Rootkits & Honeypots

System Software – MSSF – DCU - 17/12/2009

Angel Alonso angel.alonso@mail.dcu.ie

CISSP, CISA, CISM, CCNA | SANS / GIAC: GCIH, GCIA, GCFA, GCFW, GSNA

Source: http://caballerozp.blogspot.com

Agenda

1. Rootkits
2. Real incident involving Rootkits
3. Honeypots
4. Sebek: a real honeypot

Lir lules

- During s of the Linux kernel's f
- What ha scalls through loading a

Source: http://www.filmsy.com

Typical scenario

1. Reconnaissance of the target
2. Access to the target somehow
3. Privileges escalation
4. Hide, delete evidences, assurance access == Rootkits!!!!

Rootkit Definition

- **NSA: *"A hacker security tool that captures passwords and message traffic to and from a computer. A collection of tools that allows a hacker to provide a backdoor into a system, collect information on other systems on the network, mask the fact that the system is compromised, and much more. Rootkit is a classic example of Trojan Horse software. Rootkit is available for a wide range of operating systems. "***
- **A tool or set of tools used by an intruder to hide itself masking the fact that the system has been compromise and to keep or reobtain administrator-level (privileged)access inside a system.**
- ***Hide activity, Provide unauthorized access, Eavesdropping tools, Hacking tools, Systems logs cleaners, etc.***

Taxonomy I

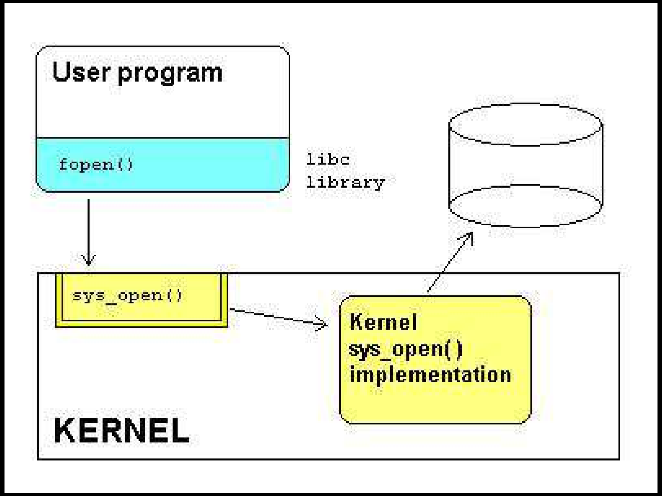
- User mode == Trojan Horse backdoor
  - Modification of some system binaries to hide FILES, PROCESSES, SNIFFING, CONNECTIONS, TASKS, LOGINS, LOGS, etc. (ls, ps, ifconfig, netstat, who, cron, syslog, etc
    - Many binaries to be modified and very dependent on the OS
    - Can be detected easily (checksums) ✖
- Kernel mode
  - Loadable Kernel Modules (LKM) (device drivers in Windows)
    - Modify the system call by loading a module. It is possible to modify or infect a 'trusted' module as well.

