Efficient deployment of honeynets for statistical and forensic analysis of attacks from the Internet

Stephan Riebach, Erwin P. Rathgeb, Birger Toedtmann

University Duisburg-Essen
Computer Networking Technology Group
Institute for Experimental Mathematics
Ellernstr. 29
45326 Essen, Germany
Phone: +49 201 183-7636, Fax: +49 201 183-7673
(riebach,erwin.rathgeb,btoedtmann)@exp-math.uni-essen.de

Keywords: Security, Forensics, Honeypots, Honeynets

Abstract: The use of honeynets as a means to detect and observe attacks originating from the Internet as well as to allow forensic analysis is a technique that has received increasing attention in the research community. However, it has not yet been investigated how effective honeynets are and to what extent their efficiency can be actively improved. Therefore, after a short introduction to the honeynet concept and its implementation options, a case study will be presented providing some insight into this issue. For this case study, a honeynet has been implemented and a multilevel escalation strategy has been defined and employed to clarify to what extent the detected attacks represent just the "average" level of malware activity and to what extent honeynet owners can actively attract attacks or even influence specific types of attacks.

1 Introduction

Along with the increasing penetration of powerful personal computers and the rapid evolution of the internet, the issue of malware and hacker attacks has exploded at the same pace. To cope with this issue, appropriate protection tools, like e.g. firewalls or intrusion detection systems (IDS) are available for all segment sizes, starting from single PCs at home to large corporate networks. However, there is always a tradeoff between security and effort expended, against usage restrictions. In order to be able to balance these conflicting issues for a given scenario, a sound and fairly detailed risk assessment is crucial – both with respect to the probability of specific incidents and with respect to their sophistication. The latter is important, as the threats encountered in today's networks range from blind, fully automated worm and virus activities, via attacks with prefabricated scripts requiring no specific knowledge to highly specific and targeted attacks by expert hackers.

For obvious reasons, it is rather difficult to perform systematic measurements and observations in this area in real life production networks. Therefore, artificial

networks specifically set up to observe and document attack activities for offline analysis have been proposed which are now commonly known as “honeynets”. Over the last few years, this concept has been evolved and refined significantly and is becoming more widely used now. The information gathered in honeynets has already proven quite useful in the forensic analysis of attack activities and has provided valuable insight into various attack mechanisms.

After some intrusions in our own network, and based on the positive results published so far, we have implemented a honeynet in order to evaluate this concept. In addition to actually developing an understanding about the frequency and sophistication of attacks threatening our network, our goal was also to gain some insight into the factors determining the “efficiency” of a honeynet, i.e. its ability to actually attract attacks. This is especially interesting if the honeynet is used not to gather information, but instead as a decoy to divert attackers away from a production network operating in parallel. Even though a white paper [HON04] suggests that it is sufficient to just activate the honeynet as it will be found and attacked without further activities necessary, we wanted to find out to what extent the attractivity of a honeynet can possibly be influenced by the honeynet owner. Therefore we defined a phased escalation strategy which increasingly exposed our honeynet to the internet and observed the results in a field study whose first results will be presented in this paper.

After a short review of the honeynet concept in general and its various implementation options, we will describe our honeynet setup in some detail. We will also define and motivate the various steps of our escalation study before presenting some results with respect to the quantity and quality of the detected attack activities as well as with respect to the escalation study in particular.

2 Honeynets – Goals, Concepts and Implementation

The term “honeynet” was coined by a group of security experts organized in the “Honeynet Project” (www.honeynet.org). This group promotes the development and application of honeynet concepts and is the main source of the definitions used in this section.

The basic idea that led to the development of honeynets was to detect, observe and document the activity of hackers inside a computer network. Honeynets are highly specialized, artificial networks which have to be kept strictly separate from the actual production networks, have no real users – and thus no real traffic activity – and don’t contain any real information (user data). To be able to observe attacks, honeynets have to be vulnerable to a certain extent which means that they cannot be strictly protected by firewalls and that their systems should at least show some of the common vulnerabilities. Honeynets are highly controlled which means that elaborate monitoring and logging facilities capture and document all activity to provide comprehensive data for forensic analysis. Because they are artificial, all traffic in a honeynet is by definition suspicious, and traffic originating from a host in a honeynet is an indication that this system has likely been taken over.

