Sichere Webportale –

Wie stell ich's an?

Inhalt

1. **Stereotyp: Worst Practice**
   - > **Die drei Dilemmas**
2. **Software-Sicherheit**
   - > **Best Practice**
3. **Audit**
   - > **Status Tools**

Disclaimer / Ernüchterung

1. **40 Minuten sind viel zu wenig**
2. **Keine coolsten Hacks**
3. **Keine Software-Techniken**
4. **Keine „Raketenwissenschaft“ ;-)**

1. Die drei Dilemmas

- **Dilemma 1: Software**
  - > Entwicklung/Kauf + Deployment:
    - **Wichtigste nicht-funktionale Kriterien:**
      - Preis (Saturn-Hansa lässt grüßen)
      - Schnell, Schnell
    - **Zeitdruck**
      - *„Projektplan einhalten: top Prio!“*
      - Bestenfalls:*„Wir machen Sicherheit später“*
        - » Technisch wenig sinnvoll
        - » Finanziell ebenso: Redesign teurer
      - Schlimmstenfalls:
        - » (No Plan,) Build, Run ~ „Cowboy Coding“

1. Die drei Dilemmas

- **Dilemma 1: Software**
  - > Eigene Statistik mittelprächtig (mit Ausnahmen)
    - **Probleme in Produktion aufgefallen**
      - Glücklicherweise Kurve bekommen
      - Wieviele andere da draußen nicht?
    - **Warum glauben viele Betreiber ungetesteter Portale, dass diese sicher sind??**
  - > Problem:
    - **Prozess Software-Patch**
      - **langsamer als OS-Patch!** (!Inhouse)
        - » Bestellvorgang
      - **Manchmal gibt es gar keinen**
        - » Ex-Mitarbeiter, Student
        - » Keine Software-Wartung

1. Die drei Dilemmas

- **Dilemma 2: Audit**
  - > Verständnisproblem Auftraggeber:
    - **Infrastruktur- vs. Websicherheit („*Wir haben doch ne Firewall*“)**
    - **Ähnliches stellenweise beim Dienstleister**

→ „Pentester“ macht Nessus-Scan (schlimmstenf.: PDF=Report)

Komplett andere Baustelle:

1. Die drei Dilemmas

- **Dilemma 3: Technikgläubigkeit**
  - > 'Ne WAF wird's schon richten
    - **Schmeiß ein Tool drauf und schon ist die Welt i.O.**
    - **Ignoriert**
      - Ursache: Software
      - Komplexität: Websicherheit
      - Nicht vergessen:
        - » WAF != Firewall (trotz „F“)
          - > Komplexität Layer 3/4 vs. Layer 7
          - > Kein Verhüterli gegen „alles“ (→ Best Practice WAF)
            - Wer konfiguriert's?
            - ...

1. Die drei Dilemmas

- **Compliance-Anforderungen? (branchenspezif.)**
  - > Datenschutzgesetze (B2C)
  - > Regulatorien
    - **Direkt: PCI-DSS + ISO 2700x**
    - **Indirekt + abhängig von Gesellschaftsform:**
      - SOX, Basel II, KonTraG, ..
      - Finanzkrise: Haben ja gesehen, wie gut's funktioniert ;-)
      - Zweifel, ob Software-Sicherheit dann ein Argument ist

Buzzword Bingo / Verkaufsargument?

→ Wo kein Richter, da kein Henker

2. Software-Sicherheit

- **Strukturiert vorgehen!**
- **Ursachenorientiert agieren**

- **Richtiges Abwägen mit Risiko fürs Geschäft**

2. Software-Sicherheit

- **Herkunft Webapplikation**

  > Drei Typen, Ursprung:
  - COTS (nicht nur „C“, auch OSS)
  - Outsourced
  - Inhouse

  > Egal welcher Typ:
  - **Sicherheit als Kriterium zu berücksichtigen**

    →Managemententscheidung!

2. Software-Sicherheit

> COTS

- **Metriken über SW**
  - Security-Advisories (Anzahl, Severity)
  - Hersteller und ähnliche Software?
- **Quellcode + Doku?**
  - Sicherheitsarchitektur?