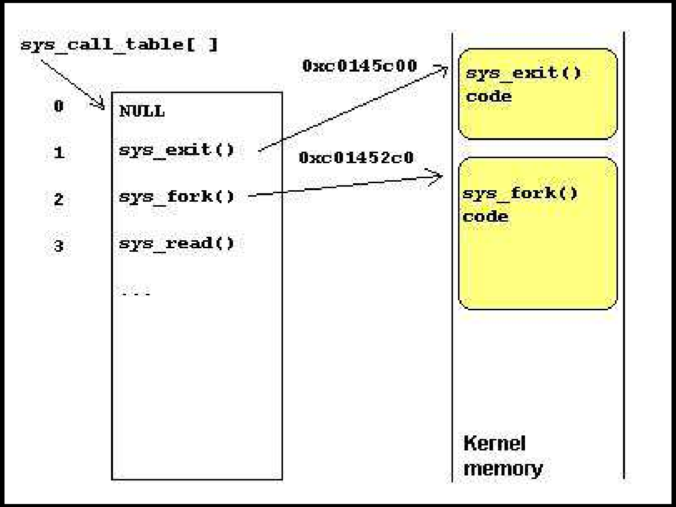
Taxonomy II

- Kernel mode
  - Patch the running kernel: modify the kernel image running in memory through /*dev*/*kmem*
  - Patch the image /*boot*/*vmlinuz*
  - Create a fraudulent VFS: run a exact copy of the real system in a virtual environment (UML, VMware..). This has not been implemented.
  - Run programs in kernel mode: User program can run in the kernel space hence is able to modify the kernel structures and memory. (Kernel Mode Linux project)

Syscall Implementation

- Kernel Space is defined in GDT (Global Descriptor table) and mapped to every process.
- Syscalls through INT 0x80. EAX == number of the syscall (from sys_call_table[]) EBX, ECX, EDX, ESI, EDI for parameters





- Source: http://www.giac.org/certified_professionals/practicals/gcux/0243.php

Syscalls Replacement

- The arguments issued to the system call must be obtained from the user space. -> Access to user space memory
- Declare of *extern void* sys_call_table[]*
- Examples of functions:
  - Hide file contents: intercept sys_open() and block if some pattern in the filename
  - Hide directories: sys_chdir(), sys_mkdir()
  - Hide network connections: sys_read() to */proc/net/tcp* and */proc/net/udp*
  - Hide processes: sys_getends() to /proc

Example of module

The module is loaded through “insmod module.o” and becomes part of the kernel.

```
int init_module(void) {

official_example_call = sys_call_table[ SYS_example_call ];

sys_call_table[SYS_example_call ] = (void *) hacked_example_call;

}

void cleanup_module(void) {

sys_call_table[SYS_example_call ] = (void *) official_example_call ;

}
```

Detection Linux Rootkits

- File Integrity / HIDS (Osiris, Tripware, AIDE). RPM can check the integrity of the binaries.
- Some ideas:
  - /proc /cmdline, /proc/modules, /proc/kcore
  - Big size files , files without user/group, files with "," as name, MAC times.
  - Binary analysis: strace (user mode rootkit)
  - Network layer: external nmap (port knocking could be an issue!!), promiscuous mode (ip link) or /var/log/messages
- Tools: Chrookit, Rootkithunter, Kstat, Module hunter, Unhide

Prevention

- Hardening the SO: patches, services, accounts, compilers, modutils, etc
  - Use some security baseline and tools: CIS, bastille, LIDS, Tiger.
  - Add security tools: grsecurity, SELinux, etc
- Systrace: capture all the systemcalls (IPS)
- Locking LKM: baseline of LKM
- Security VS usability: disable kernel modules? ?

Examples of other rootkits

- **Sony BMG CD copy protection scandal** **http://en.wikipedia.org/wiki/Sony_BMG_CD_copy_protection_scandal**
  - **Rookit that modifies the way Windows plays CDs. Besides, it creates a vulnerability (malware). It uses software with GNU license.**

- Rootkits headed for BIOS http://www.securityfocus.com/news/11372

- Cisco IOS rootkit: http://eusecwest.com/sebastian-muniz-da-ios-rootkit.html

- MacOSX rootkits in BlackHat: http://www.blackhat.com/presentations/bh-usa-09/DAIZOVI/BHUSA09-AdvOSXRootkits-PAPER.pdf

- SSM rootkits (System Memory Management) in Intel: http://www.eecs.ucf.edu/~czou/research/SMM-Rootkits-Securecom08.pdf

- Hardware virtualization rootkits: http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf

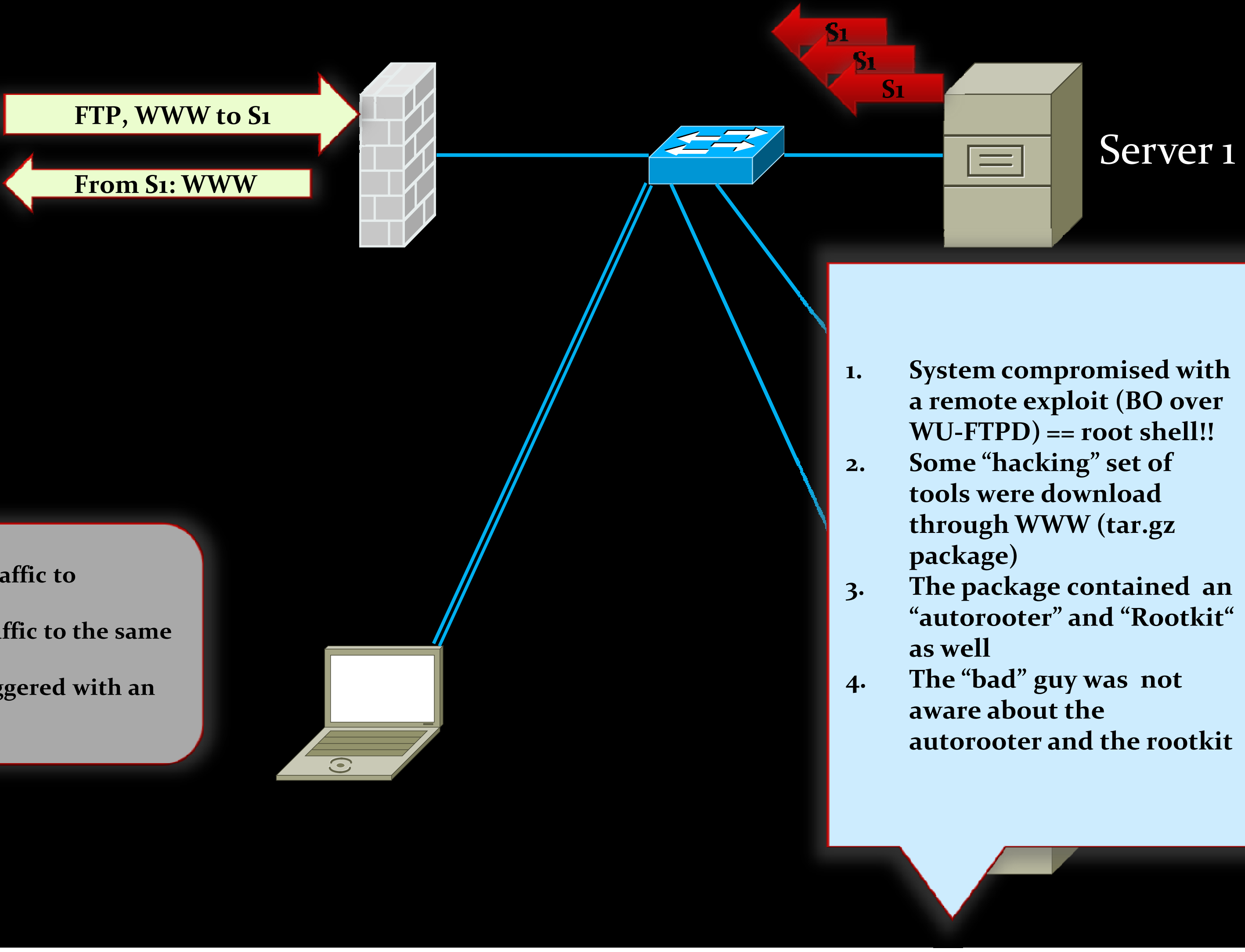
- Oracle rootkits:

http://www.red-data-base-security.com/wp/oracle_rootkits_2.0.pdf

Bibliography

- Linux kernel rootkits: protecting the system's "ring-zero" http://www.giac.org/certified_professionals/practicals/gcux/0243.php Raul Siles May, 2004
- Hiding processes (understanding the linux scheduler): http://www.phrack.org/show.php?p=63&a=18 Rainer Wichmann 2002
- Finding hidden kernel modules (the extreme way) http://www.phrack.org/phrack/61/p61-0x03_Linenoise.txt madsys August, 2003
- The Implementation of Passive Covert Channels in the Linux Kernel http://invisiblethings.org/papers/passive-covert-channels-linux.pdf Joanna Rutkowska October, 2004
- Analysis of the torn rootkit http://www.securityfocus.com/infocus/1230 Miller November, 2000
- Linux Kernel Rootkits http://la-samhna.de/library/rootkits/index.html Rainer Wichmann 2002

Real Scenario



1. System compromised with a remote exploit (BO over WU-FTPD) == root shell!!
2. Some "hacking" set of tools were download through WWW (tar.gz package)
3. The package contained an "autorooter" and "Rootkit" as well
4. The "bad" guy was not aware about the autorooter and the rootkit

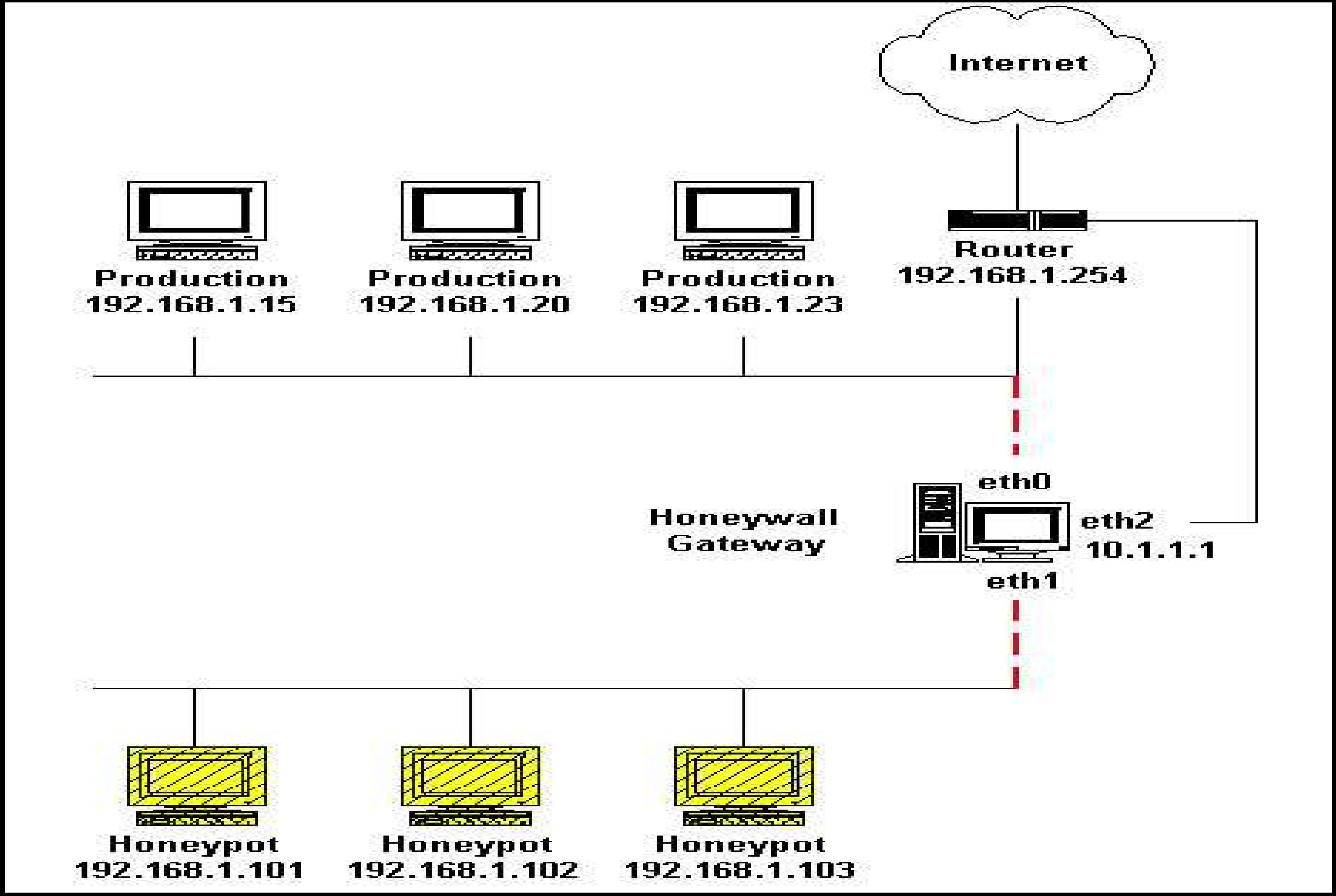
1. S1: "weird" traffic to Internet
2. S1: "weird traffic to the same VLAN"
3. S1: Alerts triggered with an IDS

