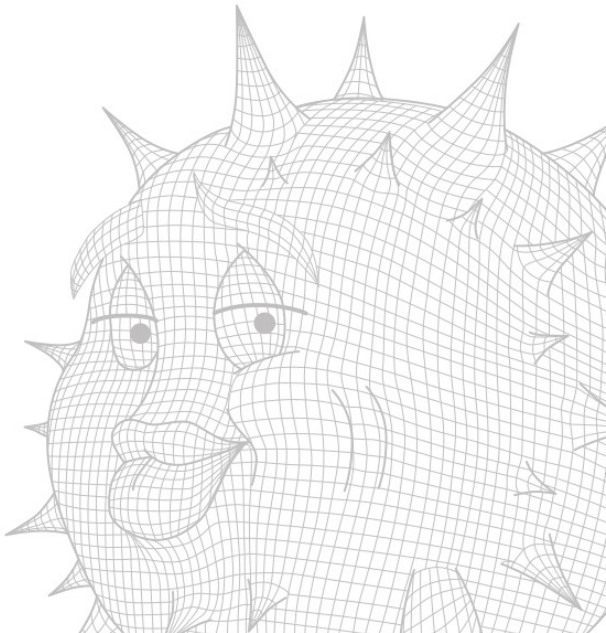
IT-Security

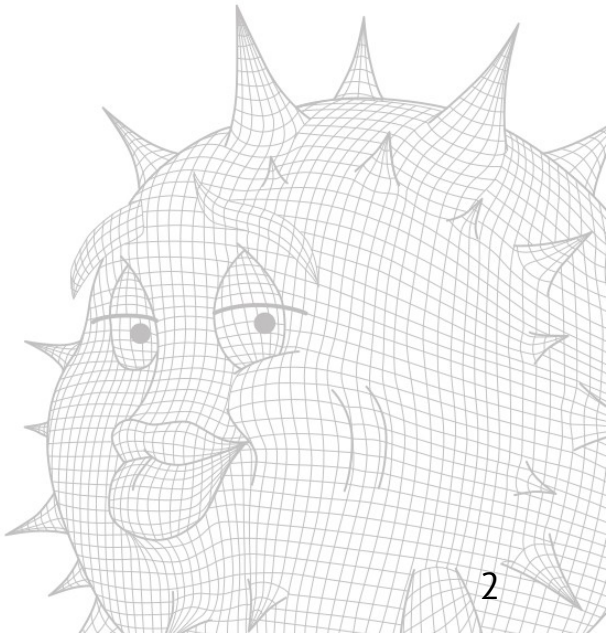
ein kurzer Rundgang

Henning Brauer
<info@henningbrauer.com>



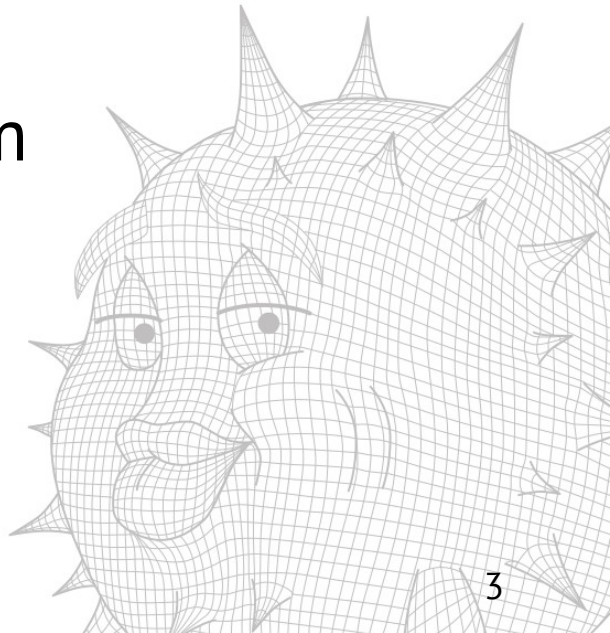
Wer spricht da eigentlich

- OpenBSD-Entwickler seit 2002
  - unixoides Betriebssystem mit Fokus auf Security
  - gilt als eines der sichersten Systeme der Welt
  - Keimzelle vieler Sicherheitstechniken
  - Basis vieler Firewall- und VPN-Appliances
- Author von pf, OpenBSD‘s Paketfilter
  - auch Standardfirewall von Apple OS X und iOS, Oracle Solaris, QNX
- Author von OpenBGPd und OpenNTPd
- Security- und Network-Consultant
  - Netzbetreiber, Sicherheitsbehörden, Finanzdienstleister, …