In addition to the "honeypot" computers to be scanned, probed or attacked, a "data capture" function is required to make the honeynet useful. In addition to storing all data packets for offline forensic analysis, online monitoring with host and network based Intrusion Detection Systems (IDS) is useful to provide immediate notification about ongoing attacks as well as a basis for targeted forensic analysis. The data capture function can be distributed among several computers (including also the honeypots) or concentrated in a centralized device.

Because honeynets are intentionally vulnerable, so called "data control" mechanisms must be implemented to ensure that intruders cannot misuse compromised honeypots for further attacks. There are several ways to perform data control, e.g., limiting the outgoing bandwidth, restrictive outgoing packet filtering or, adding packet loss and high delays to outgoing connections [HON01]. Because it is particularly important that only the honeypots are visible and accessible for intruders, the data control and capture functions have to be hidden from intruders in order not to reveal the honeynet character of the network. In addition they have to be protected against any manipulation.

The honeynet concept has evolved significantly over the past few years, in particular with respect to implementing data capture and control functions [HON03a]. There is a broad spectrum of realization options for honeynets ranging from software emulating specific aspects of operating systems, applications and services (e.g. Honeyd, see www.honeyd.org) to real networks with hosts providing real services and applications. Whereas simple emulations allow only limited interaction, honeynets with live systems allow full interaction. However, the latter require significantly more effort for setup, configuration and maintenance.

3 Honeynet setup for the case study

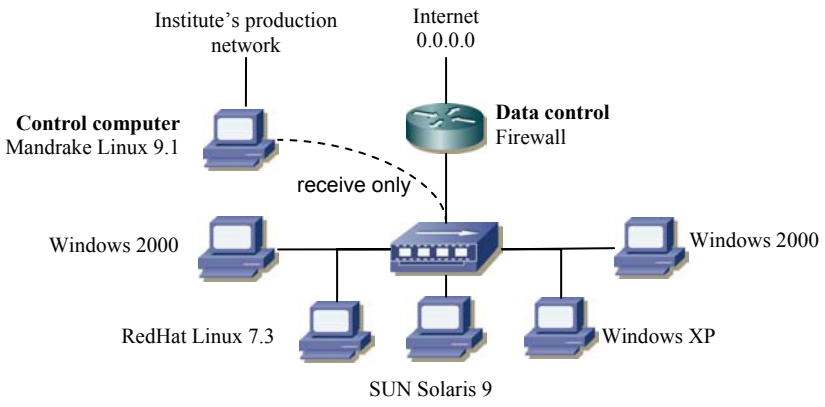

Fig. 1. The honeynet setup for the case study

In this case study, the honeynet architecture shown in Fig. 1 was used with 5 honeypots connected via a 100baseT hub. The honeynet was connected to the internet via a router (dual homed Linux machine) providing the data control function. We used packet filtering and additional bandwidth limitations for the outgoing traffic to avoid enabling successful attacks to be launched from compromised honeypots.

The control computer was set up with two network interfaces. A modified network cable was used to connect one of these interfaces to the honeynet. By modifying the cable as described in [ICO02], the control computer could receive all traffic from the honeynet but could (physically) not transmit any packets towards the honeynet. Therefore, it was fairly impossible for intruders in the honeynet to detect the presence of this machine and subsequently to attack it. Furthermore, the log data and reports collected on this machine could not be modified from the honeynet, preventing attackers from covering their tracks. Due to these precautions, it was possible to connect the other interface of the control computer to a production network for maintenance and remote data retrieval.