- **Absicherung:**
  - Recht: Wer haftet?
  - Audit (abh. von Einsatz und geschäftl. Risiko)

2. Software-Sicherheit

> Outsourced

- **Hat Firma Reputation (gut/schlecht/keine) in puncto Sicherheit?**
- **Nachfragen bei Anbieter**
  - Quellcode + Doku?
  - Sicherheitsarchitektur?
  - Jede Frage: Awareness auf Anbieterseite
- **Vorgaben machen:**
  - Sicherheitskriterien im Vertrag (Anwalt):
    - » OWASP Secure Software Contract Annex
    - » SANS Application Security Procurement Language
    - » Erfolgreicher Audit im Vertrag als Abnahmekriterium
  - Praxis:
    - » Die wenigsten versuchen's
    - » Leider auch nicht immer möglich

2. Software-Sicherheit

> Inhouse

- **SDLCs, Entwicklungsmethoden:**
  - Konservative (z.B. einfaches Waterfall-Methode):
    - » Requirements > Design > Implement > Test > Maintenance
  - Agile Entwicklungsmethoden (XP, Scrum, RAD, …)
    - » Keine langfristige Planung (Scrum: < 30 Tage)
    - » Kleine Tasks mit kleinen Teams (auch R>D>I>T>M z.B.)
    - » Iterativ Ziel nähern

- **Agile Entwicklung = Rückschritt in puncto Sicherheit ?!**
  - Nur 46% Anforderungen an sichere SW-Entwicklungsprozesse

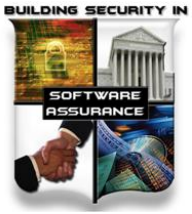
2. Software-Sicherheit

> Inhouse

- **SDLC, Prozesse:**
  - QM: ISO 900x
  - SPICE (SW Process Improvement and Capability Determination, ISO 15504)
  - CMMI, CMMI-DEV (DEV = Prozessentwicklung)
  - ISO 12207 = Software Life Cycle Processes

- **Nur: Wie Entwicklungsmethoden: keinen Fokus auf Sicherheit**

- **Interessant (NCSD vom DHS)**
  - BSI: Build Security In
  - Software Assurance (SwA)



2. Software-Sicherheit

> Inhouse

- **Modelle/Prozesse + Sicherheit:**
  - MS SDL (>= 2004), Simplified / Process Guidance: 19/148 Seiten
  - OWASP CLASP (2006), Ursprung IBM
    Comprehensive, Lightweight Application Security Process
    - » Lightweight?? ;-)

  - Cigital + Fortify: SSF (Software Security Framework) /
    BSIMM2 (Building Security In Maturity Model)

  - OWASP's OpenSAMM (Software Assurance Maturity Model)
    - » Inkl. Assessment (externe Questionaire)