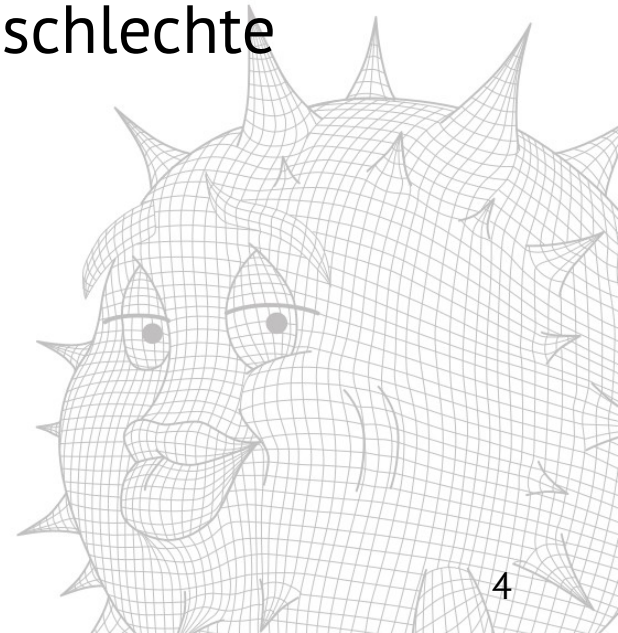
Das Problem ist nicht neu

- Morris Worm legte 1988 das Internet fast lahm
  - „das Internet“ damals: ~60000 Systeme, 10% befallen
  - sollte gar kein Schadprogramm sein, Programmierfehler machte es zu einem



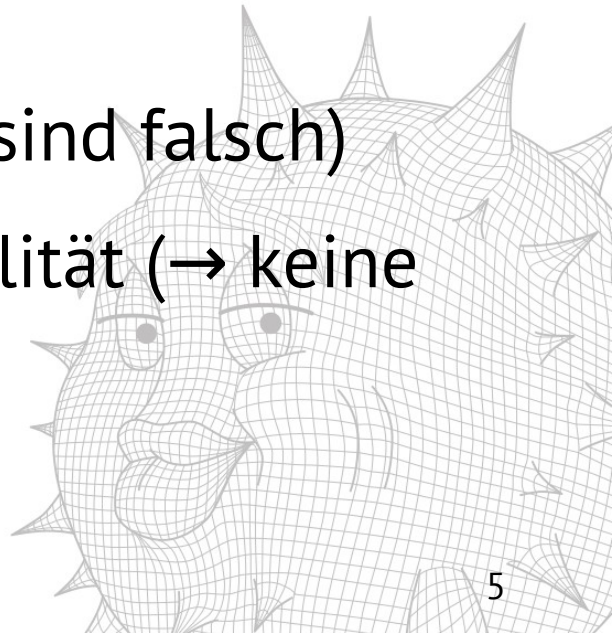
Problemursachen

- Softwarefehler
- Falscher Softwareeinsatz
  - Vergessene „Hintertüren“, Admin-Zugänge, schlechte Passwörter
- Anwenderfehler
  - Mailanhang SieHabenGewonnen.doc.exe



Softwarefehler

- ursächlich ist mangelhafte Software-Qualität
  - Programmierer wissen es nicht besser (→ Ausbildung, Problembewusstsein)
  - Security als Afterthought (→ Prozesse sind falsch)
  - Kunden zahlen für Features, nicht Qualität (→ keine Zeit für sauberen Code)