3.1 The Honeypots

As shown in Fig. 1, 2 of the honeypots were configured with Windows 2000 operating system , 1 with Windows XP, 1 with RedHat Linux 7.3 and the last with Solaris 9. This mix of operating systems was chosen because it is fairly typical for our production networks. Microsoft's operating systems are commonly used in the client desktop environment (CDE), with the migration from Windows 2000 to Windows XP starting at the time of our experiments. Unix-based systems (Linux and SUN Solaris) are also used in the production network for personal desktops and typically also as servers providing common services, e.g. http, ftp or nfs.

With respect to the vulnerability of the honeypots we updated to a patch level which was fairly typical for an environment where there is only limited central administration of the systems and the users have to take responsibility for their systems themselves. This means that the systems were not deliberately kept completely unmaintained and vulnerable to make the honeynet character of the network not too obvious. However, not all currently available security patches had been installed to also give also attackers using standard exploits (prefabricated attack scripts) a chance for success. The Windows systems, e.g., got upgraded with the current service pack and the patch for the RPC security leak published in July 2003.

Because of the heterogeneous operating systems, different Host Intrusion Detection Systems (HIDS) had to be used. The freeware tool "Tripwire" [TRIP04] was installed on the Linux system[1], "AIDE" [LETH04] was installed on the Solaris system. Because there was no appropriate freeware HIDS available for Windows the software "InstallWatch" was used there as an alternative. This software is originally intended to monitor installation routines on Windows systems. Therefore, it maintains a database of all system files, the registry and other user selected files and reports the

[1] "Tripwire" is no freeware for Solaris and Windows

changes after completing an installation. By installing small programs in regular intervals, a so called “poor man Tripwire system” [FLOYD00] is realized. Since the honeypots were not accessible remotely from the production network for security reasons the log files of all HIDSs were collected manually in regular intervals.

Complete images of the software installations of all honeypots were saved for all phases of the escalation study. Therefore, a compromised system could be restored to its original state with minimum effort. Before cleaning up a compromised system, we also saved a complete image for offline analysis and possible reinstallation for further observation.

3.2 Traffic monitoring and data capturing

The control computer was responsible for monitoring and capturing all network traffic in the honeynet. For traffic monitoring, the Network Intrusion Detection System (NIDS) “SNORT” [ROE04] was installed to identify known attack signatures in the honeynet traffic continuously and in real time. The SNORT log files were automatically archived once a day. Since they contain all incidents in chronological order only, they were automatically processed locally on the control computer by “SnortSnarf” [SIL03] which produced formatted statistical reports providing, e.g., an overview of the 10 most frequent targets, sources and attack signatures as well as statistics on all detected attack signatures sorted by severity classification and frequency. These statistics presented in HTML were automatically published on a web server on the control computer and could be remotely inspected from the production network. In addition, the major statistics files were automatically sent to the honeynet operator once a day.

For data capturing, the software utility “tcpdump” [TCP04] was deployed. With tcpdump all data traffic occurring in the honeynet was saved into daily dump files including all protocol overhead (addresses, etc.) from OSI layer 2 upwards. This high volume data could be retrieved remotely from the production network for offline analysis and archiving. To assure the permanent availability of these vital honeynet components, SNORT and tcpdump were monitored by using “Daemontools” [BER02] and automatically restarted after irregular shutdowns to avoid data loss.

3.3 Maintenance and data analysis

During normal operation, the honeynet generated roughly 1 Mbyte of SNORT log data and 75 Mbyte of tcpdump logs per day. This raw data was completely archived for statistical and forensic analysis. The statistical evaluation was highly automized as described above. The port scan log files generated were transformed into the CSV-format for detailed analysis in MS Excel. The automatic formatting and publishing of the SNORT logs allowed for a quick inspection and gave indications about potentially successful attacks which were then followed up. In addition, the Host Intrusion Detection Systems (HIDS) of the honeypots were collected and inspected on a daily basis so that a compromised honeypot could be identified rather quickly. In addition,

sporadic in-depth control and analysis of the honeypots was performed, to minimize the probability of undetected attacks.

Manual forensic analysis was performed in several cases where successful (non-automated) attacks could be detected. For manual inspection of the tcpdump log data, the programs “tcptrace” [OST04] was used to identify successful TCP connections related to successful attacks. “Ethereal” [ETH04] was then used to fully decode the packets of interesting connections up to the application level.