2. Software-Sicherheit

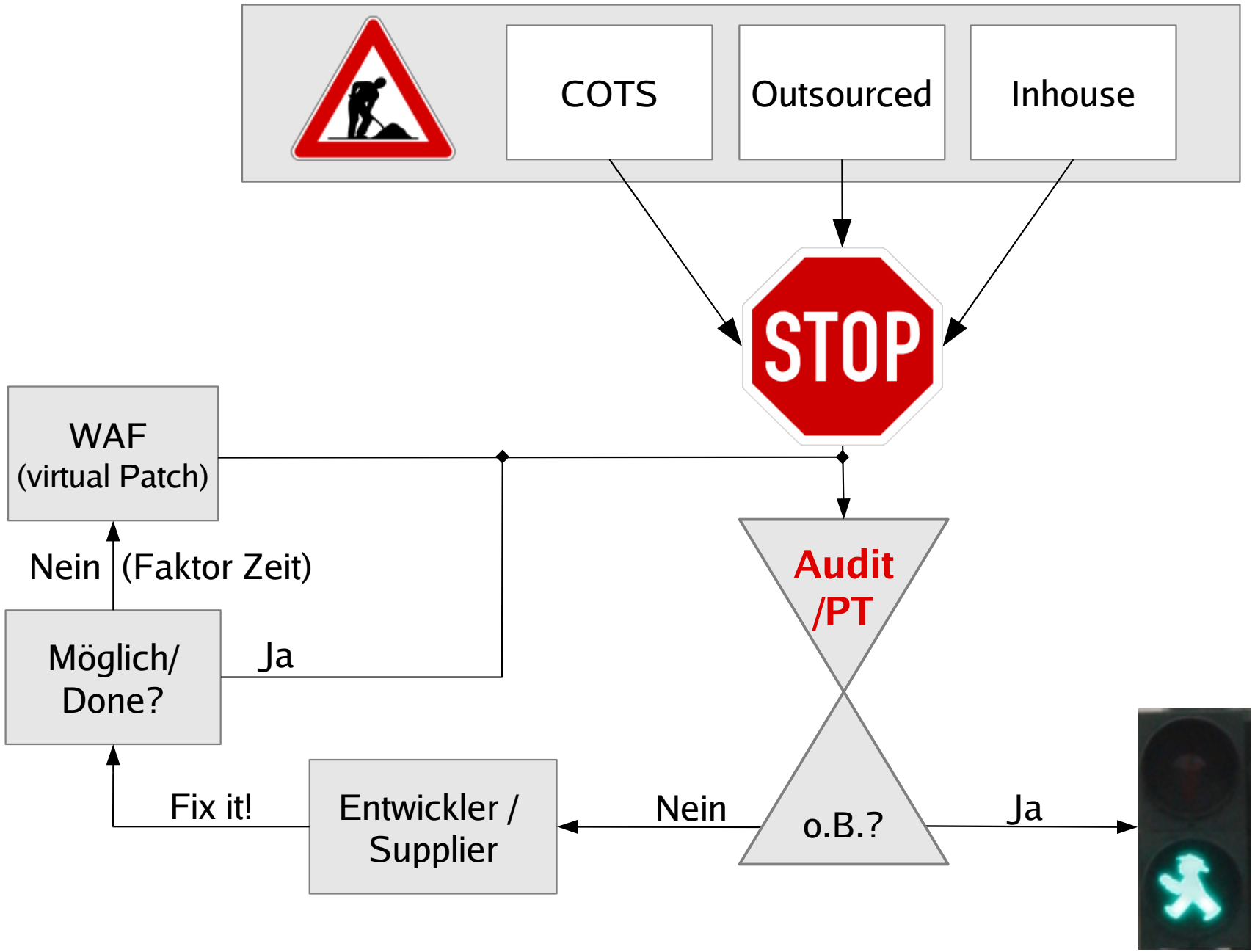
> Inhouse

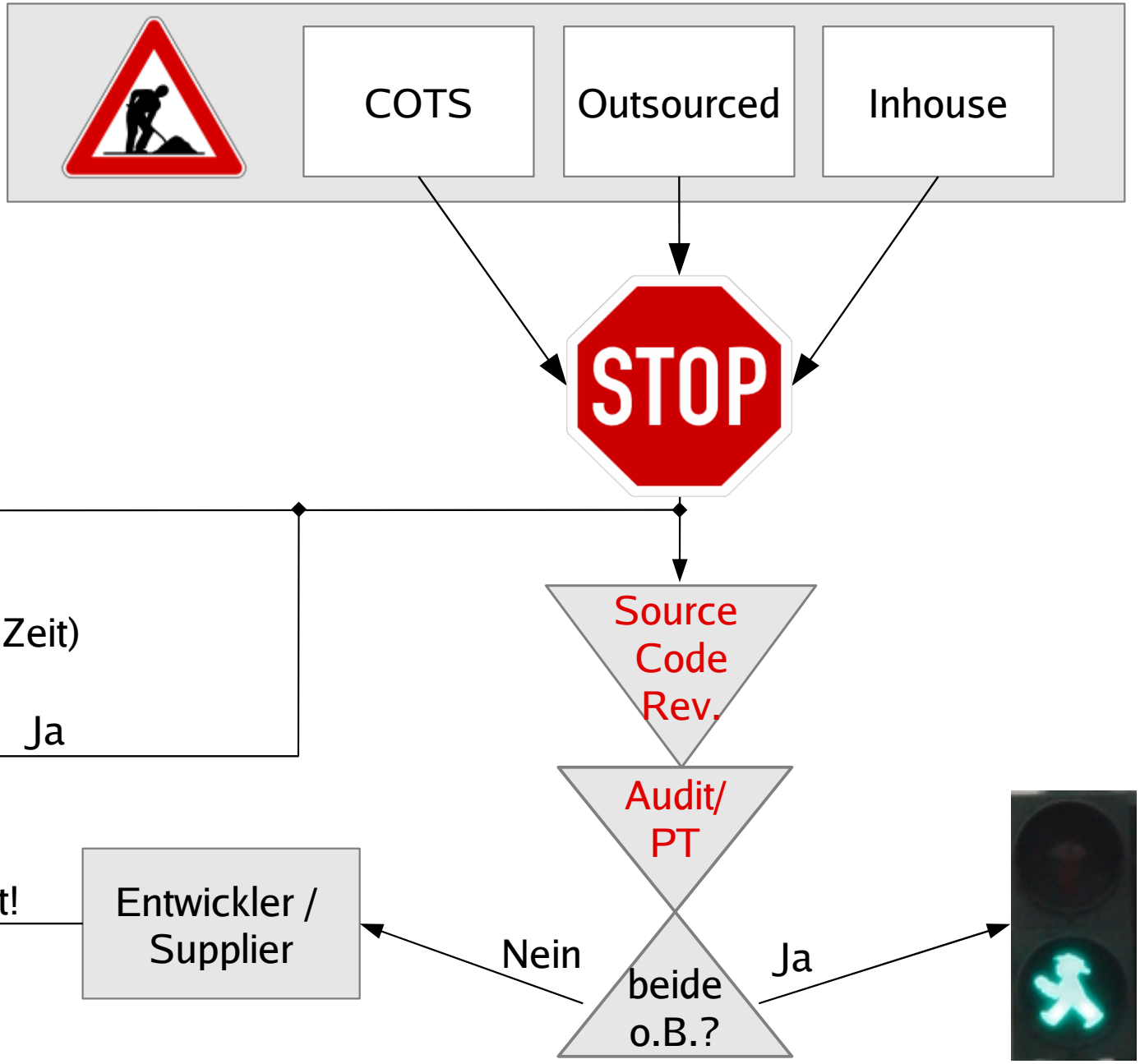
- **Entwickler**
  - Einige Modelle gehen technisch ins Detail, use it!
  - In a nutshell:
    - » Sich *vorher* Gedanken machen
    - » Sicherheitsarchitektur Bestandteil der Softwarearchitektur
      - > Zentrale Funktion zur Eingabeüberprüfung
      - > (Zentrale) Fehlerbehandlungsroutinen
      - > Stored Procedures / Prepared Statements
      - > uvm.
    - » „Lebender Prozess“, d.h. ständig Verbessern

2. Software-Sicherheit

> Inhouse

- **Entwickler**
  - Gute Quellen:
    - » SANS (Einstieg):
      - > Protecting your Webapps,
      - > What works-Poster
    - » Security Guides: PHP, RoR, Java?, (ASP.NET)
    - » BSI Websicherheit (2006, leider wenig Java, .NET, kein RoR)
