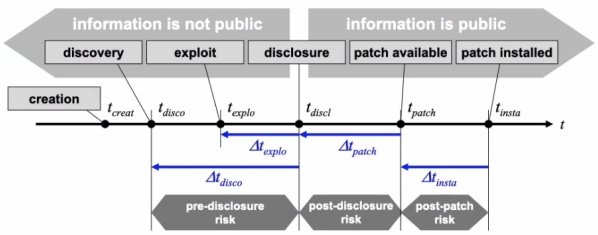
C. Erhöhung der Privilegien

(f) Lebenszyklus einer Vulnerability

i. Diagramm

A. Vorderseite Zeichnen Sie ein Diagramm zum *Lebenszyklus einer Vulerabilitiy* (VL 8, S.6)

B. Rückseite



ii. *zero day*

A. Vorderseite Was ist ein *zero-day* Exploit? (VL 8, S.6)

B. Rückseite Ein Exploit für eine gefundene *Vulnerability*, die noch nicht veröffentlicht wurde.

iii. *pivotal point*

A. Vorderseite Was ist der *pivotal point* im Lebenszyklus einer Vulnerability? (VL 8, S.6)

B. Rückseite Die Bekanntmachung deiner Vulnerabilitiy. Dieser Zeitpunkt markiert eine risikoreichen Abschnitt im Lebenszyklus einer *Vulnerabilitiy*, da ein Patch noch nicht in allgemeinem Umlauf ist.

iv. Vulerability Types

A. Vorderseite Nennen Sie drei *Klassen* von Vulnerabilities und geben Sie Beispiele (VL 8, S.7)

B. Rückseite

- Memory:
  - Stack / Heap / Integer Overflows
- Concurrency:
  - Race conditions
  - Deadlocks
  - Livelocks
- Validierung:
  - Cross-site Scripting
  - SQL Injection
  - Remove file inclusion

2. Stack and heap overflows

   (a) Buffer Overflow

      i. Vorderseite Was ist ein *Buffer Overflow*? Wie funktioniert ein *Buffer Overflow Exploit*? (VL 8, S.27)

      ii. Rückseite

         - Ein Buffer Overflow passiert, wenn über die Grenzen eines vorher angelegten Datenpuffers hinausgeschrieben wird.
         - Das kann für einen Exploit genutzt werden, wenn die Daten, die über die Grenze hinausgeschrieben werden, angrenzende Variablen überschreiben, die den Kontrollfluss ändern.

   (b) Gründe für Buffer Overflow

      i. Vorderseite Nennen Sie zwei *Ursachen* für Buffer Overflows. Geben Sie Beispiele. (VL 8, S.28)

      ii. Rückseite

         - Pufferspeicher ist zu klein:
            - Unbeschränkte Stringfunktionen (`gets`, `strcpy`, `sprintf`, ...) können Pufferspeicher überlaufen lassen
         - Falsche Berechnung eines Speicherorts oder Speichergröße:
            - Arithemtische Überläufe (Addition zweier Ganzzahlen führt zu einer Zahl, die größer ist, als der Datentyp), Pointerarthmetik
            - Es wird nicht beachtet, dass Type-Casting Speichergrößen ändert
            - Gefährliche Funktionen wie `strncpy`, `strncat` werden verwendet (NULL Terminator nicht garantiert).

(c) Exploitation of Stack Overflow

i. Vorderseite Nennen Sie zwei Arten, mithilfe derer Buffer Overflows für *Exploits* genutzt werden können.

ii. Rückseite

- Manipulation lokaler Variablen und vorheriger Stackframes: Änderung der Kontrollflusses
- Manipulation der return Adresse: Injektion von ausführbarem Code

(d) Shellcode

i. Vorderseite Definieren Sie den Begriff *Shellcode*. Gibt es davon Variationen? (VL 8, S.33)

ii. Rückseite Shellcode wird so genannt, weil es früher tatsächlich Code war, der auf dem System eine Shell startet. Heute wird damit injizierter (Maschinen-)Code bezeichnet, der ausführbar ist. Mit dessen Hilfe wird oft weiterer Code nachgeladen (*staged shellcode*). Passiert es über das Netzwerk, nennet man es eine *bindshell* (Shellcode macht einen Port auf).

(e) Heap Overflow

i. Vorderseite Wie kann ein *Heap Overflow* als Exploit genutzt werden? (VL 8, S.35)

ii. Rückseite

- Abhängig von der Implementierung können per Heap-Overflow Metadaten des Heaps (z.B. Pointer auf den nächsten Block) samt Speicherbereich überschrieben werden. So kann Speicherinhlat manipuliert werden, oder eine *Unlink Attack* implementiert werden.