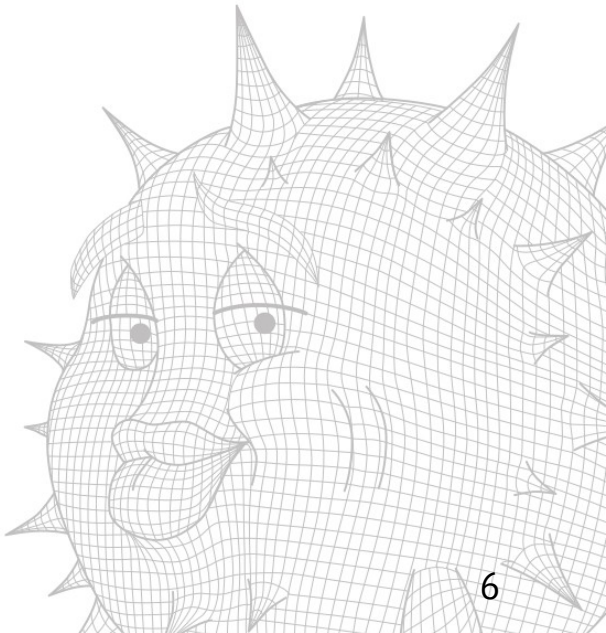


sichere Software ist möglich

```
int
main(int argc, char *argv[]) {
   char foo[4]; /* reicht für 3 Zeichen + Nullbyte */
   char bar[20];

   strcpy(foo, argv[1]);
}
```

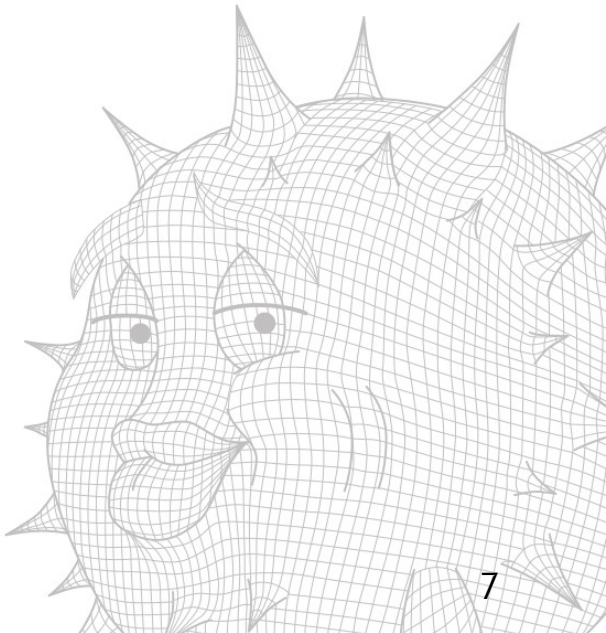
- buffer overflow wenn argv[1] länger als 3 Zeichen ist
- Angreifer kann bar überschreiben – und mehr



sichere Software ist möglich

- Problem 1: keine Input Verification

  was nicht statisch im Programm ist, ist untrusted Input und muss verifiziert werden



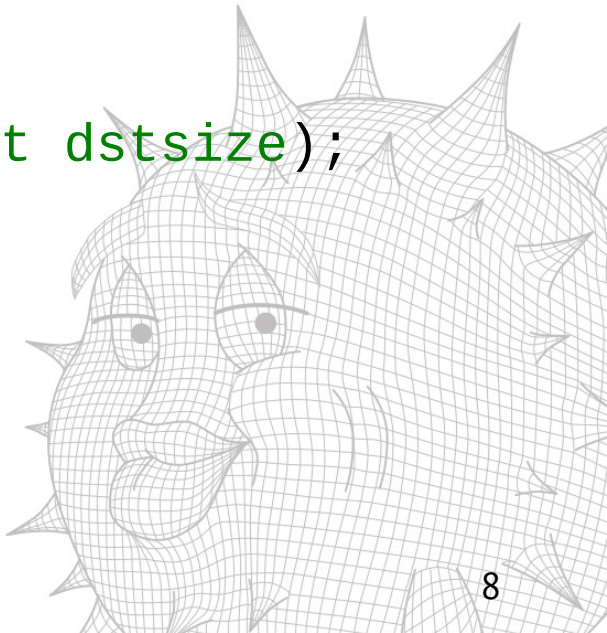
sichere Software ist möglich

- Problem 2: schlechte APIs begünstigen Fehler

```
strcpy(char *dst, const char *src);
```