  - OWASP, V2 von 2005 (2010 WIP):
    A Guide to Building Secure Web Applications and Web Services





3. Audit

- **Prinzipiell mehrere Möglichkeiten:**
  - > Code-Review/-Analyse (statisch)
  - > Audit (aka Pentest / Vulnerability Assessment)

  - > Besonders bei letzterem:
    - **Intern: kann man machen**
      - Sofern nicht Programmierer selbst
      - Und nicht befangen (verschiedene Abteilungen/Chefs)
    - **Extern**
      - Unbefangen
      - Firmenexterne Perspektive
      - i.d.R. Experten auf just diesem Gebiet

3. Audit

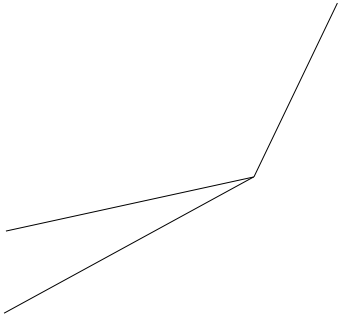
- **Assessment-Werkzeuge für Audit**
  - > Freie
    - Proxies
    - Aktive Scanner
    - Browser-Plugins
  - > Kommerzielle
    - Fast nur Scanner
  - > Beschränkung auf die wichtigsten (YMMV)