(f) Unlink Attack

i. Vorderseite Wie funktioniert eine *Unlink Attack*? (VL 8, S.36)

ii. Rückseite

- Je nach Implementierung des Heap-Allokators kann eine *Unlink Attack* ermöglicht werden, wenn eine Heap-Buffer Overflow die Metadaten der Speicherblöcke samt ihrer Pointer überschreiben kann.
- Es wird möglich die Pointerzuweisung z.B. der `free()` Funktion zu missbrauchen, um beliebige Daten an eine beliebige Stelle im Speicher zu schreiben. Das wird *arbitrary memory write* genannt und ist ein mächtiges Werkzeug

3. Memory defenses

(a) Static Measures

i. Vorderseite Nennen Sie zwei *statische* Maßnahmen gegen Overflow Attacks. (VL 8, S.39)

ii. Rückseite

- Ersetzen unsicherer Funktionen (`strcat`, `strcpy`) gegen sichere Varianten.
- Prüfung von Puffergrenzen zur Kompilierzeig (*Boundary Checking*).

(b) Dynamic Measures

i. Vorderseite Nennen Sie vier *dynamische* Maßnahmen gegen Overflow Attacks. (VL 8, S.39)

ii. Rückseite

- Prüfung von Puffergrenzen zur Laufzeit
- Zufällige Anordnung des Adressraums (*address space layout randomization*).
- Zugriffskontrolle auf Speichersegmente
- *Shadowing of metadata*

(c) Kriterien für memory-safe functions

i. Vorderseite Nennen Sie drei Kriterien für *memory-safe functions*. (VL 8, S.40)

ii. Rückseite

- Explizite und implizite boundary checks: Kein Schreiben über Grenzen hinweg.
- Konsistente Interfaces von Funktionen, die in den Speicher schreiben: Reduziert die Wahrscheinlichkeit, dass ein Entwickler sich bei der Puffergröße verschätzt.
- Entfernen unsicherer Funktionen

(d) Shadow Stack

i. Vorderseite Was ist ein *Shadow Stack*? Wogegen ist er eine Hilfe und wie? Gibt es Nachteile? (VL 8, S.41)

ii. Rückseite

- Der Schutz von Metadaten eines Stack-Frames, im x86 Instruction Set realisiert.
- Es werden wichtige Elemente des Stack wie *Return Pointer* und *Base Pointer* in einen anderen Stack kopiert.
- Im Funktionsepilog werden sie verglichen. Wenn sie sich unterscheiden, wird das Programm beendet.

Nachteile:

- Der Speicherverbrauch steigt
- Laufzeit wird erhöht, weil ein Vergleich der Werte nötig ist (Intel realisiert das in Hardware, daher geht es schnell).

(e) Canaries

i. Vorderseite Was ist eine *Stack Canary*? Wogegen ist es ein Schutz und wie? (VL 8, S.42)

ii. Rückseite

- Eine *Stack Canary* ist eine lokale Variable, die ein zufälliges Bitmuster beinhaltet und direkt vor den Stackframfe-Metadaten (Return- und Base-Pointer) steht.
- Versucht ein Buffer-Overflow letztere zu überschreiben, überschreibt er auch den Stack Canary.
- Am Ende der Funktion wird geprüft, ob der Canary noch seinen ursprünglichen Wert hat. Wenn nicht, gelten die Stackfame-Metadaten als überschrieben.
- Viele Compiler bauen das automatisch ein, wenn mit *Stack Protection* kompiliert wird.

(f) Non-Executable Memory

i. Vorderseite Wie und wogegen hilft *non-executable memory*? (VL 8, S.43)

ii. Rückseite Es verhindert das Ausführen von injiziertem Code, der im Prozess durch einen Exploit außerhalb des ausführbaren Text-Segmentes geschrieben wurde. Dazu werden alle Segmente außer des Text- (auch Code-) Segments als schreibbar, aber nicht ausführbar markiert.

(g) ASLR

i. Bedeutung

A. Vorderseite Wofür steht *ASLR*? (VL 8, S. 44)

B. Rückseite Address Space Layout Randomization.

ii. Nutzen

A. Vorderseite Wie funktioniert *ASLR*? Wofür wird es verwendet? Gibt es Nachteile? (VL 8, S.44)

B. Rückseite

- Die einzelnen Speichersegmente des Prozesses werden zufällig permutiert, sodass nicht mehr vorausgesagt werden kann, wo welcher Speicherbereich liegt.
- Das erschwert *memory corruption* Angriffe
- Es kostet Laufzeit
- Verhindert keine zweistufigen Angriffe: Zuerst wird der Speicher ausgelesen und abgeschätzt wo welcher Speicherbereich sein könnte. Danach wird in den Speicherbereich gesprungen.