- es gibt meist bessere Alternativen

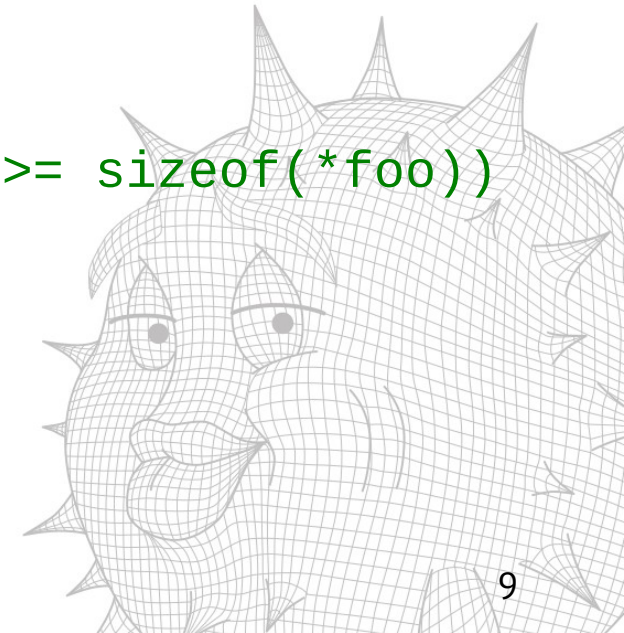
```
strlcpy(char *dst, const char *src, size_t dstsize);
```



sichere Software ist möglich

```
int
main(int argc, char *argv[]) {
   char foo[4]; /* reicht für 3 Zeichen + Nullbyte */
   char bar[20];

   if (strlcpy(foo, argv[1], sizeof(*foo)) >= sizeof(*foo))
      errx(1, „input too long“);
}
```

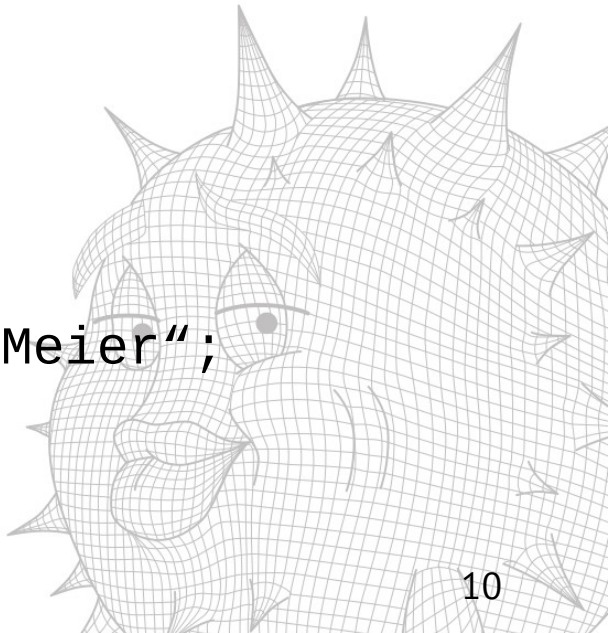


auch in webapps

http://directory.example.com/lookup?name=Meier

```
$sql = „SELECT * FROM Users WHERE lastname=\“” .
    $_GET[name] . „\“;“;
$conn→query($sql);
```

- `SQL: SELECT * FROM Users WHERE lastname=“Meier“;`



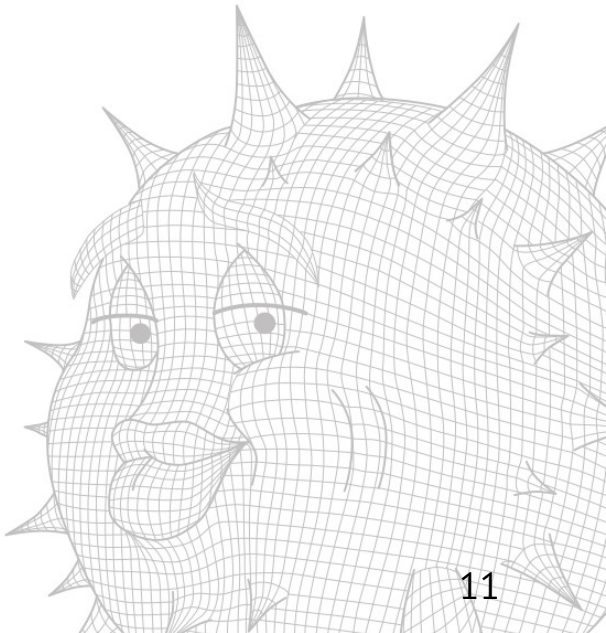
auch in webapps

http://.../lookup?name=Meier%22%3BDROP%20TABLE%20Users%3B%22

- SQL:

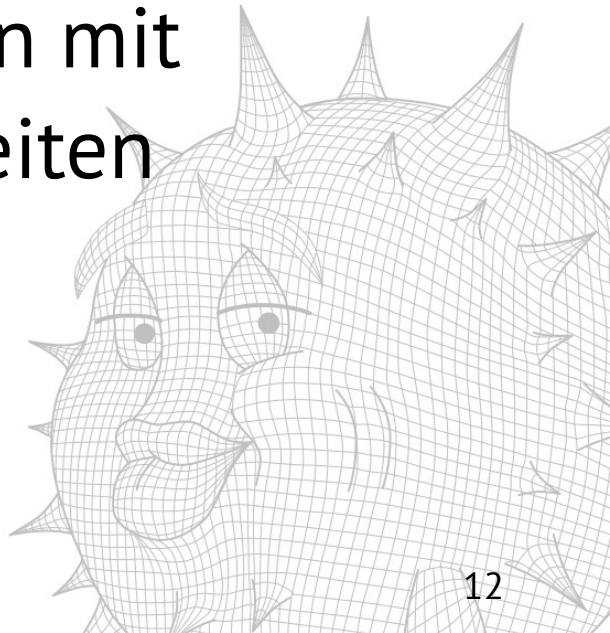
  SELECT * FROM Users WHERE lastname="Meier";DROP TABLE Users;

- Fehlerursache: keine Input Verification



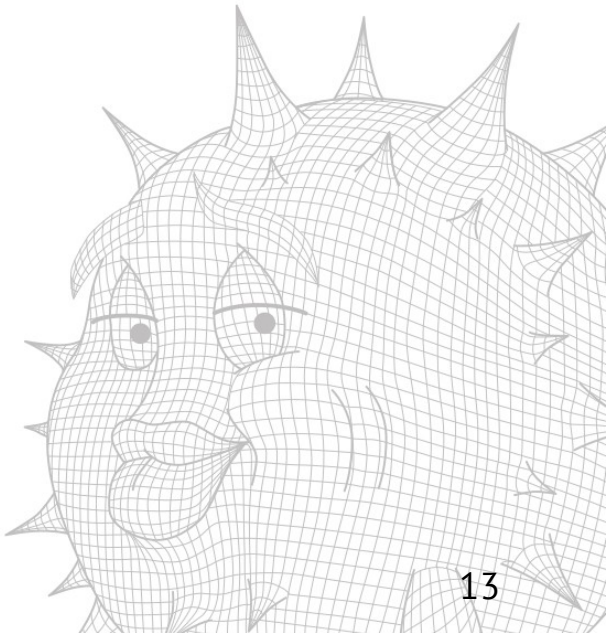
Principle of least Privilege

- Code sollte immer nur mit den minimal nötigen Privilegien laufen
- Netzwerk-Input niemals in Prozessen mit root-/Administrator-Rechten verarbeiten
- Privilege Seperation nutzen



Guidelines für Entwickler

- Design first
  - Sauberes Softwaredesign vermeidet Fehler und unwartbaren Spaghetticode
- erst denken, dann coden
- 4 Augen sehen mehr als 2
  - peer review vor dem commit
- Komplexität reduzieren