4 The phased escalation strategy

The experience reported by honeynet operators indicates that it is sufficient to just connect a honeynet to the internet and it will be found and attacked almost immediately [HON01]. We wanted to find out if the operator of a honeynet can actively influence the frequency and type of attacks. Therefore, we defined a four step escalation strategy for making our honeynet visible in the internet as follows:

In **phase 1**, the honeypots just had a basic installation of the operating system and offered only the services activated by default. Only the Linux honeypot was running a DNS server for name resolution in the local network; this server did not communicate with name servers outside the honeynet. The honeynet was then connected permanently to the internet without generating any outgoing traffic.

In **phase 2** the goal was to actively announce the existence of the honeynet in the internet. To achieve this, the host names were registered in the DNS servers of the university computing center and the zone transfer was activated in the local DNS server of the honeynet. In addition, a realistic domain name was registered for the honeynet. However, no specific services were offered by the honeypots and no outgoing traffic was generated.

In **phase 3**, we wanted to find out if the services offered by the honeypots would influence the attack patterns. In this phase we installed and activated commonly used services on the honeypots. Still, we didn’t actively generate outgoing traffic.

In **phase 4**, we installed Peer-to-Peer (P2P) file sharing applications (KazaaLite) on two of the Windows machines to evaluate if active participation in file sharing networks would have any impact on the attack patterns. To stimulate access to our honeypots, we provided some content for download. In order not to violate any copyright laws, we fabricated fake content by generating files of a predefined size filled with random numbers. These files were then converted to valid MP3 files by using the LAME (http://lame.sourceforge.net) MP3 encoder and named according to current top10 hits.

5 Results of the case study

In our case study each of the phases had a duration of three weeks. After completion of a phase, the recorded data was statistically evaluated. This analysis was mainly based on the SNORT log-files. SNORT classifies attack signatures into severity levels. In the following we distinguish between:

- **Alarm**: dangerous and harmful attacks (SNORT priority 1)
- **Warning**: suspicious signatures potentially preparing attacks (priority 2)
- **Notice**: unusual traffic not identified as dangerous (priority 3)

5.1 Attacking frequency

Since the first day the honeynet was running, activity could be detected confirming the statement that a honeynet will be found and attacked without further actions needed and that no significant "warmup" phase is required before starting statistical measurements. The honeypots have been scanned, probed or attacked every day with fluctuating intensity. Fig. 2 shows a typical summary of the recorded incidents. There is no obvious correlation between the intensity of alarms and warnings indicating the majority of attacks are blind, single phase attacks carried out without first scanning and fingerprinting the target (which would generate correlated warnings). One of our assumptions was that the attacking frequency would increase with the uptime of the honeynet because it becomes more widely known. However the measurements show that this is not the case and that external factors (malware activity) clearly define the attack frequency. Outbreaks of worm activities reported during the study could clearly be correlated to the measured attack intensity.

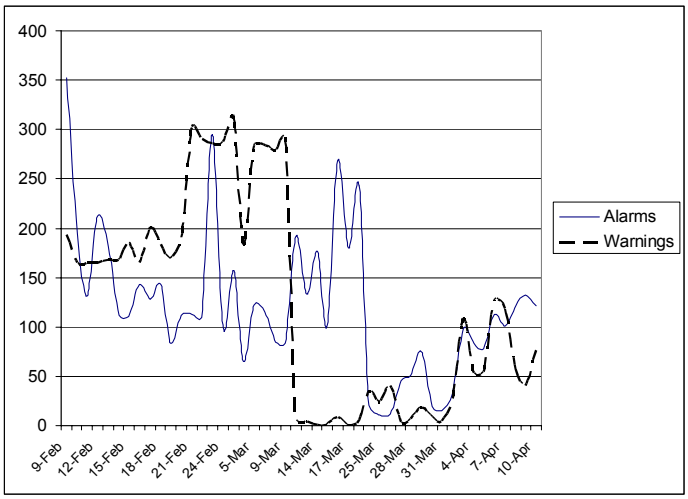

Fig. 2. Typical summary of the honeynet measurements from Feb. to Apr. 2004

5.2 Differences per operating system

In the next steps of the analysis the distribution of attacks among the honeypots was evaluated. Fig. 3 shows an aggregation of all attack signatures (priority 1-3) detected over the complete study period and their allocation to the honeypots. In each of the phases the Win2000 hosts were the primary targets of attack signatures, followed by the WinXP host. In all phases there were significantly less incidents reported for the UNIX systems Linux and Solaris.

