DIMVA 2004

A Honeynet within the German Research Network — Experiences and Results

Helmut Reiser
Ludwig-Maximilian University Munich

Gereon Volker
Technical University Munich

helmut.reiser@nm.ifi.lmu.de ,
gereon.volker@stud.tu-muenchen.de

MNM TEAM

Introduction

- **Honeypot**: single system to be
  - Probed, attacked and compromised (hacked)
  - By (unfriendly) attackers
- **Honeynet**:
  - A network of honeypots
  - Copy of the “real world” network
  - Not used in regular business

  ⇨all (network) traffic caused by attackers
- Why honeynets and honeypots?
  - Learn tactics, motives, tools and techniques of attackers
  - Learn about (new) vulnerabilities
  - Slow down an attack
- Honeynet within the German Research Network (DFN)
  - Set up at the Leibniz Supercomputing Center (LRZ)
  - Operated between July 15th and September 12th 2003

Honeynet – Design

Honeynet must fulfill three requirements / tasks

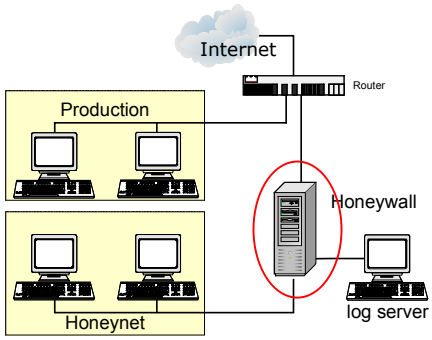
1. Data capture
   - Recording of all traffic
   - Recording of all actions
   - Inbound and outbound
2. Data control
   - Prevention of attacks sourced in the honeynet

   ⇨ No harm to other (foreign) systems
3. Data analysis
   - Efficient analysis of captured data
   - Extract relevant data out of “noise”
   - Identifying techniques used in attacks
   - Find source of attack

Data Capture Architecture: Honeywall

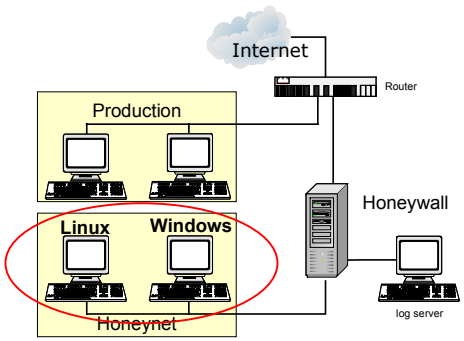
- Invisible for the attacker

  ⇨ Acts like a bridge (from attackers point of view)
  - No TTL decrement
  - No routing
  - No spanning tree protocol
- Efficient capturing, analyzing, filtering and controlling tool (for the operator)
  - All data passing can be captured (`tcpdump`)
  - Extended firewall with IDS to detect known attacks
    - Alarming
    - Reduction of data
    - “Noise“ filtering



Data Capture Architecture: Honeypots

- Windows 2000 and SuSE Linux 8.3
- Dump attackers keystrokes
  - `ComLog` (Windows)
  - `Sebek` (Linux)
    - Forwards key strokes to log server
    - Modified rootkit (hardly detectable)
    - Able to capture secure shell (ssh) keystrokes
- Forwarding local logfiles to log server (modification prevention)
  - Windows `Eventlog` to `syslog` (log server)
  - Linux: forward `syslog` to log server
  - Camouflage: second “hidden” log daemon



Data Control: Honeywall

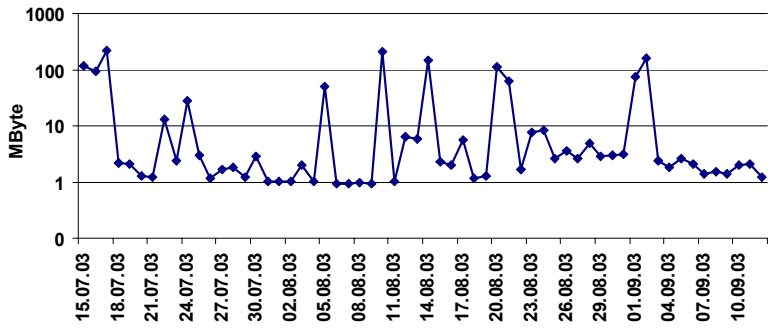
- Extended firewall with Intrusion Detection System (IDS)
  - Firewall forwards **outbound** traffic to IDS (`snort_inline`)
  - IDS drops known attacks (signature based)
  - Even “automatic“ attacks like worms could not attack foreign hosts
- What about “unknown” attacks?
- Firewall restricts number of outgoing connections
  - 15 connections per day
  - Asymmetry (could be suspicious for attacker)
- Alarming of the operator
  - Monitoring firewall logs with `swatch`
  - New entry, swatch sends an email
  - SMS messages for outgoing connections
  - Grouping mechanisms and message rate limited