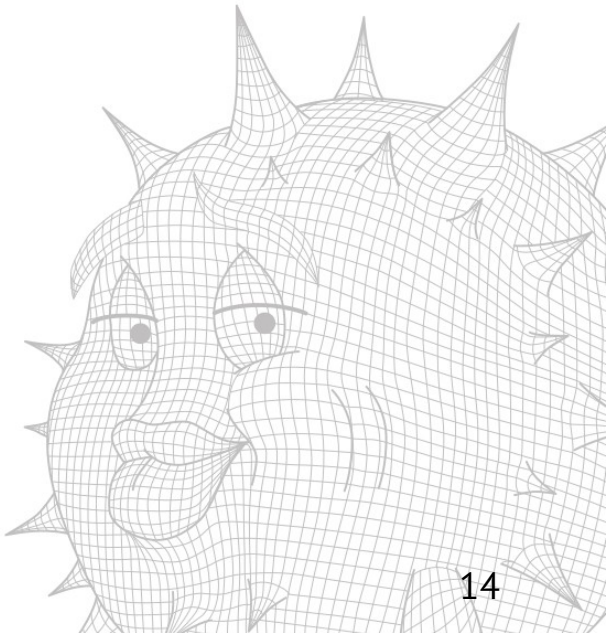
auch Systemadministratoren

Keep

It

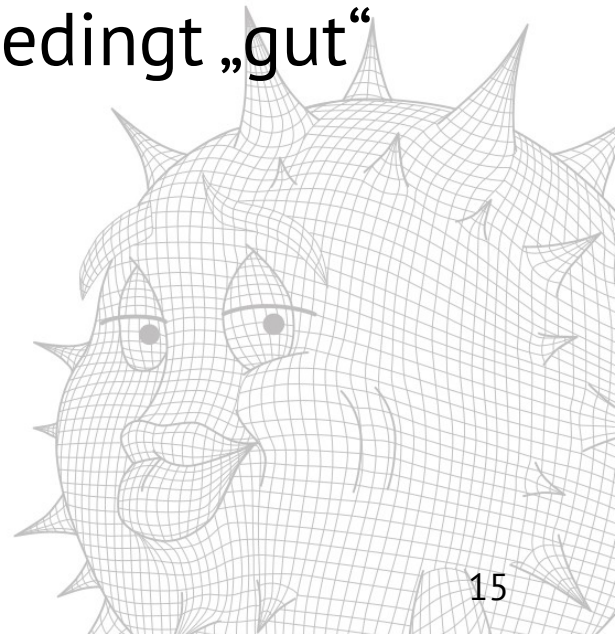
Simple &

Stupid



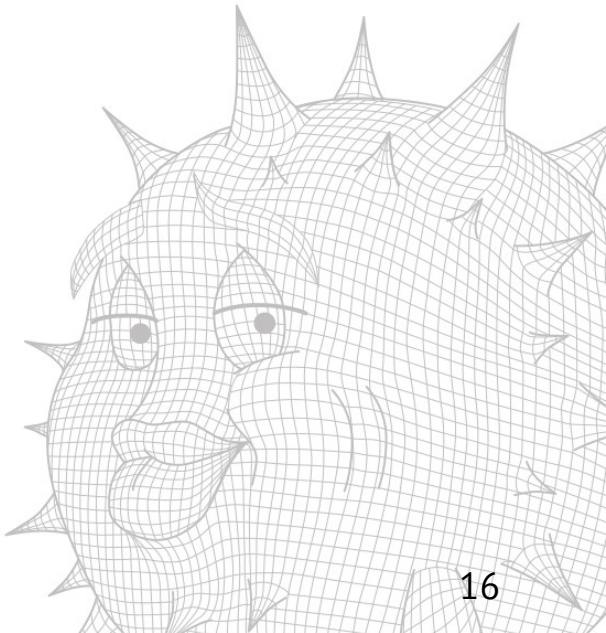
Systemadministratoren

- Open Source tendenziell besser als Closed
  - nicht immer
  - besser als „sehr schlecht“ ist nicht unbedingt „gut“



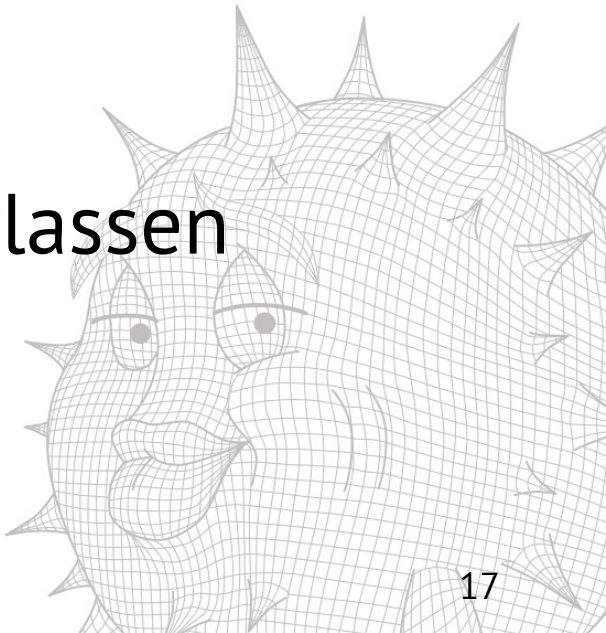
Angriffsoberfläche reduzieren

- was nicht da ist, kann nicht attackiert werden
- was nicht zugänglich ist, kann nicht attackiert werden
- „kann das weg?“
  - Altlasten entsorgen!



