

# User Mode Linux as a Honeypot



Seminar User-Mode Linux

Christian Delis

17.02.2006

WS 05/06

[cdelis@uni-koblenz.de](mailto:cdelis@uni-koblenz.de)

# Agenda

- Motivation
- Einführung
- Anforderungen
- Tools
- Anwendung in UML
- Praxis

# Motivation

Defensive:

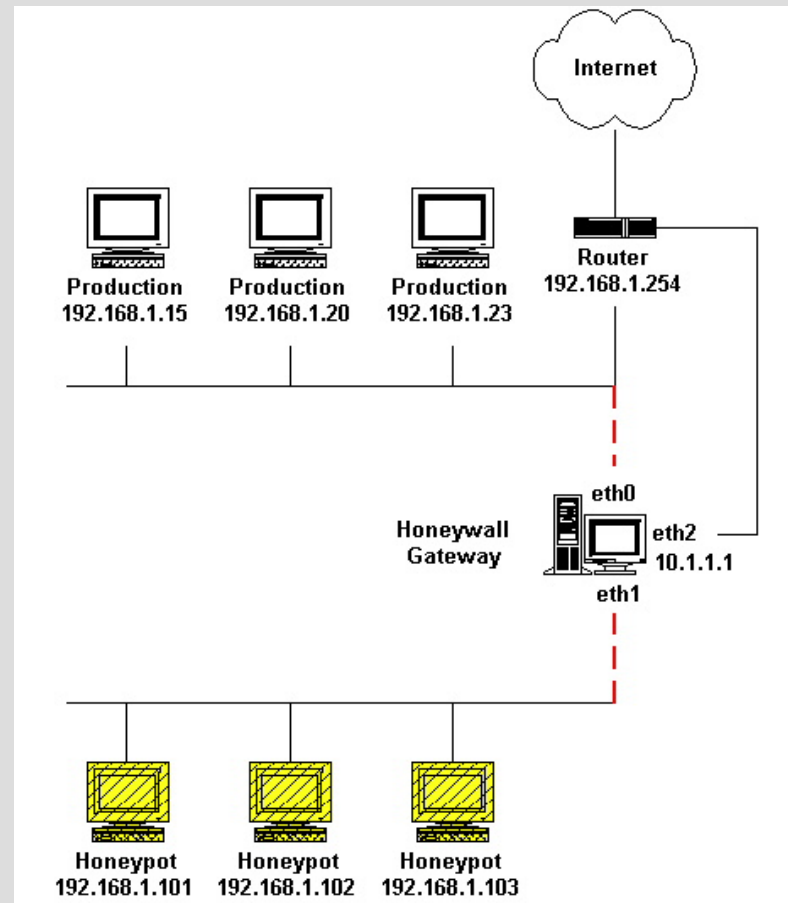
- Verschlüsselung
- Firewalls
- Intrusion Detection Systems

# Was ist ein Honeypot?

„A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.“

Lance Spitzner, Gründer der HoneyNet Research Group

- Honeypot
- Honeyynet
- Honeywall



# Warum ein extra System dafür?

- Kein Produktivsystem.

# Warum ein extra System dafür?

- Kein Produktivsystem.
- Normale Nutzung oder Angriff?

# Warum ein extra System dafür?

- Kein Produktivsystem.
- Normale Nutzung oder Angriff?
- Normalen Netzbenutzer unbekannt

# Warum ein extra System dafür?

- Kein Produktivsystem.
- Normale Nutzung oder Angriff
- Normalen Netzbenutzer unbekannt
- Kein Filtern von Logfiles
- Eingehende / Ausgehende Verbindungen

# Honeypots - Wozu?

Informationen:

- Werkzeuge
- Trends
- Vorgehensweisen
- Ablenkung

# Interaktionsgrad

- Erlaubte Interaktion
- Interaktion --> Lernen --> Nutzen
- Interaktion vs. Risiko

# Niedriger Interaktionsgrad

- Dienste werden emuliert
- Software ist leicht zu installieren
- Trendanalyse
- Keine Informationen über Tools und Taktiken

# Mittlerer Interaktionsgrad

- Sicherheitslücken werden simuliert.
- Automatisierte Scans
- Sammeln von Schadprogrammen
- Analyse von Schadroutinen

# Hoher Interaktionsgrad

- Reale Hardware, Betriebssysteme
- Viel Lernen
- Hohes Risiko

# Anforderungen - Honeyypot

- Architektur
- Hoch kontrollierte Umgebung.
- Anforderungen:
  - Datenkontrolle
  - Datenerfassung

# Datenkontrolle

- Kontrolle über den Datenfluss
- Beherrschung der Aktivität
- Mehrschichtig
- Risiko Minimierung