Honeypots

“If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.” *The Art of the War - Sun Tzu*

An example of a **honeypot** is a system used to simulate one or more network services that you designate on your computer's ports. An attacker assumes you're running vulnerable services that can be used to break into the machine. This kind of **honeypot** can be used to log access attempts to those ports including the attacker's keystrokes. This could give you advanced warning of a more concerted attack.” Source: http://www.honeypots.net

Topology Example



Source: http://www.honeynet.gr

Features of Honeypots

- Historically: Intrusion Detection System (IDS)
- No production value
- Honeypot VS Intrusion Detection System (no FP)
- Currently, no sufficient taxonomy in this area
- Useful for *0-days* attacks
- Work in encryption environments
- Risk of take over ✗

Taxonomy I

- Interaction level
  - Low : Limited interaction. i.e : SSHD that doesn't give real access
  - High: fully simulation of the service.
- Data capturing
  - Event: change in the state
  - Attack: threatening the security policy
  - Intrusion: break the security policy
- Containment:
  - Block : the attack is block and never reach the target
  - Defuse: the attack reach the target but is modified
  - Slow down: the attacker is slowed down to limit the spreading malicious activity

Source: http://www.mcs.vuw.ac.nz/comp/Publications/archive/CS-TR-06/CS-TR-06-12.pdf

Taxonomy II

- Distribution appearance:
  - Distributed: multiple systems.
  - Stand-alone: single system
- Communication Interface
  - Network Interface: the honeypot can be directly communicated via a network interface
  - Not network hardware interface: USB Keys, CDROM..
  - Software: API
- Multi tier: server or client

Some examples

- Google Hack Honeypot: logs the attempts of exploit through Google search. i.e: version of a vulnerable CMS
- Honeyclient: monitors behaviour of IE while browsing to suspicious URLs.
- Honeyd: several servers with different services (SMTP, FTP, HTTP..)
- Honeynet: real system reachable through the Honeywall Gateway (IDS, Iptables, logging, etc)
- Sebek: monitors all the connections and the commands launched by the intruder. Captured tool.

Sebek

- Kernel based data captured tool: intercepting syscalls at kernel level. (LKM, kernel driver in windows, kernel patch BSD)
- Based on a server-client architecture: honeywall and sebek. Covert channel with UDP (raw sockets modification)
- 1st version kernel was for kernel 2.4. Latest version for 2.6.x, windows, etc
- Sniffing the traffic is a problem when the communication is encrypted
- Intercepts all 'read' syscalls, 'socket' syscall, 'fork' syscall, 'clone' sycall.
- The information gathered from 'socket' syscall is correlated with the traffic gather from the honeywall (process and flow)
- The information gathered from 'open' syscall permits to maps processes with files. So it's possible to know which files have been opened during the intrusion.
- The information gathered from 'fork' sycalls permits to know the processes relationship and rebuild the whole execution