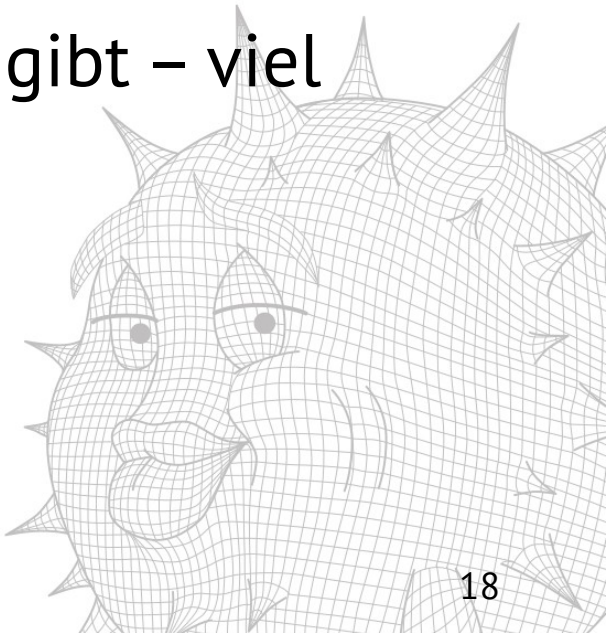
Angriffsoberfläche reduzieren

- Netzwerke segmentieren
- Zugriffskontrollen
- Firewalls nutzen
- Nie auf eine einzige Massnahme verlassen
  - „multiple layers of defense“



Software aktuell halten

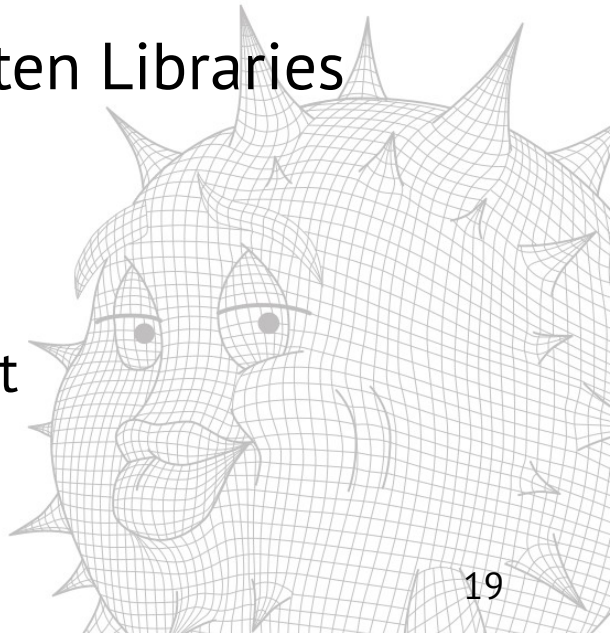
- Probleme werden gefunden & gefixt
- automatische Updates aktivieren!
  - selbst wenn es gelegentlich Probleme gibt – viel mehr Vor- als Nachteile



Negativbeispiel: Docker & Co

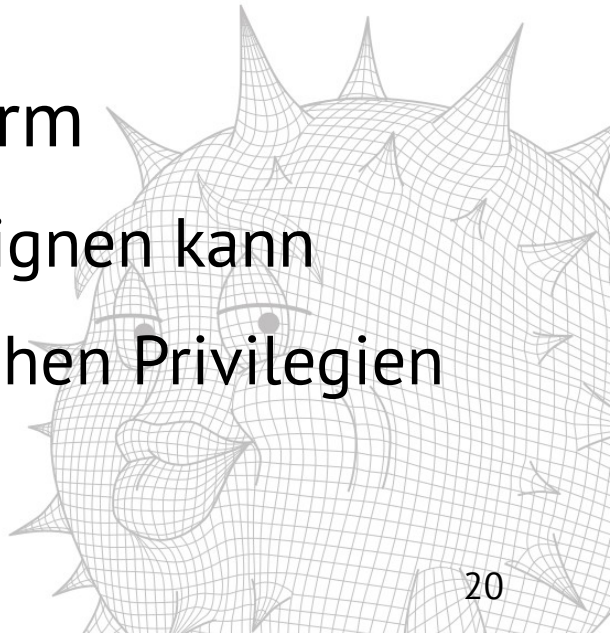
- sind die ultimative Bankrotterklärung vor überbordender Komplexität
  - „Ich habe shared libraries nicht verstanden“
- Nahezu alle Docker-Images „in the wild“ mit alten Libraries voller bekannter Fehler
- Updates der Systemlibraries helfen nichts