# Datenkontrolle

- Bandbreitenbeschränkung
- Anzahl ausgehender Verbindungen

# Datenerfassung

- Überwachen und Mitloggen aller Aktivitäten
- Soviel Daten wie möglich sammeln, ohne das Hacker es mitbekommt.
- Problem Verschlüsselung

# Datenerfassung

- Daten nicht lokal speichern
- Manipulation der Daten
- Was?
  - Netzwerkverkehr
  - Verbindungen
  - Systemaktivität

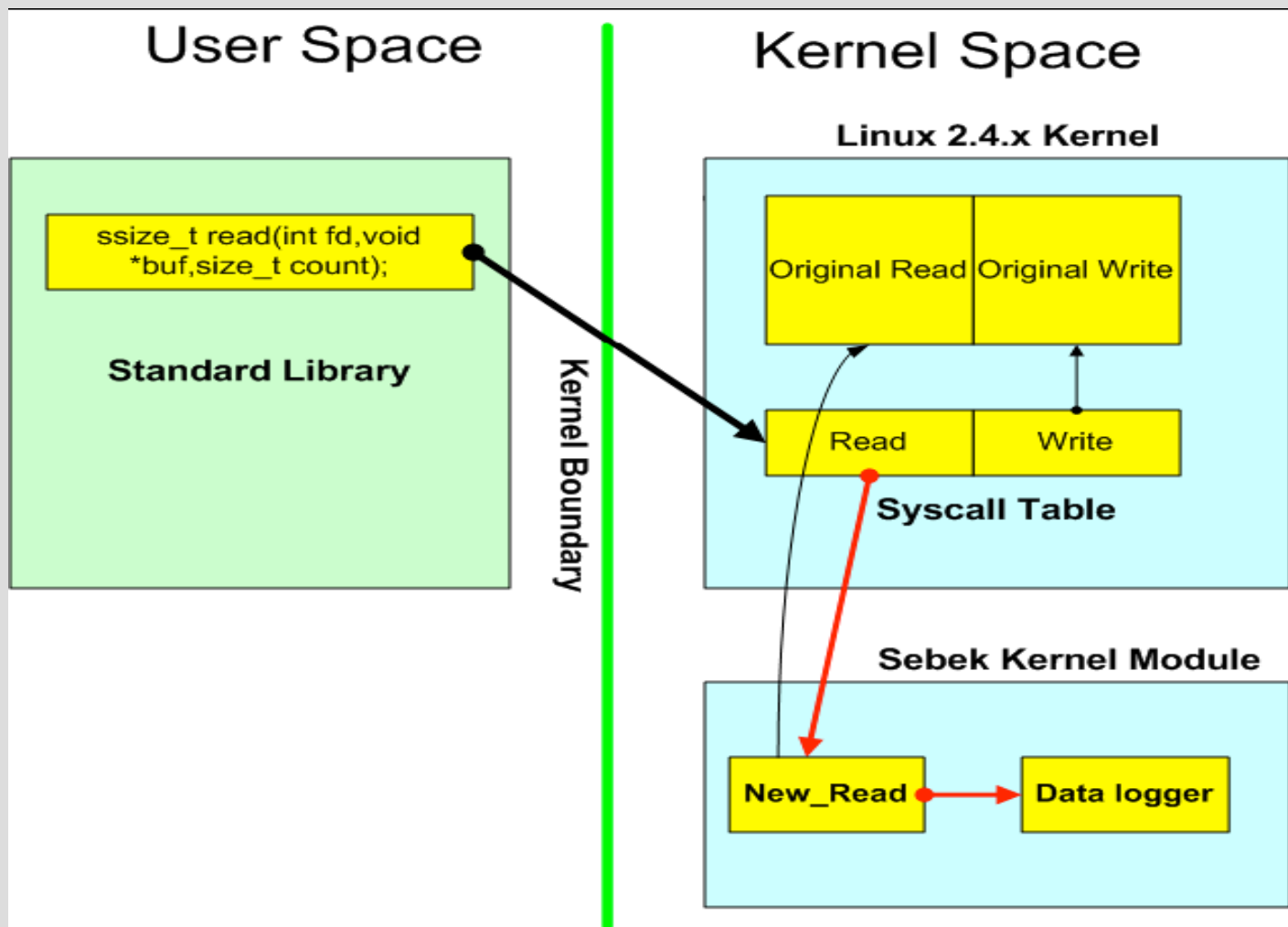
# Risiko

- Nutzung des Honeynet für Attacken.
- Gefahr der Erkennung
- Ausschalten der Datenkontrolle/erfassung
- Missbrauch für illegale Aktivitäten