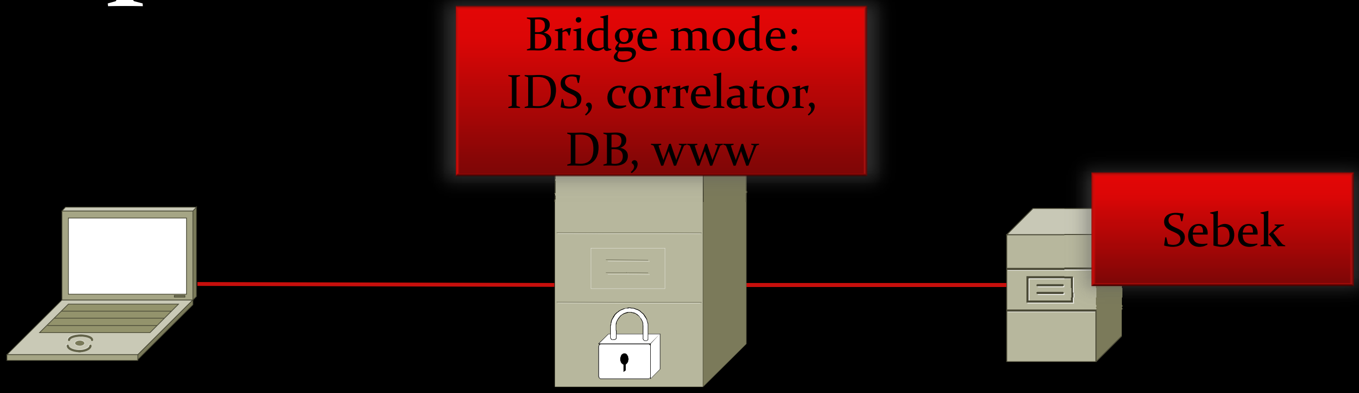
Sebek: ‘issues’

- Capture the response received from the attacker (already done with some patches)
- Can be detected: cat /proc/modules
- Sebek Linux sycall table modification can be detected and overwritten.
- It does not survive after a reboot: (Kernel Patch?? – not so flexible to analyze a real intrusion)

Honeywall

- Tool to gather the information from different honeypots
- It correlates: hosts, processes, files and network flows.
  - Sebek: with the syscalls it's possible to gather all this information and do relations between them
- Traffic captured with *tcpdump* and analyze with *pof, snort* and A*rgus*
- Database to store all the information
- Walleye: web Interface

Example of correlation



- 1st : detection of suspiciuos alerts in the flows (IDS).
  - Exploit to samba on 139/tcp
- 2nd: we can see which process is related to that flow
  - Ps: 6781 Sambad, 6781 /bin/bash, 6782 nmap, etc
- 3rd: it's possible to see which files have been opened.
  - Write /etc/passwd and /etc/shadow

Installation

- Download the package: https://projects.honeynet.org/sebek/attachment/wiki/WikiStart/sebek_disable_raw_socket_replacement-lin26-3.2.0b-bin.tar.gz
- tar fvxz sebek_disable_raw_socket_replacement-lin26-3.2.0b-bin.tar.gz && cd sebek_disable && ./configure; make;
  - This process creates a binary file
- Edit the installation file: vim “sebek_install.sh”
  - DESTINATION_PORT, MAGIC_VAL, KEYSTROKE_ONLY, SOCKET_TRACKING, TESTING, MODULE_NAME, WRITE_TRACKING
- Run the sbk_install.sh (it will load the kernel module)

Sebek Server (honeywall)

- sbk_extract: read from the network card (libpcap)
- sbk_ks_log.pl: process the logs and write to ‘stdout’
- sebekd.pl: process the logs and insert the information in a DB.
- Possible to apply filter to the ‘stdout’.

