

Fuzzing

Fabian Brunner, CNB4

Daniel Leese, CNB4

Netzwerksicherheit

M.Sc., Dipl. Inform (FH) Jakob Pietzka

M.Sc., Dipl. Inform (FH) Michael Schäfer

Hochschule Furtwangen

7. Dezember 2008

Agenda

① Was ist Fuzzing?

Geschichte

Funktionsweise

Was fuzzen?

Ergebnisse

② Bekannte Fälle

MoBB von H.D.Moore

PROTOS

mangleme

③ Fuzzing Demonstration

TAOF

FTPStress

Axman

Shellscripting

④ Quellen

Definition: Fuzzing

- Software-Testing-Technik
- Black-Box-Prinzip
 - Quellcode, Konfiguration unbekannt
- Verwendung von:
 - Speziellen Tools
 - Fuzzing Suiten
 - Selbstgeschriebenes/gescriptetes
- Finden von:
 - Programmfehlern
 - Sicherheitslücken
- Allgemeines Testen auf Robustheit

Geschichte des Fuzzing

- **1988** Professor Barton Miller
 - Operation System Utility Program Reliability - The Fuzz Generator
 - The fuzz program should be used to test various UNIX utilities. These utilities include programs like vi, mail, cc, make, sed, awk, sort, etc. The goal is to first see if the program will break and second to understand what type of input is responsible for the break.

Geschichte des Fuzzing

- **1995** “Fuzz Revisited” Report
 - “fuzz” tool → stdin

Utility	SunOS		HP-UX		AIX		Solaris	SGI	Ultrix	NEXT	GNU	Linux
	90	95	90	95	90	95	95	95	95	95	95	95
#tested	77	80	72	74	49	74	70	60	80	75	45	55
#crash/hang	22	18	24	13	12	15	16	9	17	32	3	5
%	29%	23%	33%	18%	24%	20%	23%	15%	21%	43%	7%	9%

Geschichte des Fuzzing

- **2000** Windows NT Fuzz Report
 - Random → Window Message Queues
 - SendMessage, PostMessage, Random Valid Events
- **Crash:** Visual C++ 6.0, Office 2000, IE5, ...
- Total Percent Failed 85.7%

Geschichte des Fuzzing

- **2006** Mac OS X Fuzz Report
 - **Crash:** Adobe Reader 7, Finder 10.4.3, iTunes 6.0.1, Microsoft Office 11 (2003)

Utility	Linux - 1995	Mac OS- 2006
<i>Only Mac OS:</i>		
expr		●
groff		●
zic		●
zsh		●
<i>Both Mac OS and Linux:</i>		
as		●
ctags	○	
flex	●	
gdb	●	
indent	●	●
nroff		●
ditroff/troff		●
ul	●	●
vi(ex)/vim		●
Number crash/hang:	5	10
Number tested:	54	135
Percentage:	9%	7%

Geschichte des Fuzzing

- Schlüsse:
 - Linux: 9 %
 - Mac OS: 7 %
 - Windows: **85,7 %**

Nicht 100% Fair da:

- Konsolen und GUI Applikationen in einen Topf geworfen
- Große Zeitabstände

Tendenzen jedoch durchaus erkennbar :-)

Funktionsweise

- Völlig zufällige Daten senden:
 - Kann Vielzahl von Fehlern aufdecken, auch solche an die man niemals denken würde.
 - Kann (sehr) lange dauern, es muss der komplette Raum möglicher Eingaben durchlaufen werden.
- Vorherige Analyse der möglichen Eingabedaten:
 - Eingaberaum begrenzen durch vorheriges Studium des “Protokolls”
 - Filter für unsinnige Eingaben können im Voraus berücksichtigt werden
 - Gezielter “Angriff”
 - Fehler können übersehen werden da nicht alle Möglichkeiten durchgegangen werden

Funktionsweise

Die 3 Methoden des Fuzzing:

- Manuell
 - Jede Eingabe wird manuell generiert und abgesendet.
 - Größtmögliche Kontrolle
 - Reaktionen des Programms können direkt einer Eingabe zugeordnet werden
 - Erfordert (viel) Zeit und Kenntnis der Materie
- Automatisch
 - Fuzzer generiert Eingaben nach vordefinierten Regeln selbsttätig und analysiert so gut es geht ob das programm sich noch normal verhält
 - Anomalien beenden das Fuzzing oder werden geloggt
 - Große Eingabemengen können leicht ausprobiert werden, Fehler können jedoch übersehen werden
- Halbautomatisch
 - Mittelweg zwischen den o.g. Methoden
 - Eingaben werden automatisch generiert, die Auswertung übernimmt der Benutzer

Was fuzzen?