1.1.9 9 Vorlesung (Malware)

1. Typen von Schadsoftware

   (a) Aspekte von Schadsoftware

      i. Vorderseite Nennen und erklären Sie drei Aspekte von Schadsoftware. (VL 9, S.5)

      ii. Rückseite

      - Self-replication: Schadsoftware kopiert und verbreitet sich automatisch
      - Parsitismus: Schadsoftware injiziert schädlichen Code in anderen Code oder Daten.
      - Unerlaubte Kommunikation: Schadsoftware kommuniziert mit dem Angreifer

   (b) Backdoors

      i. Definition

         A. Vorderseite Definieren Sie den Begriff *backdoor*. Wo kann eine *Backdoor* platziert werden? (VL 9, S.6)

B. Rückseite

- Versteckter, bösartiger Code, der der Sicherheitsmechanismen zur Zugangsbeschränkung umgeht und unerlaubten Zugang zu Ressourcen verschafft.
- Kann platziert werden in:
  - Als Programm im System
  - als Code in einer Library oder einem Programm
  - im BIOS / der Firmware
  - in der Hardware

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat eine *Backdoor*? (VL 9, S.6)

B. Rückseite

- Replikation: ja bzw. möglich
- Parasitismus: nein (obwohl ich der Meingung bis dass doch)
- Kommunikaiton: ja

(c) Logic Bomb

i. Definition

A. Vorderseite Definieren Sie den Begriff *Logic Bomb*. (VL 9, S.8)

B. Rückseite Schadsoftware, die

- von einem voher bestimmten Ereignis ausgelöst wird (z.B. Zeitpunkt ist eingetreten),
- tief im System versteckt ist,

- der automatischen Sabotage des Ziels dient

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat eine *Logic Bomb*? (VL 9, S.8)

B. Rückseite

- Replikation: möglich
- Parasitismus: nein
- Kommunikation: nein

(d) Trojaner

i. Definition

A. Vorderseite Definieren Sie den Begriff *Trojaner*. (VL 9, S.9)

B. Rückseite Schadsoftware, die *gutartiges* und *attraktives* Verhalten vorgibt, um unwissentlich auf einem System installiert zu werden. Wird oft mit anderen Malware-Typen kombiniert.

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat ein *Trojaner*? (VL 9, S.9)

B. Rückseite

- Replikation: möglich
- Parasitismus: nein
- Kommunikation: je nach dem, welches Ziel der Trojaner hat ($\rightarrow$ Kombination mit anderen Malware-Typen)

(e) Virus

i. Definition

A. Vorderseite Definieren Sie den Begriff *Comptervirus*. (VL 9, S.11)

B. Rückseite

- Code, der sich in Programme und Dokumente injiziert
- Werden diese Dokumente geöffnet, wird die Malware ausgeführt und kann sich durch das Suchen anderer Wirte vermehren.
- Klassisch sind es Programme gewesen, aber heute können es auch Dokumente oder Websites sein.

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat ein *Computervirus*? (VL 9, S.11)

B. Rückseite

- Replikation: ja
- Parasitismus: ja
- Kommunikation: nein

(f) Wurm

i. Definition

A. Vorderseite Definieren Sie den Begriff *Computerwurm*. (VL 9, S.12)

B. Rückseite

- Schadsoftware, Verwandter des Computervirus
- Repliziert sich selbst, aber ist nicht parasitär.

- Verbreitet sich über Schwachstellen im Computernetworking
- Payload des Wurms kann beliebige Schadsoftware sein

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat eine *Computerwurm*? (VL 9, S.12)

B. Rückseite

- Replikation: ja
- Parasitismus: nein, sonst Virus
- Kommunikation: ja, denn er ist nicht parasitär

(g) Spyware

i. Definition

A. Vorderseite Definieren Sie den Begriff *Spyware*. (VL 9, S.14)

B. Rückseite

- Schadcode, der sensible Daten auf dem Zielsystem sammelt (z.B. per Keylogging oder Form Grabbing)
- Kann Aspekte eines Trojaners haben (→ hat einen attraktiven Nutzen)
- Großer Bereich: Von Adware bis zu schweren Verbrechen
- Wird manchmal aus *Stealer* genannt

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat *Spyware*? (VL 9, S.14)

B. Rückseite

- Replikation: nein
- Parasitismus: nein
- Kommunikation: ja, denn es dient der Spionage

