



**OWASP**

Open Web Application  
Security Project

# **Denial of Service auf Applikationsebene**

## **OWASP-Stammtisch FFM**

### **30.10.2014**

#### **Alexander Klink**

#### **PricewaterhouseCoopers AG WPG**

# Agenda

- Abgrenzung
- HashDoS
- XML Entity Expansion
- Dekomprimierung
- Regular Expressions
- Benutzer aussperren
- Gegenmaßnahmen



# Abgrenzung

- Applikationsebene
- Kein (reiner) Netzwerk-DoS
- DDoS sollte nicht nötig sein
- Möglichst „clever“
- Wo enden Performance-Probleme und wo beginnt DoS?



# HashDoS

- Hashtables / Dictionaries
- $h[\text{'key'}] = \text{value}$
- Worst case Einfügekomplexität:  $O(n^2)$
- Crosby & Wallach (Usenix Sec 2003): Perl
- Klink & Wälde (2011): PHP, ASP.NET, Java, Python, Ruby, v8, ... – ein POST-Request langte
- Mittlerweile oft Gegenmaßnahmen, wie randomisiertes Stringhashing / SIPHash, etc.



# HashDoS (II)

- Aber: Hashtables, die andere/eigene Objekte als Schlüssel haben, sind ggf. weiterhin verwundbar (vgl. MS14-053)
- Bedenken: Kann ein Angreifer Hashtables füllen, gibt es Begrenzungen, etc.?
- Gerne auch bei Parsing (JSON, YAML, ggf. XML) verwendet.

# XML Entity Expansion

- AKA „Billion Laughs“-Angriff

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ELEMENT lolz (#PCDATA)>
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

- Verbraucht ~3 GB RAM beim Parsen



# *XML Entity Expansion (II)*

- Einfache und effektive Gegenmaßnahme:
- DTD-Parsing verhindern
- (verhindert auch eine Handvoll andere Security-Probleme)

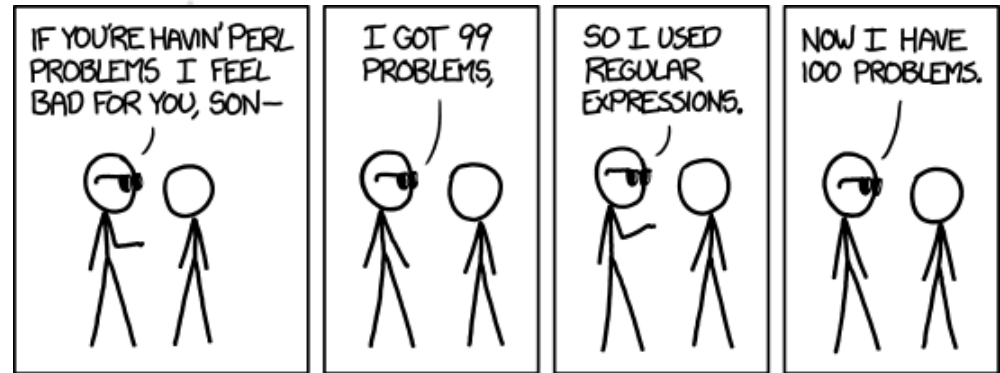
# Dekomprimierung

- Gerne mal transparent
- 256 GB Nullen lassen sich problemlos in wenige Hunderte KB komprimieren
- Limits einführen
- Libraries testen
- Unterstützt \$format Komprimierung?





# Regular Expressions



- Der Parser des kleinen Mannes
- Regex-Engine baut (üblicherweise) ein Nondeterministic Finite Automata
- Matching worst case ist exponentiell
- Z.B.: matche `a@aaaaaaaaaaaaaaaaaaaaaaaaaa`  
`gg.`