3. Audit

> Freie

- **Proxies:**
  - Burpsuite (Portswigger, 2010)
  - Paros (2006)
  - Webscarab (Webscarab-NG) (OWASP, 2007)
  - ratproxy (passiv, CSRF) (lcamtuf, stale)
  - Tamper Data (FF Plugin)

- Burpsuite Professional
- Paros Pro (MileSCAN Web Security Auditor)



3. Audit

> Freie

- **Scanner (viel getan in letzter Zeit)**
  - w3af (1.0rc3, Debian)
  - Grendel-Scan (1.0, stale)
  - websecurify NEW (pdp, 0.5)
  - XSS Me (FF Plugin) NEW Secom labs
  - SQL Me (FF Plugin) NEW (SAMM questionaire)
  - skipfish NEW (lcamtuf)

3. Audit

> Kommerzielle

- **Sich bewegender Markt, teuer**
  - HP WebInspect (ex SPI Dynamics)
  - IBM Rational Appscan (ex Watchfire)
  - Cenzic Hailstorm
  - Acunetix WVS
  - NTOSpider (NTO = Foundstone Spinoff)
  - art of defense: hyperscan (2008 Server/Appliance)
  - Mativuna Netsparker NEW
  - Qualsys WAS NEW
  - Rapid7 – NeXpose NEW  (*auch* )

4. Conclusio

> Wurzel des Ganzen: Software

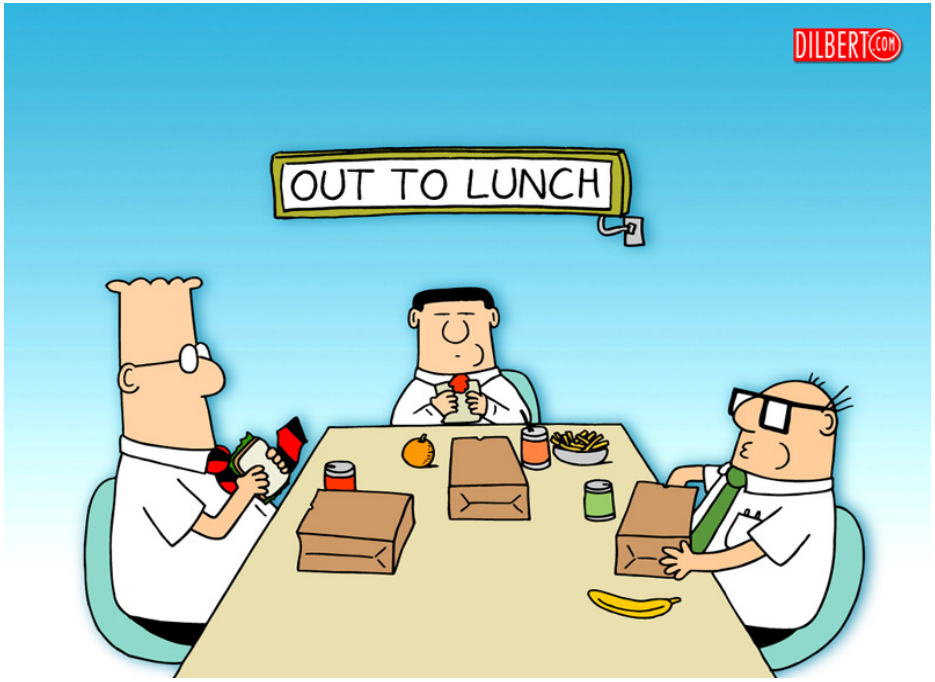
- **Unabhängig von der Herkunft: Sicherheit!**
  - Nicht nur Frage der Entwickler
  - Managementsache!

> Vor Deployment ins WWW:

- **Default: Unsicher (unless otherwise proved)**
- **Audit: Better be safe than sorry**

Danke fürs Zuhören!

Fragen? Oder gleich



mail ät drwetter punkt eu