(h) Botnet

i. Definition

A. Vorderseite Definieren Sie den Begriff *Botnet*. (VL 9, S.15)

B. Rückseite

- Schadsoftware, die ein Computersystem kontrolliert
- Das *Botnet* ist nicht nur die Software, sondern das Netzwerk infizierter Rechner ("Zombies")
- Kontrolliert den Rechner von außen
- Das Netzwerk wird für verschiedene Aspekte, wie DDoS oder Passwort-Cracking verwendet

ii. Aspekte

A. Vorderseite Welche *Malware-Aspekte* hat ein *Botnet*? (VL 9, S.15)

B. Rückseite

- Replikation: möglich
- Parasitismus: nein
- Kommunikation: ja, denn es dient der Fernsteuerung des infizierten Systems

(i) Ransomware

i. Vorderseite Definieren Sie den Begriff *Ransomware*. (VL 9, S.16)

ii. Rückseite

- Schadcode, der zur Erpressung dient, indem er Ressourcen ausschaltet oder Dateien auf dem Zielsystem verschlüsselt

(j) Scareware

i. Vorderseite Definieren Sie den Begriff *Scareware*. (VL 9, S.16)

ii. Rückseite

- Schadcode, der dem Nutzer Angst einjagt, um ihn zu einer Tat zu bewegen. Z.B. dem Kauf eines Produkts, wie eines Viruscanners

(k) Dropper

i. Vorderseite Definieren Sie den Begriff *Dropper* im Kontext von Schadsoftware. (VL 9, S.16)

ii. Rückseite

- Eine Form von Trojaner, der dazu dient, weitere Schadsoftware auf dem infizierten System zu installieren

2. Bösartige Mechanismen

(a) File infection

i. Orte

A. Vorderseite Nennen Sie drei Möglichkeiten bzw. Stellen in einer ausführbaren Datei, an die sich ein Virus platzieren kann. Gibt es je Ort Schwierigkeiten? (VL 9, S.19)

B. Rückseite

- **Prepended**: Virus platziert sich vor den ausführbaren Code. Funktioniert für interpretierten Code, ist aber schwierig für kompilierten Code, denn der hat oft feste Adressen, die sich verschieben würden.
- **Appended**: Funktioniert auch gut für kompilierten Code, allerdings ist es auffällig, dass nach dem *Entry Point* erst ein weiter Sprung stattfindet.
- **Fragmented**: Viruscode ist im infizierten Code verteilt. Besonders für kompilierten Code ist es sehr schwer festzustellen, dass die Datei infiziert ist. Ist aber auch technisch anspruchsvoller.

(b) Verbreitung von Würmern über das Netzwerk

i. klassisch

A. Vorderseite Wie verläuft die *klassische* Ausbreitung eines *Computerwurms*? Wieso ist es heute schwierig? Worauf wird ausgewichen? (VL 9, S.21)

B. Rückseite

- Klassische Verbreitung:
- Ein infiziertes Quellsystem scannt alle Hosts in Netzwerk nach Netzwerkverwundbarkeiten (z.B. Netzwerkapplikationen mit Schwachstellen → Microsoft IIS)
- Das Quellsystem infiziert einen gefundenen Rechner automatisch durch die gefundene Schwachstelle
- Das infizierte System wiederholt den Vorgang
- Schwierig weil viele Schutzmechanismen dazugekommen sind:
  - Subnetze sind geschützt und abgeschirmt

– Desktopfirewalls

- Alternativ wird auf Kommunikationsanwendungen wie Emails (per Anhang oder HTML) ausgewichen, denn sie erlauben das Überschreiten von Netzwerkgrenzen.

ii. Drive-by Downloads

A. Vorderseite Wie funkioniert ein *Drive-by Download* von Computerwürmern? (VL 9, S.23)

B. Rückseite

- Der Angriff sucht Schwachstellen im Browser per JavaScript (oder Flash, Silverlight, Java-Applet).
- Die Schwachstelle wird genutzt, um Schadsoftware auf den Rechner des Besuchers zu laden.
- Der Besucher muss auf die Seite gelockt oder eine bereits populäre Seite infiziert werden.

(c) Obfuscation (Verstecken von bösartigem Code), wird auch *Packing* oder fälschlicherweise *Encryption* genannt.

i. Polymorhpismus

A. Vorderseite Was ist *Polymorphismus* im Kontext von gepackter Malware? (VL 9, S.24)

B. Rückseite

- Damit nicht einfach nach Signaturen der gepackten Malware gesucht werden kann, wird der Einpackprozess für jede Datei leicht verändert (z.B. ein anderer Schlüssel).
- Der *Decoder* (Entpacker) wird mit in die Datei injiziert und kann zur Laufzeit die Malware und die Daten, die die Malware braucht, auspacken.