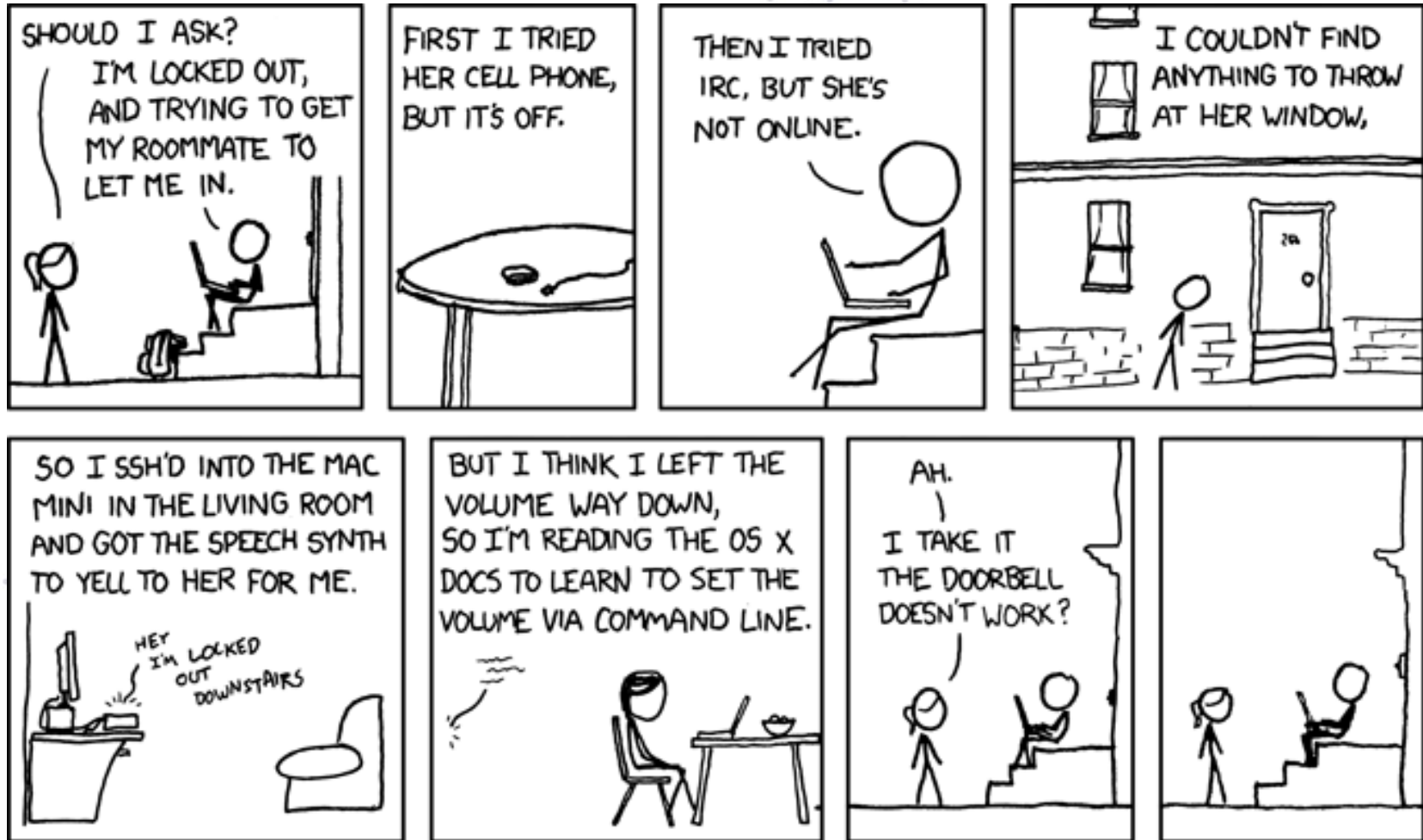
`^[a-zA-Z0-9+]( [\._-]?[a-zA-Z0-9+] ) * @ ([a-zA-Z0-9+]( [\._-]?[a-zA-Z0-9+] ) * ( [ \. ] { 1 } [a-zA-Z0-9]{ 2, } ) + ) $`

# *Regular Expressions*

- Capturing-Groups überprüfen
- Relativ handarbeitslastig
- Libraries anschauen (Validierung von Emailadressen, etc.)
- „Richtige“ Parser verwenden



# Benutzer aussperren



OWASP

Open Web Application  
Security Project

WWW.OWASP.ORG

# *Benutzer aussperren (II)*

- n-mal falsches Passwort für anderen Benutzer angeben → Benutzer gesperrt
- Meist nur auf Zeit (hilft aber nur bedingt)
- Automatisierung mit bekannten/errätzbaren Nutzernamen
- Interfaces für interne Authentifizierungssysteme (AD) aus dem Internet erreichbar
- Entsperrvorgang oft manuell(!)



# *Gegenmaßnahmen*

- Auch mal auf Worst-Case-Performance schauen
- Limitierung von Speicher/CPU-Zeit
- Angreifer dazu zwingen, auch CPU-Zeit aufzuwenden (Hashcash, etc.)
- Performanceprobleme ernstnehmen



# Fragen? Feedback?

Alexander Klink  
Senior Consultant  
Risk Assurance Solutions  
PricewaterhouseCoopers AG WPG

[alexander.klink@de.pwc.com](mailto:alexander.klink@de.pwc.com)  
0160-92392323