Some papers..

- Know your Enemy: Web Application Threats
  http://www.honeynet.org/papers/webapp
- Know your Enemy: Tracking Botnets
  http://www.honeynet.org/papers/bots
- Know Your Enemy: Malicious Web Servers
  http://www.honeynet.org/papers/mws
- Know your Enemy: Phishing
  http://www.honeynet.org/papers/phishing
- Know Your Enemy: Containing Conficker
  http://www.honeynet.org/papers/conficker

Bibliography

- Sebek: the honeypot projet http://www.honeynet.org/project/sebek/ https://projects.honeynet.org/sebek/

- Sebek 3: tracking the attackers, part one http://www.securityfocus.com/infocus/1855/2

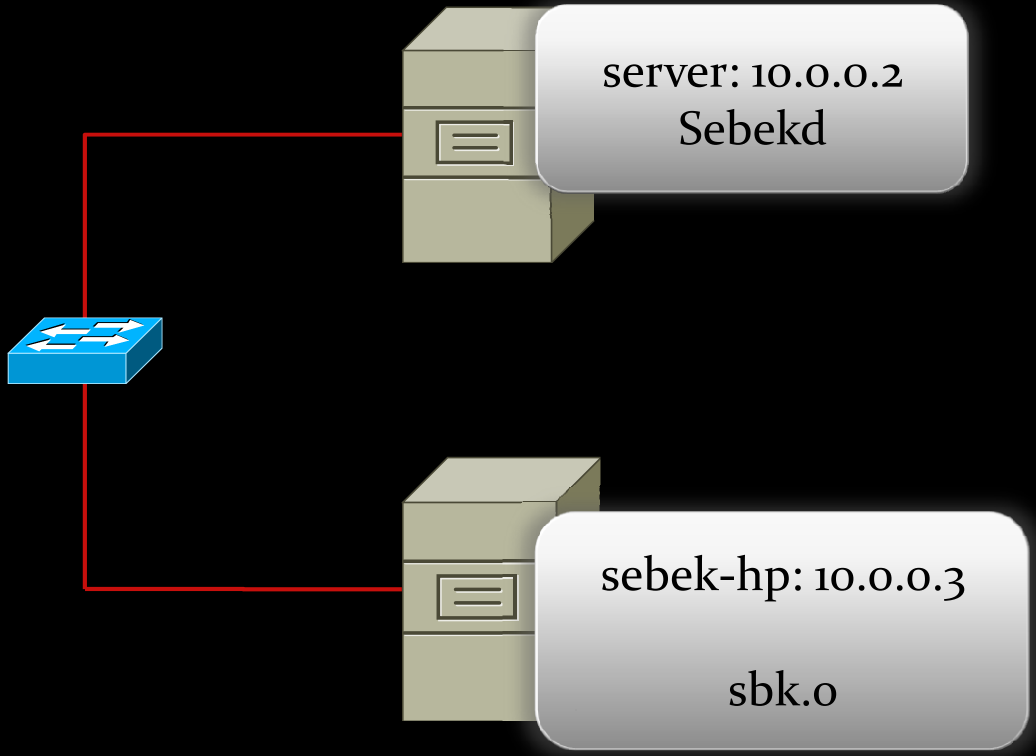
- Sebek 3: tracking the attackers, part two http://www.securityfocus.com/infocus/1858/2

- Building and Installing Sebek client in Ubuntu Server 7.10 https://projects.honeynet.org/sebek/wiki/Building%20and%20Installing%20Sebek%20client%20in%20Ubuntu%20Server%207.10

- Know your enemy: sebek http://old.honeynet.org/papers/sebek.pdf

- Xebek: a next generation honeypot monitoring system http://www.authorstream.com/Presentation/aSGuest18341-186692-ppt-honey-pot-entertainment-powerpoint/

Sebek example



Questions?