- Nicht nur Malware nutzt das verpacken von Code, sondern wird auch im Kontexten der Softwareentwicklung z.B. für Kopierschutz genutzt.

ii. Emulation

A. Vorderseite Wozu dient *Emulation* im Kontext von gepackter Software? Wie funktioniert sie? (VL 9, S.25)

B. Rückseite

- Emulation ist eine weitere Möglichkeit, Schadsoftware zu packen (sie zu verstecken).
- Anstatt dass der Schadcode versteckt wird, können die Instruktionen auch in als Bytecode kodiert werden. Ein mitgeschickter Emulator für dann diesen Bytecode aus.

iii. Maßnahmen gegen gepackten Code

A. Vorderseite Wie kann gepackter Schadcode in Dateien bzw. Code gefunden werden? (VL 9, S.24)

B. Rückseite Die Datei kann in einer geschützten Umgebung ausgeführt oder geöffnet werden. Währenddessen kann das Verhalten der Datei bzw. des öffnenden Programms beobachtet werden.

(d) Cloaking & Evasion

i. Vorderseite Was ist *Cloaking & Evasion* im Kontext von Schadsoftware? (VL 9, S.27)

ii. Rückseite Das Verstecken von Schadsoftware oder Daten für den Betrieb der Schadsoftware. Es soll die Analyse der Schadprogramms erschweren. Versteckt wird beispielsweise:

- Daten der Schadsoftware im Dateisystem (verschlüsselt)

- Verschlüsselte Kommunikation, z.B. verschlüsselt in URLs

(e) Command-and-control communication

i. Vorderseite Was ist *command-and-control communication* im Kontext von Botnets? (VL 9, S.29)

ii. Rückseite

- Hierarchische Anordung der Kommunikation innerhalb eines Botnetzes:
  - Der *Botmaster* steuert mithilfe seiner *master server* eine Menge von *proxy bots*, die wiederum ihr eigenes Netzwerk an *worker bots* haben, welche dann konkrete Angriffe ausführen
- Peer-to-peer Kommunikation auch möglich

3. Untergrundwirtschaft Wie verdient man Geld mit IT Kriminalität?

(a) Maschen

i. Vorderseite Nennen Sie x Möglichkeiten, mit einem Botnetz Geld zu verdienen. (VL 9, Teil 3)

ii. Rückseite

- Spam:
  - Wenn 1% der Spamnachrichten zu einer Transaktion führen und man täglich 300000 Spamnachrichten verschickt, führt das zu 3000 Transaktionen.
  - Die Transaktionen selbst können legitim sein.
  - Der Verkäufer versendet Spam nicht selbst, sondern engagiert den Besitzer eines Botnets
- Erpressung per DDoS:

- Viele Dienste (z.B. Onlineshops) müssen erreichbar sein, um Geld zu generieren
- Mit der Androhung von DDoS (→ Botnetz) kann Schutzgeld erpresst werden

- Lösegelderpressung:
  - Daten können auf dem infizierten Host verschlüsselt werden
  - Entschlüsselung nur gegen Bezahlung

(b) Outsourcing von Malware-Infektion und -Verteilung

i. Vorderseite Beschreiben Sie das *Pay-per Install* Geschäftsmodell im Kontext der Malware-Verbreitung (VL 9, S.36)

ii. Rückseite

- Es ist eine Form der Spezialisierung in der IT-Kriminalität. Die Verteilung von Schadsoftware und die Erstellung von Schadsoftware sind zwei verschiedene, komplexe Tätigkeiten
- Malware-Autoren spezialisieren sich auf die Erstellung von Malware und engagnieren einen Verteilungsdienst, der die Malware vertreibt
- Die Vertreiber rechnen pro Malware-Installation ab

1.1.10 10 Vorlesung (Intrusion detection systems)

Reaktive Sicherheit. Vorher ging es um die Verhinderung von Schaden, jetzt geht es um die Reaktion auf Angriffe.

1. Grundlegendes und Monitoring

(a) Definitionen

i. Konzepte

A. Vorderseite Was sind die drei Konzepte der reaktive Security?

B. Rückseite

- Vulerabilitiy Assessment
- Intrusion Detection
- Computer Forensics

ii. Angriff

A. Vorderseite Wie ist der Begirff *Angriff* in der IT-Sicherheit definiert? (VL. 10, S.3)

B. Rückseite Der Versuch ein Schutzziel zu verletzen.

iii. Intrusion

A. Vorderseite Wie ist der Begirff *Intrusion* in der IT-Sicherheit definiert? (VL. 10, S.3)

B. Rückseite Erfolgreicher Angriff, also eine erflogreiche Verletztung eines Schutzziels

(b) A generic detector

i. Vorderseite Aus welchen vier Komponenten besteht ein *IDS* (*Intrusion Detection System*)? (VL 10, S.5)

ii. Rückseite

A. **Monitoring** of data: Das Verhalten von Programmen, das Beobachten des Network-Traffics

B. **Analysis** of data: Parsen von Netwerkdaten, *Feature Extraction* aus Programmen

C. **Detection** of threats: Missbrauchsmuster- oder Anomalieerkennung

D. **Response** to threats: Alarmierung des Personals, Blockieren der Handlung

(c) Monitoring

i. Network-based

A. Vorderseite Wie funktioniert grundlegende Netzwerküberwachung? Welche Vorteile und Schwierigkeiten gibt es? (VL 10, S.7)