- Netzwerkprotokolle / Netzwerkanwendungen (*TAOF*)
 - netcat
 - Z.b. bei FTP: USER <SP> <username> <CRLF>
- Argumente, Umgebungsvariablen, stdin, file-descriptors (*fuzz, cat und pipe*)
 - char** argv, args[]
 - Windows Registry
- APIS (syscalls, library calls) (*Axman*)
 - WM_SENDMESSAGE und co.
- Dateien (*File Fuzzer*)
 - ini, conf
 - doc, Bilder, ...

Ergebnisse des Fuzzings

- **1. DoS durch**
 - Crash des Programms
 - Array out of bounds exception
 - Buffer overflow
 - Stack overflow
 - ...
 - Programm reagiert nicht mehr (Freeze)
 - Überlastung der Hardware
- **2. Code in System einschleusen, z.B. Shellcode**

Fuzzingtools-Übersicht: Spezialisierte Tools

- Netzwerk
 - Spike (Netzwerkprotokolle, Sprache: C)
 - **TAOF** (The Art of Fuzzing, alle TCP/UDP Protokolle)
 - Bed (Netzwerkprotokolle, Sprache: Perl)
 - Smudge (Netzwerkprotokolle)
 - DHCPfuzz (DHCP-Clients)
 - **FTPStress** (FTP-Server)
 - Fuzzball2 (TCP/IP-Header)
 - IRCfuzz (IRC-Clients)
 - Protos (Tool-Sammlung für SIP, SNMP, DNS und andere)
 - SSHredder (Sammlung mutierter SSH-Pakete)
- Browser
 - COM-Bust (Common Object Model)
 - **Axman** (Microsoft ActiveX)
 - CSS-Die (Stylesheet Implementierung)
 - DOM-Hanoi (Data Object Model)
 - mangleme (automated broken HTML generator)

Month of Browser Bug (MoBB)

- H.D. Moore veröffentlichte in nur 30 Tagen 25 Schwachstellen im Internet Explorer
- meist fehlerhafte Funktionen in den ActiveX-Controls
- entdeckt durch Fuzzing-Tools von H.D. Moore

Schwachstelle bei Verarbeitung der vom Session Initiation Protocol (SIP) INVITE messages

- Ermöglichen Buffer Overflow oder Denial of Service Angriffe bei:
- Cisco IP Phone Model 7940/7960
- Ingate Firewall
- Alle Versionen von SIP Express Router bis 0.8.9
- Mozilla Firefox IBM SecureWay V3.2.1 (Solaris und Windows)
- Network Associates PGP Keyserver 7.0
- ...

Schwachstellen in Browsern

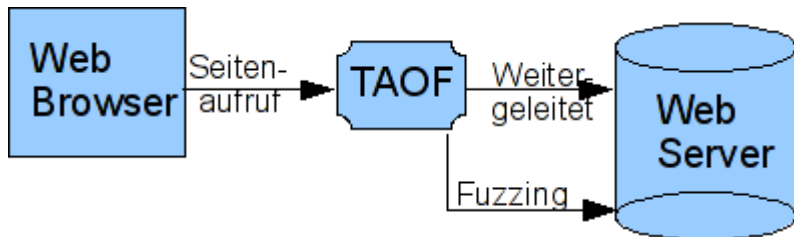
- Microsoft Internet Explorer: u.A. IFrame Bug
- Netscape Navigator / Mozilla Firefox
- Links
- Lynx
- Opera
- Safari
- ...