Data analysis

- Logfile analysis: finding the “valuable” packets
  - Coping with a huge amount of data (up to 200 MB per day)
  - `snort` logs with `ACID`
  - Firewall logs with `iptables_log`
  - Charting, summarizing, efficient query mechanisms
- Binary packet analysis: investigate the interesting packets
  - Inbound and outbound traffic dumped with `tcpdump`
  - `Ethereal` (Unix) and `Packetyzer` (Windows)
  - Decoding of several protocols; searching within the data
- Investigating source of attack: finding hostname, subnet or domain
  - Reverse lookup for the hostname
  - `traceroute` and `visualroute` finding “geographical” location
  - `P0f` for the identification of attackers operating system (passive fingerprinting)

Results: General Observations and Traffic

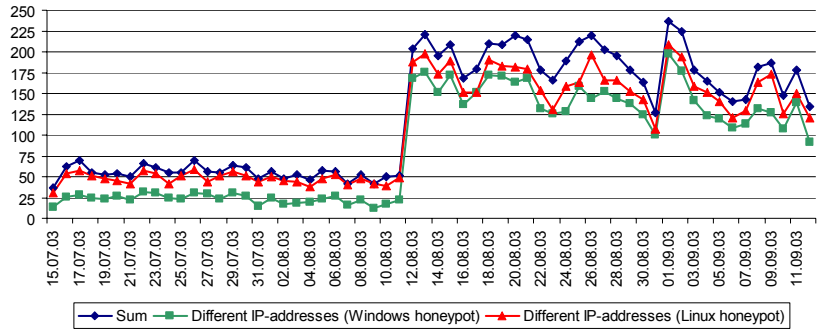
- General Observations
  - At no time existence of the new subnet was propagated
  - Honeynet got online 8:55 am (GMT+1) on July 15th
  - First successful attack two minutes later (CodeRed2 on MS IIS)
- Honeynet Traffic [MByte/day]



Results: Number of Sources

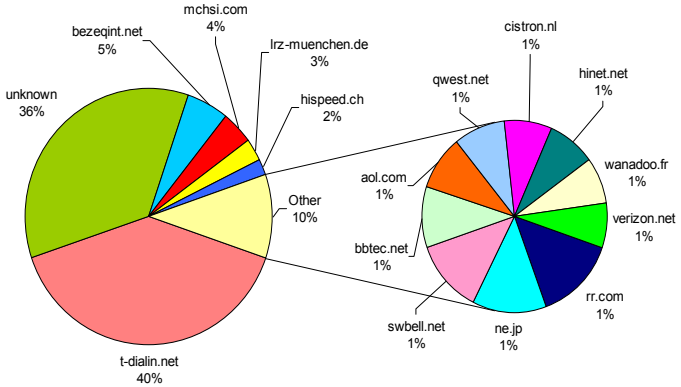
- Number of different IP-addresses per day

  Remark:
  - Filter rules for NetBIOS normally activated at DFN access router
  - These filters have been disabled for the honeynet of August 12th



Results: Sources of Attacks

- DNS lookup promptly triggered on the firewall
  - 40 % Digital subscriber line customers of „Deutsche Telekom“
  - 36 % Unknown:
    - Reverse lookup disabled
    - IP-Spoofing



