Speicher- und Dateisysteme

Thema: Datenintegrität

Vortragender: Christian Rosenberg

Betreuer: Michael Kuhn

Gliederung

- Definition Datenintegrität & Methoden
- Memory Corruption & Disk Corruption
- ZFS

Definition

- Keine einheitliche Definition
- „Verhinderung unautorisierter Modifikation“
- „Korrektheit von Daten und Systemen“
- Arten von Integrität (nach Prof. Biskup)

Sequenznummern & Quittierungsmeldungen

- Sequenznummern

  → Vermeidung von Duplikaten

  → Berücksichtigung der Reihenfolge

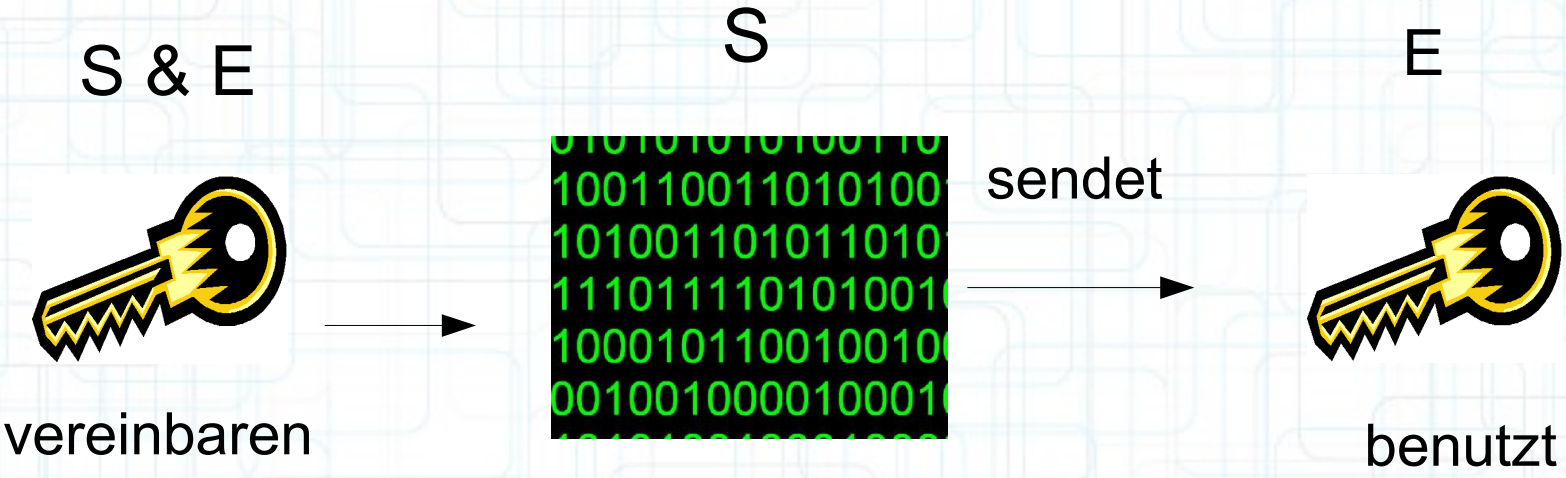
- Quittierungsmeldungen

  → Bestätigung von Erhalt oder Verarbeitung einer Datei

Prüfsumme & MAC

- Prüfsumme → Schutz vor unabsichtlicher Änderung & Falscheingabe

- Message Authentication Code (MAC)



CRC

- Zyklische Redundanzprüfung (CRC)

  → Schutz vor Zufallsfehlern
- 1. Bitfolge als Polynom
- 2. Generatorpolynom (GP) erstellen
- 3. Bitfolge / GP

  → Rest an Datenfolge
- 4. Neue Datenfolge / GP

CRC

- Beispiel CRC erzeugen :

  11011**00000** (Datenfolge + n Nullen)

  110101 (GP)

  0000110000

  110101

  ------

  101 (Rest) → 00101 (n-Stellig)

CRC

- Beispiel CRC-Prüfung:

  11011**00101** (Bitfolge + CRC)

  110101 (GP)

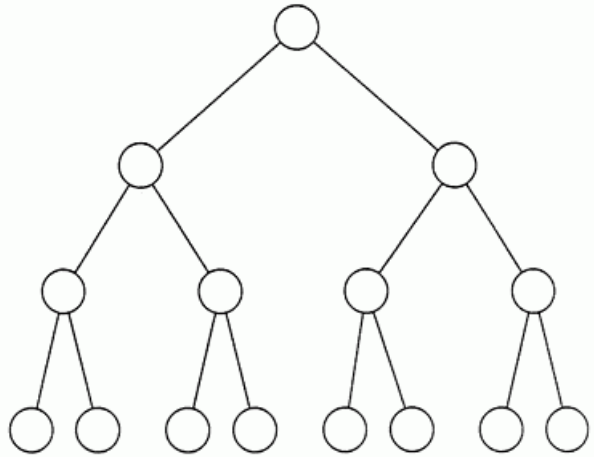
  110101

  110101

  000000

Hash-Baum & Hash-Funktion

- Meistens Binärbäume



- Jeder Zweig kann anhand Wurzel einzeln geprüft werden

→ Bei Signaturen und Veränderung von Daten

Hash-Funktion

- Beispiel zu Tiger-Funktion:

  "Franz/Frank jagt im komplett verwahrlosten Taxi quer durch Bayern"

  4df42db66c8d84269d4b7157b92a87be717a a1a5834a3050

  9cee0eb7b596ba0f435d42c33ddf8eff7fabb8 6922aa4bc6

Zusammenfassung

- Datenintegrität = Korrektheit von Daten
  → wichtig bei Datenübertragung und Datenhaltung

- Schutzmethoden

Memory Corruption

- Unbeabsichtigte Änderung des Speicherinhalts

- Folge: unerwartetes Verhalten
- Ursachen: physikalisch und softwaretechnisch
- Fehlerbehebung:

  1.ECC

  2.wiederherstellbare Programmmodelle

  3.Tools

Disk Corruption

- Daten in unerwartetem Zustand

- Durch physikalische Beschädigung oder Bugs in Systemsoftware
- Fehlerbehebung durch:
  1. Prüfsummen
  2. Redundanz
  3. RAID-Systeme

CERN-Studie

- 15.000 TB Daten an CERN-Institut untersucht
- Disk Errors: 500 Fehler an 100 Knotenpunkten
- RAID Errors: 2,4 Petabyte → 300 Fehler

  → Fehlerrate: 10^-14
- Memory Errors: 3 Fehler (zuviel)

CERN-Studie

- 8,7 TB Nutzdaten, 34000 Daten → 22 Fehler

  → 1 von 1500 Daten fehlerhaft

- Fazit: Fehlerrate insgesamt 3 * 10^-7

  → Fatal: Fehlerrate in einzelnen Komponenten

- 1 TB Festplatte → 3 fehlerhafte Dateien

Zusammenfassung

- Memory Corruption – Beschädigung der Speichereinheit
- Disk Corruption: fehlerhafte Daten auf Datenmedien

- Memory Corruption & Disk Corruption unterschätzte Probleme
- Ausmaß größer mit steigender Datenmenge

ZFS

- Zettabyte File System: von Sun Microsystems entwickelt
- 128 Bit
- Copy-On Write:

  → Datenblöcke an freien Platz geschrieben

  → Ziel: Vermeidung von Kopiervorgängen

ZFS

- Software-RAID:
  Redundanzen physischer Datenträger
  → Vorteil: ZFS unterscheidet freie und
  belegte Datenblöcke
  → Zeitersparnis
  - Paritätsgesichtert → keine „Write Holes“

ZFS

- Prüfsumme: Jeder Datenblock versehen
- Mehr Fehler erkannt als bei normalen Prüfsummen

→ Dateisystem ständig konsistent

ZFS

- Deduplizierung:

  redundante Datenblöcke eliminiert

  → Speicherplatz gespart

- Technische Details:

  Max. Größe Datei(system): 16 EiB

  Max. Anzahl Dateien: 2^48

Zusammenfassung

- ZFS ist Dateisystem für sehr große Datenmengen

- Besonderheit: aktive Sicherstellung von Datenintegrität

- Ehemals für Server konzipiert, mittlerweile auch auf Privatrechnern verwendbar

Vielen Dank für eure Aufmerksamkeit !!!

Quellen

- http://www.zdnet.com/blog/storage/data-corruption-is-worse-than-you-know/191
- http://institute.lanl.gov/resilience/conferences/2009/HPCResilience09_Michalak.pdf
- http://en.wikipedia.org/wiki/Data_corruption
- http://hub.opensolaris.org/bin/view/Community+Group+zfs/selfheal
- http://www.usenix.org/event/fast10/tech/full_papers/zhang.pdf
- https://blogs.oracle.com/bonwick/entry/zfs_end_to_end_data
- http://en.wikipedia.org/wiki/Message_authentication_code
- http://en.wikipedia.org/wiki/Checksum
- http://en.wikipedia.org/wiki/Hamming_code
- http://de.wikipedia.org/wiki/Zyklische_Redundanzprüfung

Quellen

- http://de.wikipedia.org/wiki/Hash-Baum
- http://en.wikipedia.org/wiki/Hash_tree
- http://oss.oracle.com/projects/data-integrity/
- http://www.spinics.net/lists/linux-btrfs/msg14359.html
- http://en.wikipedia.org/wiki/Btrfs#Checksum_tree
- http://de.wikipedia.org/wiki/Sequenznummer
- http://de.wikipedia.org/wiki/ZFS_(Dateisystem)
- http://de.wikipedia.org/wiki/Hashfunktion