Fuzzingtools-Übersicht: PROTOS

- Seit 1996 Security Programmers Group der Universität von Oulu in Finland
- Seit 2001 Universitäts-Spin Off **Codenomicon Ltd**
- Sammlung von eigenständigen Fuzzingprogrammen (in Java)
- Unterteilung in einzelne Testsuites (snmp, dns, sip, ...)
- Tausende von Angriffspunkten
- Jeder von PROTOS gefunde Fehler lässt sich (theoretisch) als DoS nutzen, viele sogar zur code injection

TAOF (The art of fuzzing)

- auf Netzwerkprotokolle spezialisierter Fuzzer
- von Rodrigo Marcos

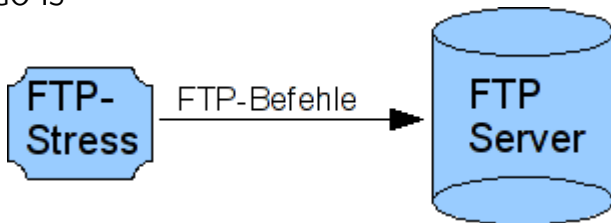


TAOF (The art of fuzzing)

TAOF vorführen

FTPStress

- GUI basierter FTP Fuzzer
- von INFIGO IS

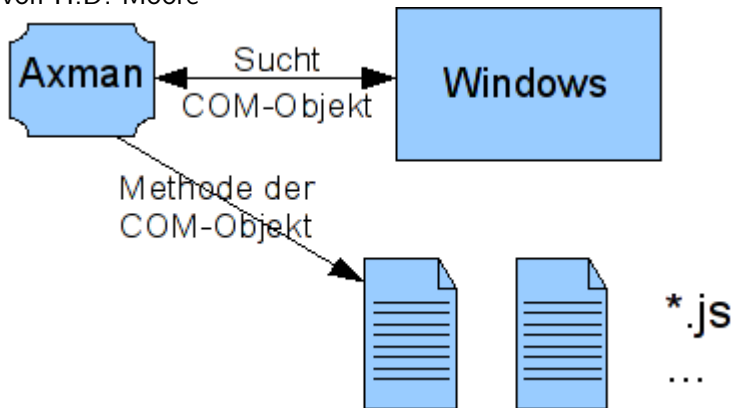


FTPStress (The art of fuzzing)