Results: Kind of Attacks

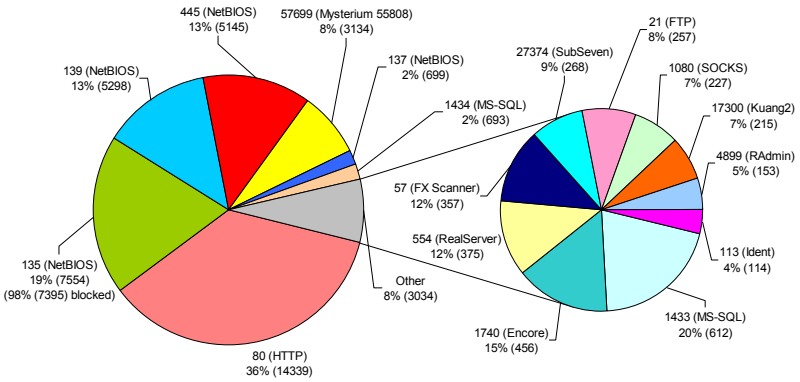
- Web Attacks
  - Mostly against Microsoft IIS
  - Plenty of well known vulnerabilities
- Worms
  - Blaster appeared on August 11th 10:56 pm; variants on 20th
  - Source: client within the Munich Research Network
  - `Snort_inline` prevented further dissemination
- (Distributed) Denial of Service (DoS and DDos)
  - DNS Servers of different US providers probably became victims
  - Addresses of honeypots have been used spoofing the source
  - Victims replied to honeypots with SYN/ACK Packets

Results: Kind of Attacks (cont.)

- "Mysterium 55808"
  - Packets with large window size 55808
  - Destination port 57669
  - No payload data
  - Intrusec and ISS called causing trojan "Stumbler"
    - Maybe for scanning purposes
- Noise
  - Well known backdoor or trojan ports, e.g.:
    - Skydance (Port 4000)
    - RAdmin (Port 4899)
    - ……
  - Proxy Ports (e.g. 8080) or SOCKS (1080)
  - ……

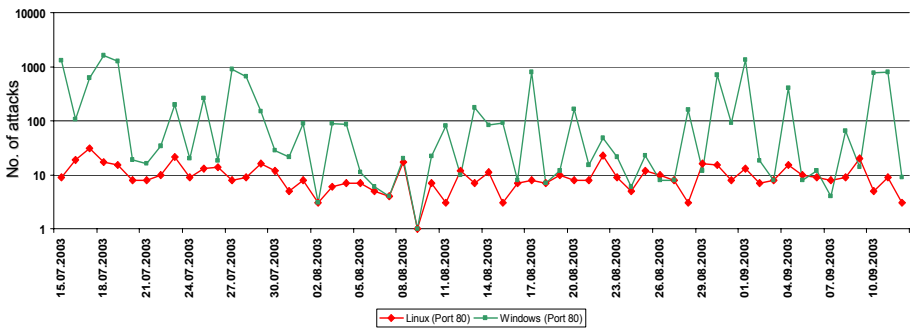
Results: Distribution of Attacks

- Attacked ports (services) and frequency
  - 47 % against Windows NetBIOS
  - 36 % against Web Servers (95 % against IIS)



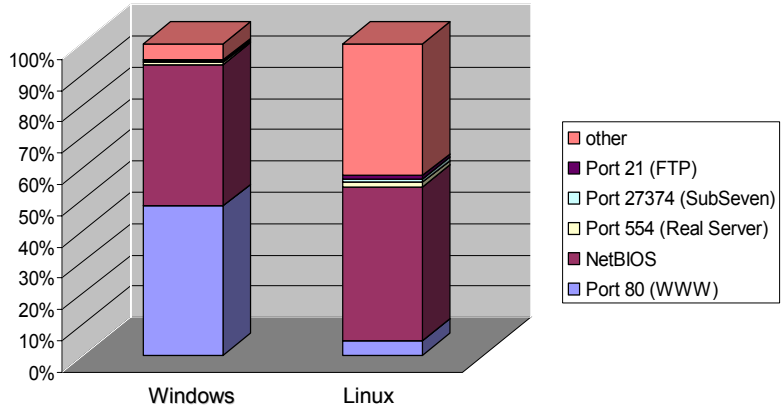
Results: Goals of Attacks

- 69 % of all attacks hit Windows
- Web server attacks: comparison Windows / Linux
  95 % against IIS



Results: Attacks Regarding OS

- Well known attacks against Windows
- More not well known (other) attacks against Linux



Lessons Learned

- “Unknown” systems are extremely fast under attack
- “Unfortunately” no “real” or “clever” hostile take over
- Windows was the favorite target (69% of all attacks; 95% of web server attacks)
- Most of the attackers are script-kiddies
- Data Control works: no harm to foreign systems, no distribution of worms
- 90 / 10 Rule:
  90% of the attacks can be prevented with 10% effort
  - Implement a firewall
  - Block services which are a chinch to exploit
  - Efficient patch management
  - Use saved time to spend more time for the lacking 10%