  → Sicherheitslücken werden in der Praxis nie gefixt



Negativbeispiel: Virenscanner

- nur reaktiv, also immer zu spät
  - Heuristiken, „adaptive“, … funktionieren nur in den Werbebroschüren
- Vergrössern die Angriffsoberfläche enorm
  - alles falsch designed, was man falsch designen kann
  - sehr viel sehr schlechter Code mit sehr hohen Privilegien

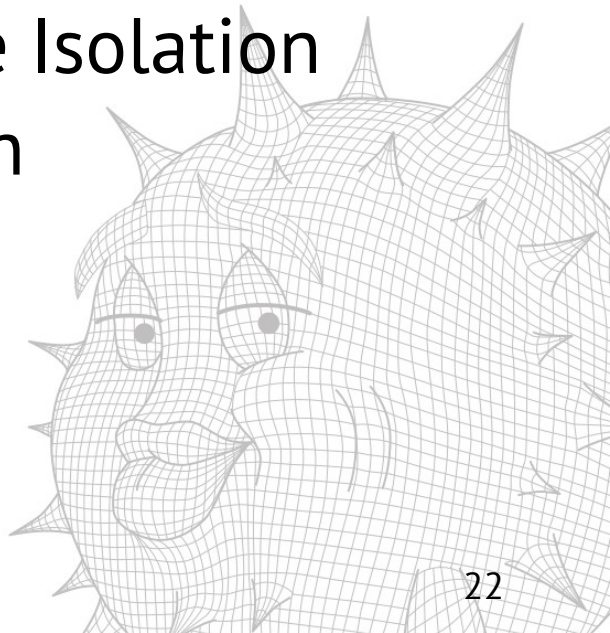


Security Software / Appliances

- Kaufen Sie ***TotalUltraThreatDefender*** ©™®☺♭€€€€

Cloud

- Es gibt keine Cloud, es gibt nur die Server der Anderen
- Problem verschoben, aber nicht behoben
- Häufig verschlimmbessert: mangelhafte Isolation zwischen Instanzen/Kunden/Anwendern
- Anbieter greifbar?



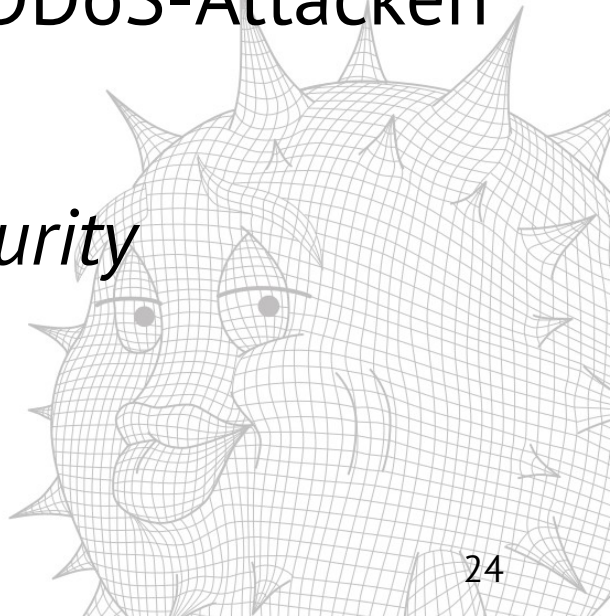
“serverless”

- Essen Sie ***kitchenless***, wenn Sie Pizza bestellen?
- Auch hier: Problem nur verschoben
- Generelles Problem der IT-Branche – viele buzzwords, bei genauerer Betrachtung nichts als heisse Luft

Internet of ~~Terror~~ Things

- Grausam: meist schnell & billigst irgendwo irgendwie zusammengeklöppelte Hard- und Software
- Eine der grössten jemals gemessenen DDoS-Attacken kam von webcams

The „S“ in „IoT“ stands for Security



IT-Security ernst nehmen

- IT-Security ist ein Prozess, kein Zustand
- Denken Sie bei jeder Veränderung Ihrer IT-Landschaft über Security-Implikationen nach
- Geben Sie Mitarbeitern Zeit
- Schulen Sie Ihre Mitarbeiter
- Holen Sie sich Unterstützung