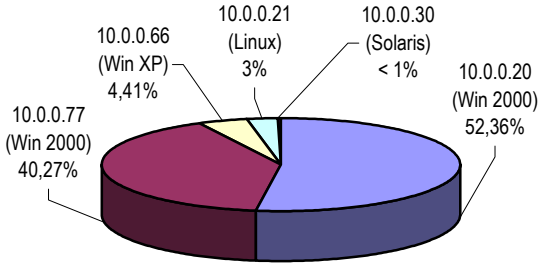

Fig. 3. Percentage of attack signatures per operating system[2]

The fact that nearly 97% of all signatures are targeted towards Windows systems was not unexpected since it makes sense to concentrate the effort to develop attacks on the clearly dominating operating systems. However, the conclusion that windows systems are significantly more insecure cannot be derived from these measurements as will be shown in the following.

5.3 Classification of attack signatures

We found a significant difference in the number of attack signatures per attack source (IP address of attacking host) between the Windows systems (average of 1) and the UNIX systems (average of 3). This led to the assumption that Windows attacks tend to be more blindly executed without first probing the target to prepare the attack. Since multi-phase attacks are more difficult to implement, single-phase attacks are probably automated to a larger extent. To validate this assumption the detected attack signatures were analyzed in more detail. By using the information provided by the "SANS Internet Storm Center" [http://isc.sans.org] the attack signatures of several active internet worms, especially MS-Blaster, Randex and SLAMMER have been identified. As a result, 81.2% of all alarms could be identified as automated worm attacks against Windows systems as shown in Fig. 4.

Only 1925 alarm signatures (18.8% of all detected alarms) were not worm related and thus potentially involve active human interaction. Since all of the worms try to exploit the same well known system vulnerability of Windows, all of the worm attacks can be

[2] We used a public Class-C network for our honeynet, whose address we do not publish here for security reasons.

blocked by installing one software patch. Therefore, the vast majority of attacks on Windows systems can obviously countered with minimum effort.

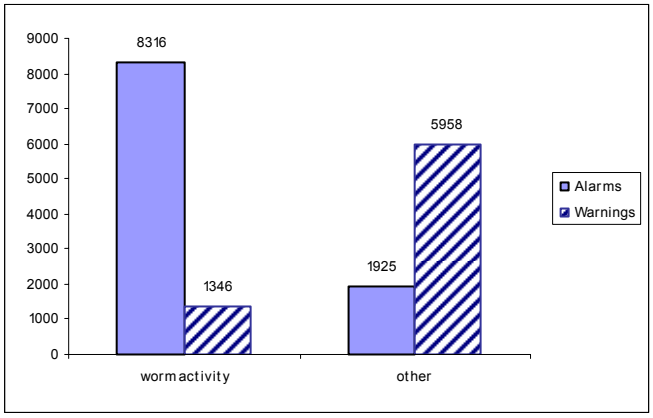

Fig. 4. Classification of attack signatures

For Linux systems, no specific worm activities could be identified. For the operation of a honeynet, the high number of identical worm attacks is only interesting for statistical purposes. Therefore it would make sense to automatically count and then filter out known worm signatures before generating the detailed reports, thus reducing the data volume and enabling the honeynet operator to concentrate on the more interesting attacks.