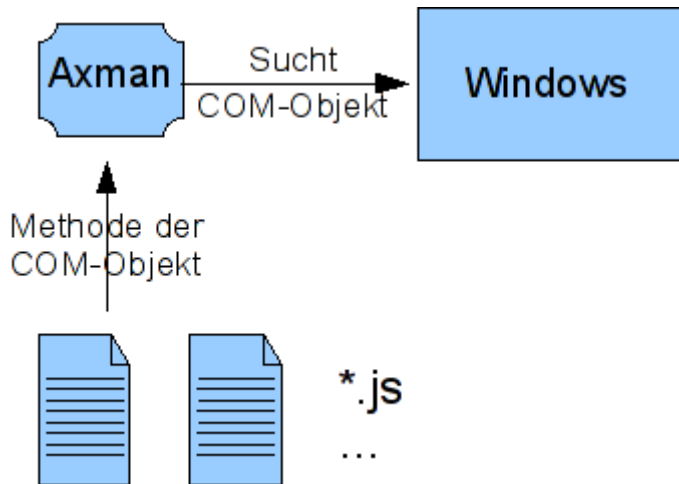
FTPStress vorführen

Axman

- Webbasierter ActiveX Fuzzer
- von H.D. Moore



Axman



Axman

Axman vorführen

Shellscripting

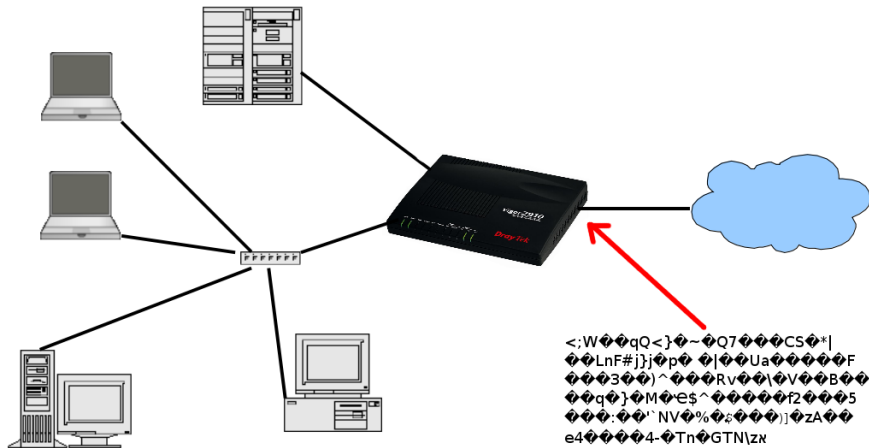
DrayTek Vigor 2910 Dual WAN Security Router Hardwarefirefall fuzzen

- Ca. 150 Euro
- provides a great balance of security, VPN, routing Content filtering helps protect against inappropriate web usage or time wasting
- Bla, bla ...
- “Eine Appliance löst alle Probleme”



Praxisbeispiel: HW-Firewall Fuzzen

Versuchsaufbau:



Shellscripting

DrayTek Vigor 2910 Dual WAN Security Router Features:

Vigor 2910 Router Specification

- * Combination Ethernet router, VPN Device, **Firewall** and Load-Balancer
- * Primary Ethernet WAN Interface
- * Selectable secondary WAN Interface
- * Load Balancing across both WAN ports with automatic or user-defined policies

- ...

- * Internet Firewall facilities featuring :
 - o Automatic Keep-state facility for **tracking packets and denying unsolicited ...**
 - o **Selectable DoS/DDoS protection**
 - o IP Address anti-spoofing
 - o User-configurable packet-filtering
 - o NAT/PAT for **Automatic LAN/WAN Mapping and Security**
 - o NAT Port Redirection with automatic internal ranging
 - o **Real Time Data Flow Monitor**, with **instant block** (cut of any user immediately!)

Shellscripting

Vigor vorführen

Shellscripting

easyHacking.sh von Fabian Brunner und Daniel Leese

```
xterm
#!/bin/bash

# endless loop
while [ 1 ]
do
    #cat /dev/urandom | netcat 192.168.0.1 21 & # FTP
    #does nothing

    #cat /dev/urandom | netcat 192.168.0.1 22 & # SSH
    #sure crash ~10sec

    #cat /dev/urandom | netcat 192.168.0.1 23 & # Telnet
    #no crash, but intresting behaviour

    cat /dev/urandom | netcat 192.168.0.1 80 & # HTTP
    #sure crash ~10sec

    #cat /dev/urandom | netcat 192.168.0.1 443 & # HTTPS
    #does nothing
done

"easyHacking.sh" 20L, 430C                20,1                Alles
```

Shellscripting

Folgen:

- Verbindungen gestört
- VPN unterbrochen
- Eventuell neue IP-Adresse
 - Websitzungen
 - SSH,FTP, ...
 - Online Banking
 - ...
- Exploit möglich?!?

Shellscripting

Ausblick:

- Rausfinden welche Zeichenfolge(n) genau Absturz verursachen
- Auch mit geringer Datenrate (Internet) möglich?
- Passwort setzbar!?

```
xterm

Password: *****
Password: ****
Password: *****
Bad Password, Bye-Bye :-(\

System administrator is connecting from 192.168.1.10
Reject the connection request !!!
0000

Password: *****

*** WARNING *****
* System has no password. *
* Please set password, using "sys passwd" commands. *
*****

Type ? for command help

> }0sT?^D
^C^C
```

Quellen

- 22C3, Fuzzing - Breaking software in an automated fashion (<http://events.ccc.de/congress/2005/fahrplan/events/537.de.html>)
- 2006 Mac OS X Fuzz Report, Barton Miller, University of Wisconsin-Madison
- 2000 Windows NT Fuzz Report, Barton Miller, University of Wisconsin-Madison
- B.P. Miller, L. Fredriksen, and B. So, “An Empirical Study of the Reliability of UNIX Utilities”, Communications of the ACM 33
- <http://www.heise.de/security/Die-Axt-im-Walde-/artikel/76512/0>
- <http://theartoffuzzing.com/>

Ende

Noch Fragen?