B. Rückseite

- Netzwerkdaten werden an einem Knotenpunkt abgefangen und geparst
- Daraufhin werden sowohl die Header- als auch der Payload der Pakete inspiziert
- Vorteil: Ein ganzes Netzwerksegment wird geschützt
- Nachteil: Verschlüsselung und die Masse der Daten beschränken die Effektivität

ii. Host-based

A. Vorderseite Wie funktioniert grundlegende Host-Überwachung? Welche Vorteile und Schwierigkeiten gibt es? (VL 10, S.8)

B. Rückseite

- Jede Datei wird auf Schadcode-Signaturen überprüft (Batch oder On-access)
- Programmverhalten wird beobachtet: Welche Systemcalls führt das Programm aus? Redet es mit dem Netzwerk? Öffnet es Dateien, die nichts mit dem Programm zu tun haben? Schaut es sich die Eingaben der Tastatur an?
- Vorteil: Sehr feingranular: Man kann jede Handlung des Programms beobachten
- Nachteil: Hat große Laufzeitkosten

iii. Application-based Monitoring

A. Vorderseite Was ist Applikaitonsüberwachung? Welche Vorteile und Schwierigkeiten gibt es? (VL 10, S.8)

B. Rückseite

- Eine Mischung aus Host- und Networkmonitoring für eine besondere (verteilte) Anwendung.
- Beobachtung der Transaktionen und Logs der Anwendung
- Beobachtung bestimmter Applikationslogik und des Programmzustands: Wer hat Schreibrechte für bestimme Daten der Anwendung? Macht es Sinn, dass ein Datum geändert wurde, aber nicht ein anderes? (Applikationssemantik)
- Vorteil: Applikationsspezifische Angriffe werden erkannt
- Nachteile:
  - Applikation muss erweitert werden.
  - Wieder erhöhte Laufzeitkosten, allerdings nicht so hoch wie für die Beobachtung des ganzen Systems / Netzwerksegments.

iv. Code Emulation

A. Vorderseite Was ist *Code Emulation* im Kontext reaktiver Sicherheit? (VL 10)

B. Rückseite

- Ausführung des Programms / Öffnen des Dokuments in einer geschützten Umgebung
- Es wird eine bestimmte Zeit beobachtet / Programm bis zum Ende ausgeführt und geguckt, ob es schädliches Verhalten aufweist

2. Analysis and feature extraction

   (a) Netzwerkanalyse

      i. Problem

         A. Vorderseite Beschreiben Sie drei Schwierigkeiten beim Analysieren von Netzwerkdaten im Kontext von *intrusion detection.* (VL 10, S.13)

         B. Rückseite

            - Netzwerkdaten sind oft fragmentiert und nicht in Reihenfolge. Das System muss damit umgehen können.
            - Das System muss die Vielzahl an Protokollen des Internet-Layers und evtl. auch Application-Layers parsen und analysieren können. Und dabei dürfen nicht die selben Vulnerabilities der dahinterliegenden Hosts auftreten.
            - Die Payload-Daten sollten auch analysiert werden, was die Grenze zur Host-basierten Analyse auffweicht.

      ii. Vor- und Nachteile

         A. Vorderseite Was sind die Vor- und Nachteile eines Netwerk-basierten *Intrusion Detection Systems*? (VL 10, S.13)

         B. Rückseite

            - Vorteil:
               - Analyse erfolgt, bevor der Zeilhost erreicht wird
            - Nachteile:

- Eine Große Anzahl an Protokollen, die korrekt und robust implementiert werden müssen.
- Verschiedene Protokollimplementierungen können bei der Verarbeitung der Daten zu unterschiedlichen Ergebnissen kommen -> *semantic gap*

iii. Semantic Gap

A. Vorderseite Was ist die *semantische Lücke* in Parsern und Protokollimplementierungen. Wie kann das als Schwachstelle genutzt werden? (VL 10, S. 13)

B. Rückseite

- Unterschiedliche Parser und Protokollimplementierungen können gleiche Daten anders interpretieren.
- Es kann passieren, dass das die Protokollimplementierung eines *IDS* nach dem Parsing keinen Angriff feststellt, obwohl der Parser des Hosts, auf den der Angriff abzielt, angreifbar ist.