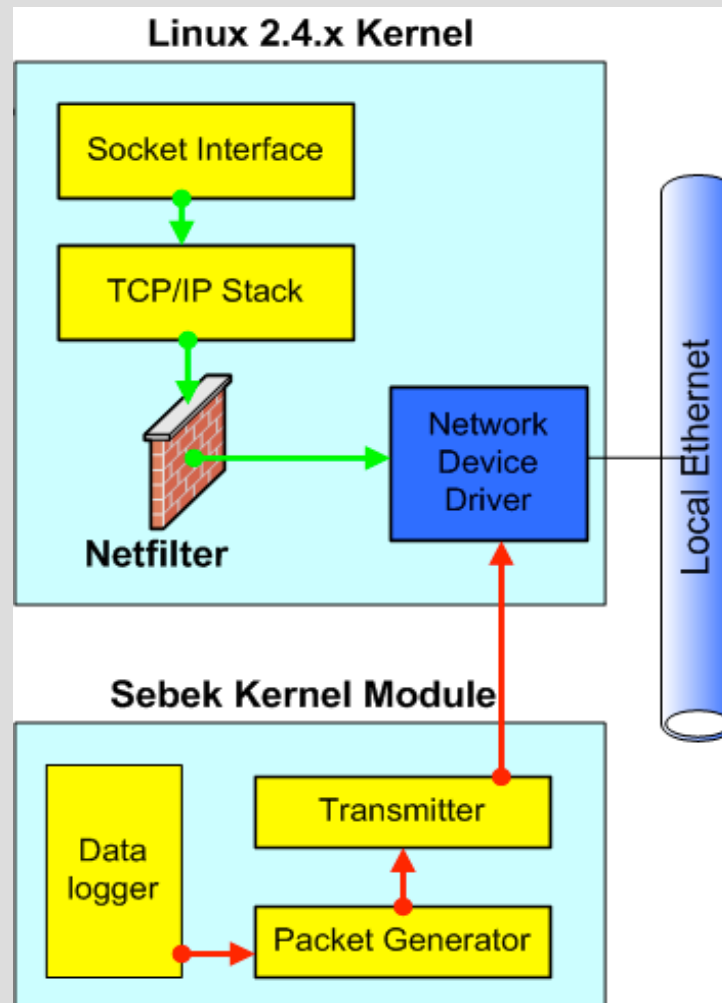
# Tools

- honeyd:  
Emuliert gesamte Netzwerkstrukturen und Computer.
- mwcollect:  
Sammeln von „Malware“
- Sebek:  
Überwacht vom Kernelspace aus alle Programme des Userspace.

# Sebek



# Sebek



# Virtuelle Honeynets

- Vorteile:
  - Administration
  - Kosten
  - Plug and Catch
- Nachteile:
  - x86 Architektur (z. B. Cisco IOS )
  - Single Point of Failure
  - Schwieriger zu tarnen

# Anwendung in UML

- Vorteile:
  - Open Source
  - Kein X Server nötig
  - #( Virtuelle Netze) > 1
  - Eingebautes Keylogging
- Nachteile:
  - Linux only
  - Kein Support
  - Keine GUI

# UML Honeypot Support

## tty logging

- tty driver patch
  - Überwachung von Terminaleingaben
  - Kein Netzwerkverkehr
- Kernelconfig:
    - CONFIG\_TTY\_LOG=y
    - Character Devices -> Enable tty logging

# UML Honeypot Support

## tty logging

- Jede Session wird in einer eigenen Datei gespeichert.
  - Uhrzeit bestimmt Dateinamen
  - Default im lokalen Verzeichnis
    - `tty_log_dir=<dir>`

# UML Honeypot Support

## tty logging

- Besser in einer Datei speichern
  - Binäres logging format
    - `tty_log_fd=3 3>tty_log_file`

# UML Honeypot Support

## tty logging

- Parser
  - `perl tty_log.pl <logfile>`
- Replay
  - `perl playlog.pl <logfile> [tty-id]`

# UML Honeypot Support

## honeypot proc filesystem – hppfs

```
host1:~# cat /proc/cpuinfo
processor          : 0
vendor_id        : User Mode Linux
model name       : UML
mode             : tt
host             : Linux sonne 2.6.15-gentoo-r1 #4 Fri Feb 10
                 21:48:11 CET 2006 i686
bogomips         : 4443.34
```

# UML Honeypot Support

## honeypot proc filesystem – hppfs

- **/proc faken**
- **/proc Einträge**
  - hinzufügen
  - entfernen
  - modifizieren

# UML Honeypot Support

## honeypot proc filesystem – hppfs

- hppfs liest die proc Hierarchie im Arbeitsverzeichnis des UML Kernels
  - Kein Eintrag – Original Eintrag
  - Datei oder Link – Überschreibt UML Eintrag
  - Verzeichnis
    - remove – Eintrag wird entfernt
    - Unix Socket r – Eintrag wird dynamisch generiert
    - Unix Socket rw – Filtern von UML Einträgen