5.4 Impact of the escalation strategy on the attack frequency

One goal of the escalation study was to find out if there are methods to increase the attractivity of the honeynet and to attract more attacks. After eliminating attack frequency variations due to worm activity, the only configuration change deemed significant was the full activation of the DNS server. After starting the local DNS server and configuring the DNS reverse lookup on Dec. 16th 2003, the number of alarms increased significantly, as shown in Fig. 5, although no other configuration changes were made and no unusual waves of worm activity were reported. In particular, the DNS configuration caused a rise of the Microsoft specific RPC attack on port 135, so it can be assumed that this attack is correlated to DNS traffic.

The results of phase 4 were not quite as expected. The P2P search and download processes produced an enormous amount of data traffic, but no correlation could be detected between the P2P traffic and any attack signature. None of the IP addresses used in the P2P communication was involved in any non-P2P signature. Furthermore there was no temporal correlation between attacks and P2P traffic; also the overall attack frequency didn't increase significantly. From our results it can be concluded that making the honeynet known in the DNS is useful whereas actively generating traffic is not worthwhile and also clogs the log files with irrelevant data.

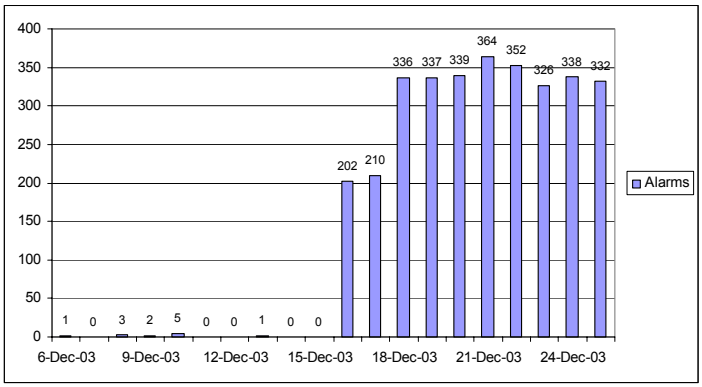

Fig. 5. Number of alarms in phase 2

5.5 Impact of the escalation strategy on the attack diversity

As shown in Table 1, the attacks generating alarms concentrated on the system services activated by default in the first two phases. In phase 3 when we activated http and ftp services, the variety of attacks increased significantly and these services became targets of specific attacks. However, the attacks were still unspecific in a sense that a significant part of the attacks on the web server were targeted towards the Microsoft IIS although only an Apache server was running. We also identified several attack signatures and a significant amount of attacks on popular services and applications not provided at all, e.g. SQL and SMTP servers. As a result, there seems to be a clear correlation between the variety of the popular services provided by the honeynet and the diversity of attack types.

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**
Appl. not provided	-	-	2	2
HTTP	-	-	3	4
FTP	1	-	5	5
System services (act.)	3	2	4	5
Total	**4**	**2**	**14**	**16**

Table 1. Number of different alarms per phase (grey fields mark provided services)

5.6 Usefulness for forensic analysis

In addition to statistical monitoring of attack activities, several distinct successful attacks were identified, observed and analyzed. In one example, one of the Win2000 honeypots was infected with the worm W32.Randex.Q. Besides trying to spread by

automatically probing vulnerabilities on Microsoft's ports 135/tcp and 445/tcp, the worm installed a backdoor and automatically reported it to a remote server camouflaging the communication as IRC traffic (chat). Subsequently, various activities indicating human interaction were performed on the compromised system using different user names. Finally a program to send spam mails was installed and activated. Due to the data control mechanisms employed in the honeynet, both worm infection and spam distribution could be confined to the honeynet automatically.

Another attack specifically analyzed was a classical multi-phase attack. The attacker first scanned the network with ICMP packets to find active computers and then scanned on port 111/tcp to identify systems providing the RPC portmapper, which is typical for UNIX systems. In the third step the attacker did a standard RPC query for the "cachefsd" service on the Solaris honeypot to find out that this service is provided on port 32775/tcp. Subsequently the attacker launched a buffer overflow attack for the vulnerability known already since 2001. This proves that honeynets are efficient in a sense that non-automated (interesting) attacks can be observed in fairly regular intervals.