(b) Host-basierte Analyse

i. Problem

A. Vorderseite Beschreiben Sie die Vor- und Nachteile beim Analysieren bei der Host-basierten Analyse Kontext von *Intrusion Detection.* (VL 10, S.14)

B. Rückseiten

- Vorteil:
  - Die Verteidigung ist effektiv, weil sie direkt man Angriffziel erfolgt
- Nachteile bzw. Schwierigkeiten:

- Der Schadcode kann obfuskiert (gepackt) sein. Das kann das Auffinden von Schadcode erschweren.
- Laufzeitkosten können die Nutzerfreundlichkeit reduzieren

ii. Aspekte der Host-basierte Analyse

A. Vorderseite Nennen Sie die drei Aspekte der *Host-basierten* Analyse im Kontext von *Intrusion Detection*. (VL 10, S.14)

B. Rückseite

- Der Angriff kann gepackt sein, daher muss es Maßnahmen zum Entpacken geben (*deobfuscation*)
- Statische Analyse aller Dateiformate und des Inhalts
- Dynamische Analyse von Code

iii. Obfuscation

A. Vorderseite Dient *Code Obfuscation* nur Angriffen? (VL 10, S.14)

B. Rückseite Nein, das Obfuskieren von Programmmechanismen im Code wird z.B. für Kopierschutz verwendet.

3. Detection concepts

(a) Feature Extraction

i. Definition

A. Vorderseite Was ist *Feature Extraction* im Kontext von *Intrusion Detection Systemen*? (VL 10, S.15)

B. Rückseite Die Überführung von Daten in eine analysierbare Repräsentation. Das kann sein:

- Statistische Analyse: Zählen von Länge, Häufigkeit von Symbolen, Vorkommen der selben Daten, Entropie
- Tokenization bzw. String Representation: Die Zerstückelung der Daten an bedeutenden Punkten. Erhöht Vergleichbarkeit.
- Parsen der Daten → Parse-Tree erlaubt syntaktische und semantische Analyse

(b) Misuse Detection vs Anomaly Detection

i. Misuse

A. Vorderseite Was ist *Misuse Detection* im Kontext von *Intrusion Detection Systemen*? (VL 10, S.20)

B. Rückseite

- Eine der zwei klassischen Schulen der IT-Sicherheit: Modelliert das Böse
- Definiert klare Missbrauchsmuster und ihre formale Repräsentation (genant Angriffssignatur, -heuristik, -muster)
- Ein Angriff ist, was vorher als Angriff definiert wurde.

ii. Anomaly

A. Vorderseite Was ist *Anomaly Detection* im Kontext von *Intrusion Detection Systemen*? (VL 10, S.20, 25)

B. Rückseite

- Eine der zwei klassischen Schulen der IT-Sicherheit: Modelliert das Gute

- Modelliert formal, was eine normale Nutzung des Systems darstellt (statistische Modelle, Spezifikationen, Wahrscheinlichkeiten, Maschinelles Lernen).
- Ein Angriff ist, was von der Normalität abweicht.
- Inhärente semantische Lücke: Anomalität ist nicht gleich Missbrauch.

iii. Misuse vs Anomaly

A. Vorderseite Stellen Sie *Misuse Detection* ("MD") der *Anomaly Detection* ("AD") gegenüber. Was sind die Vor- und Nachteile? (VL 10, S.21+)

B. Rückseite

- MD kann effizient und effektiv bekannte Angriffe erkennen. Eine Repräsentation von unbekannten Angriffen mit den Mitteln der MD ist nur sehr schwer zu machen.
- An der Stelle ist die AD effektiver. Ein Angriff sollte stark von der Normalität abweichen und ist daher erkennbar.
- AD kann jedoch gutwilliges, aber anomales Verhalten als Angriff einordnen (hohe Falsch-Positiv-Rate)

iv. Gefahren der Anomaly Detection

A. Vorderseite Was sind die Gefahren / Schwierigkeiten der Anomaly Detection? (VL 10, S.10,20,26)

B. Rückseite

- Inhärente semantische Lücke: Anomalität ist nicht gleich Missbrauch.

- Bei Machine-Learning-Modellen: Trainigsdaten, die Normalität abbilden, können den Angriff bereits beinhalten (→ vergiftet sein, *data poisoning*).

(c) Keyword-Tree

i. Vorderseite Keyword-Trees: Wie funktioniert der *Aho-Corasick* Algorithmus? Was ist seine Laufzeit? Was ist der große Vorteil? (VL 10, S.23+)

ii. Rückseite

- Siehe Vorlesung. (VL 10, S.23+)
- O(n)
- Es kann ein Baum für alle Keywords erstellt werden. Ein String, der gegen den Baum gematch wird, muss nur einmal durchlaufen werden, ohne je zu einer bereits besuchten stelle im Baum zurückzuspringen. Kann gut zur Misuse Detection eingesetzt werden.