# UML Honeypot Support

## honeypot proc filesystem – hppfs

```
honeypot# mount none /proc -t hppfs
```

```
host% mkdir proc
```

```
host% echo 'Hallo Welt' > proc/cmdline
```

```
honeypot# cat /proc/cmdline
```

```
Hallo Welt
```

# UML Honeypot Support

## honeypot proc filesystem – hppfs

- uml\_utilities: honeypot/honeypot.pl

```
host# perl honeypot.pl /home/chris/uml/2.4/ &
```

# UML Honeypot Support

## honeypot proc filesystem – hppfs

- **Auszug aus honeypot.pl:**

```
"devices" => remove_line("ubd"),  
"uptime" => proc("uptime"),  
"exitcode" => "remove",  
"filesystems" => remove_line("hppfs"),  
"version" => proc("version") )
```

# UML Honeypot Support

## skas mode

- Bisher tt Mode
- UML Prozesse laufen im gleich Adressraum wie UML Kernel
- UML Prozesse können auf Kernel und seine Daten zugreifen

# UML Honeypot Support

## skas mode

- Kritisch! Sie können sogar darauf schreiben.
- Aus dem UML System ausbrechen.
- Root Rechte

# UML Honeypot Support

## skas mode

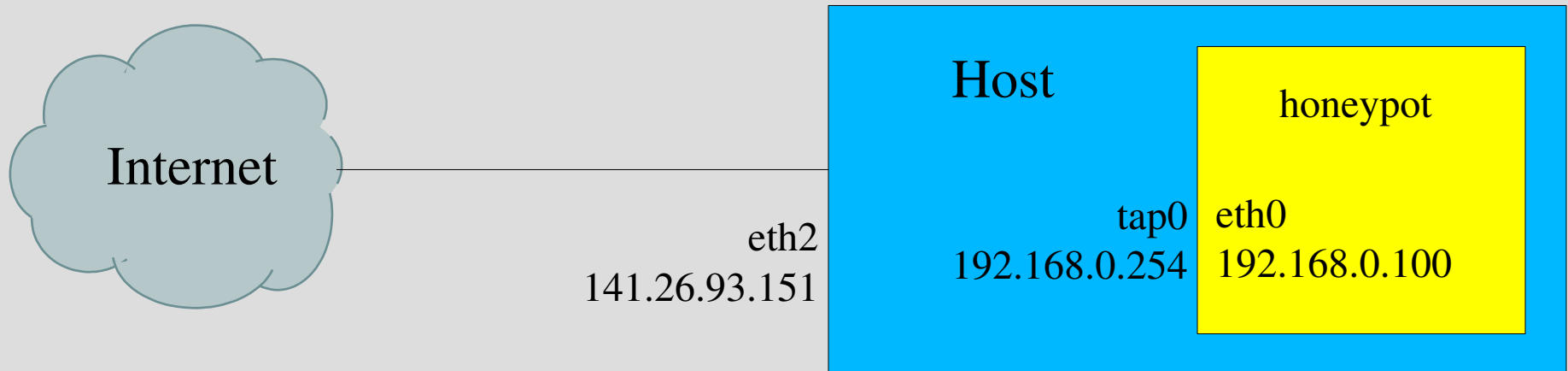
- separate kernel address space
- UML Kernel jetzt unsichtbar.
- Host Patch erforderlich
- Bootmeldung:  
Checking for the skas3 patch in the host...found  
Checking for /proc/mm...found

# Honeypot starten

- `host# perl honeypot.pl /home/chris/uml/2.4 &`
- `host# ./linux ubd0=Debian-3.0r0.ext2 umid=debian1  
eth0=tuntap,,,192.168.0.254 mem=128M  
tty_log_dir=log tty_log_fd=3 3> log/tty.log`

# Netzwerk einrichten

- `honeypot# ifconfig eth0 192.168.0.100`
- `honeypot# route add default gw 192.168.0.254`
- `nat: 192.168.0.100 <--> 141.26.94.253`



# Datenkontrolle

- Eingehende / Ausgehende Verbindungen
- rc.firewall Skript von honeynet.org
- Zählt In / Out Verbindungen
- Blockt Out Verbindungen
- Unterstützung für Sebek und Snort-Inline

# Datenerfassung

- Snort
- Erkennung von Angriffen und Einbrüchen.
- Netzwerkverkehr
- Musteranalyse nach bekannten Angriffsmustern

# Datenerfassung

- Mithören an tap0
- host# `/etc/init.d/snort start`
- `/var/log/snort/`
- `-D -u snort -i $IFACE -l $LOGDIR -c $CONF`
- host# `snort -vdr snort.log.*`

# Quellen

- [www.honeynet.org](http://www.honeynet.org) Whitepapers
- [www.user-mode-linux.org/](http://www.user-mode-linux.org/)