6 Conclusion and Outlook

In this paper we presented a practical case study of using a honeynet in a university environment. The study was performed over a period of several months to find out how efficient honeynets are for attracting, detecting, observing and documenting hacker activities within a computer network. In general, the honeynet concept proved to be quite useful, but also required a significant and continuing effort for setup, maintenance, supervision and analysis. The escalation strategy defined to identify factors which can increase the efficiency of a honeynet, i.e. the number and diversity of attack attempts per time period, led to the conclusion that the attack frequency is dominated by external factors, e.g. worm activity. However, by making the honeynet visible in the DNS system and by providing a comprehensive set of popular network services, the attractiveness of the honeynet can be increased. The active generation of traffic, e.g. by participating in P2P networks, however, seems to be counterproductive since it didn't attract more or more diverse attacks and made data analysis more difficult due to the massive amount of irrelevant data stored.

We are currently implementing a virtual honeynet [HON03b] in order to compare it to the classical setup with respect of efficiency and effort for implementation and maintenance. In addition we are considering mechanisms to filter out high volume, repeated and automated attacks to reduce the amount of stored data and to simplify the analysis of more interesting and divers attacks. Furthermore, we will continue our honeynet measurements in order to further investigate some effects we have observed.

References

[BER02] Bernstein, D.J.: Daemontools Homepage, http://cr.yp.to/daemontools.html

[ETH04] Official homepage for “Ethereal”, http://www.ethereal.com

[FLOYD00] „A poor-man Tripwire-like system on Windows 9x/NT” http://www.geocities.com/floydian_99/poormantripwire.html

[HON01] The Honeynet-Project: “Know Your Enemy: Revealing the security tools, tactics, and motives of the Black Hat community“, Indianapolis: Addison-Wesley, 2001, http://project.honeynet.org

[HON03a] Honeynet-Project: “Know Your Enemy: Honeynets” http://www.honeynet.org/papers/honeynet/index.html

[HON03b] Honeynet-Project: “Know Your Enemy: Defining Virtual Honeynets” http://www.honeynet.org/papers/honeynet/index.html

[HON03c] Honeynet-Project: “Know Your Enemy: GenII-Honeynets” http://www.honeynet.org/papers/gen2/

[HON04] Honeynet-Project: “Know Your Enemy: Honeynets in Universities” http://www.honeynet.org/papers/edu/

[ICO02] Building a “sniffing cable” by IronComet Consulting, http://www.ironcomet.com/sniffer.html

[LETH04] Lethi, Rami: “Advanced Intrusion Detection Environment” http://sourceforge.net/projects/aide

[MOOR03] Moore, David: “The Spread of the Sapphire/Slammer Worm” http://www.cs.berkeley.edu/~nweaver/sapphire/

[NOR01] Northcutt, S.; Novak J.: “IDS: Intrusion Detection-Systeme“, Bonn: mitp-Verlag, 2001.

[NOR99] Northcutt, S.: “Network Intrusion Detection – An Analysts’s Handbook“, Indianapolis: New Riders Publishing, 1999.

[OST04] Ostermann, Shawn: „Tcptrace – Official Homepage“ http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html

[PRO03] Provos, Niels: “A Virtual Honeypot Framework”, CITI Technical Report, 01.10.2003, http://www.citi.umich.edu/techreports/reports/citi-tr-03-1.pdf

[ROE04] Roesch, Marty; Caswell, Brian: “Snort homepage”, http://www.snort.org/

[SIL03] Offizial homepage for “Snort Snarf“, http://www.silicondefense.com/software/snortsnarf/

[SPIT03a] Spitzner, Lance: “Open Source Honeypots – Learning with Honeyd” vom 20.01.2003, http://www.securityfocus.com/infocus/1659

[SPIT03b] Spitzner, Lance: “Open Source Honeypots, Part Two: Deploying Honeyd in the Wild” vom 12.03.2003, http://www.securityfocus.com/infocus/1675

[TCP04] Official homepage for tcdump: http://www.tcpdump.org/

[TRIP04] Official homepage for Tripwire Open Source, http://www.tripwire.org/