4. Response

(a) Strategien

i. Vorderseite Nennen Sie drei übliche Strategien, wie ein *Intrusion Detection System* auf einen entdeckten Angriff reagieren kann. (VL 10, S.30)

ii. Rückseite

- Warnungen an Administratoren senden, Warnung loggen
- IP Adressen werden blockiert, Ausführung von Programmen gestoppt
- Infizierte Dateien werden in Quarantäne verschoben (manchmal ist sogar ein eine Reparatur der Datei möglich).

(b) Schwierigkeiten bei aktiven Reaktionen von IDS

i. Vorderseite Gibt es *Gefahren*, wenn ein *Intrusion Detection System* zur Verteidigung aktiv in das System eingreift? Wenn ja, nennen Sie Beispiele. (VL 10, S.30)

ii. Rückseite

- Ja, denn das präsentiert eine weitere Angriffsfläche. Alle falsch-positiven und falsch-negativen Einordungen führen zu ungewolltem Verhalten.
- Beispiele:
  - Der Angreifer kann eine Quell-IP Adresse spoofen und damit einen Angriff durchführen, dessen Erkennung den echten Besitzer der IP Adresse vom System ausschließt. (DoS)

(c) Intrusion Prevention System

i. Vorderseite Was ist ein *Intrusion Prevention System*? Was kann es tun? Birgt es eigene Gefahren? (VL 10)

ii. Rückseite

- Wird ein Angriff entdeckt: Anstatt nur einer Warnung werden auch Maßnahmen ergriffen:
  - Packet Dropping
  - Honeypot Redirection
  - IP Blacklisting
  - Datei in Quarantäne verschieben
- Nachteil:
  - Kann gehackt werden

(d) Honeypot

i. Vorderseite Was ist ein *Honeypot* im Kontext von *Intrusion Detection Systemen*? (VL 31, S.31)

ii. Rückseite Wird von einem netzwerkbasierten IDS ein Angriff entdeckt, können die Netzwerkpakete anstatt verworfen zu werden, stattdessen an ein einzelnes System weitergeleitet werden, welches nach einem wichtigen System (welches auch falsche vertrauliche Daten beinhalten kann) aussieht, aber eine geschützte Umgebung zur Beobachtung des Angriffs darstellt. Dieses System wird Honeypot genannt.

(e) Detection Performance

i. Definition

A. Vorderseite Wie misst man *detection performance*? (VL 10, S.34)

B. Rückseite Wie groß ist das Verhältnis von richtig-positiven Erkennungen zu im Vergleich zur Menge der falsch-positiven Erkennungen? → ROC Kurve. Für Details, siehe VL 10, S.34.

ii. Für Intrusion Detection Systeme

A. Vorderseite Warum ist für *Intrusion Detection Systeme* eine kleines Verhältnis von falsch-positiven zu richtig-positiven Erkennungen wichtiger, als eine möglichst hohe absolute Erkennung von Angriffen?

B. Rückseite Weil jeder falsch-positive Fall eine Angriffsfläche darstellt. (z.B. DoS für gespoofte IP Adressen, die aus dem Netzwerk ausgeschlossen werden)

(f) Angriffe gegen Intrusion Detection Systeme

i. Red herring

A. Vorderseite Was ist ein *Red Herring* allgemein und im Kontext der IT-Sicherheit? (VL 10, S.36)

B. Rückseite

- Ein Ablenkungsmanöver mit dem Ziel, jemanden auf eine falsche Fährte zu locken
- Man erzeugt gefälschte Angriffe, die ein *Intrusion Detection System* Alarm schlagen lassen, während ein tatsächlicher Angriff in der Masse von Warnungen untergeht.

ii. Mimicry Attack

A. Vorderseite Was ist ein *Mimicry Attack* auf ein *Intrusion Detection System*? (VL 10, S.36)

B. Rückseite Eine *Mimicry* (engl. Nachahmung) Attacke versucht den Angriff so normal wie möglich in den Augen des IDS aussehen zu lassen, um einer Entdeckung zu entgehen. → Ausnutzung einer semantischen Lücke.

iii. Poisoning

A. Vorderseite Was ist ein *Poisoning Attack* auf eine *Intrusion Detection System*? (VL 10, S.36)

B. Rückseite Wird ein IDS mithilfe vom Machine Learning auf die Erkennung von Angriffen trainiert, können Angriffsmuster in die Trainigsdaten für normales Verhalten untergemischt werden, damit das IDS diesen Angriff als normal ansieht.