U. Flegel, M. Meier (Eds.): Detection of Intrusions and Malware & Vulnerability Assessment;
pp. 113-128, Lecture Notes in Informatics (LNI) No. 46, Springer, Juli 2004

A Honeynet within the German Research Network – Experiences and Results

Helmut Reiser
Munich Network Management Team
Ludwig Maximilian University Munich
helmut.reiser@ifi.lmu.de

Gereon Volker
Munich Network Management Team
Technical University Munich
gereon.volker@mytum.de

Abstract: A honeynet is a special prepared network which is not used in normal business. It is a kind of playground to watch and learn the tactics of crackers. The only purpose of a honeynet is to be probed, attacked or compromised. During the operation other systems may not be harmed by an attack originated within the honeynet. In this paper the design, realization and operation of a honeynet built within the German Research Network (DFN) will be described. Concepts for continuously monitoring and securing the honeynet are introduced. A selection of the results of the operation phase will be presented as well.

1 Introduction

Due to increasing number and severity of attacks it is important for security administrators to know about tactics, motives and preferred targets of their adversaries. In this paper the design, operation and analysis of a honeynet built within the German Research Network will be described. A honeynet is a screened and controlled network of honeypots. A **honeypot** is a system whose single purpose is to be probed, attacked, or compromised, by attackers. Learning the tactics, tools, and motives of attackers is one reason to build honeypots. The other reason is to slow down an attack by engaging the attacker with a system whose only purpose is to be attacked. The honeypot should be sufficiently interesting for the attacker, to extend his efforts and spend a lot of time at this system. A **honeynet** is not a single system but a network. Within this honeynet different types of honeypots can be placed. The whole honeynet with all its systems is not used productively in regular business. Therefore, all traffic within this network must be originated by an attacker. The honeynet is located therefore behind a filter (the honeywall) where all inbound and outbound data is captured. This data is then analyzed to learn about the techniques of attackers [Ho01].

The honeynet described here has been set up at the Leibniz–Supercomputing Center within the German Research Network. The German Research Network is the Internet backbone for all universities and research institutes in Germany. In German it is called "Deutsches Forschungsnetz (DFN)". DFN is the national research network with upstream connections to the European research network Géant, to various commercial providers and the global Internet. DFN connects all research facilities and provides them Internet access and additional services (e.g. mail, WWW, Video-Conferencing, etc.). DFN is one of the largest Internet providers in Germany. In 2003 data transmitted per month, exceeded the petabyte barrier. The Leibniz–Supercomputing Center ("Leibniz Rechenzentrum", LRZ) in Munich operates the Scientific Network in Munich (MWN) and research facilities in the south part of Bavaria and is directly connected to one of the core routers of the DFN. The honeynet which will be described in this paper has been built at the LRZ. Its operation time lasted from July 15th until September 12th 2003.

Section 2 describes the requirements, the design, the architecture and the operation of the honeynet. Alarming mechanisms as well as analysis tools are described. The results of the deployment are pointed out in section 3. The paper is concluded by a "lessons learned" section and a view to further work.

2 Design, Operation and Analysis

To deploy a honeynet, three basic tasks must be carried out: data capture, data control and data analysis. Data capture deals with the recording of all traffic which arrives or leaves the honeynet. This must be done on several layers and on all systems within the honeynet. For this reason a few tools are installed on the honeywall and the honeypots which are described in the following.

Honeynets must be configured in a way that other systems cannot be harmed or attacked if an attacker breaks into the honeynet. This is the aim of data control. Data analysis means efficient analysis of the collected data. Therefore, tools are required which are able to help the administrator to analyze fast growing log files (up an even more than 100 MB a day) and to extract relevant information out of data noise.

In the following, we describe hardware and software architecture design to meet these requirements.

2.1 Honeynet Software– and Hardware–Architecture

Based on the ideas of Lance Spitzner, one of the founders of the honeynet project [Honb], a second-generation (Gen II) honeynet (see figure 1) has been selected for this work [Sp03]. A Gen I honeynet is placed behind a simple firewall with a capturing component. In a Gen II honeynet the "copy of the production net" is placed behind a honeynet sensor which is also called honeywall. The only way to reach a honeypot is throughout the honeywall.

A honeynet is a smaller copy of the network and system infrastructure within a certain organization. Therefore it should contain systems similar to those regularly used within the organization. In our case Linux and Windows honeypots have been placed in the honeynet.

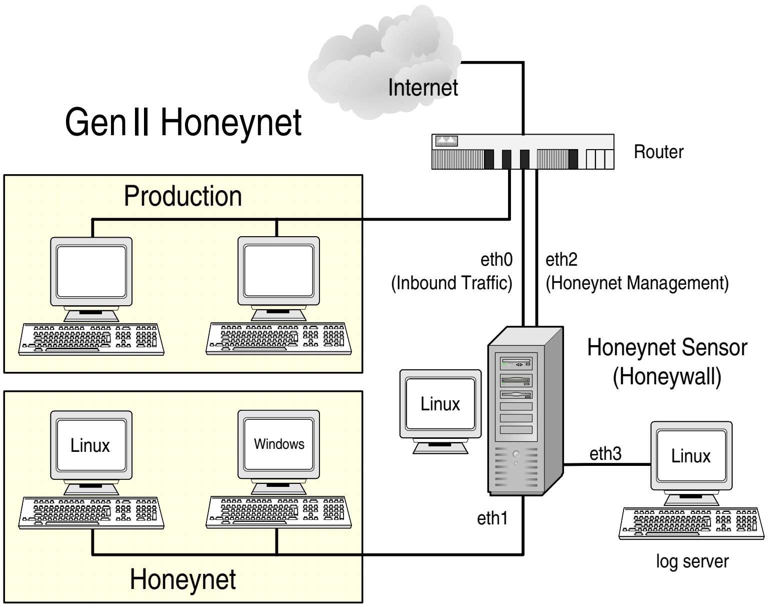

Figure 1: Gen II honeynet with additional log server

The honeywall is a more complex gateway, it is a bridging firewall with an intrusion detection system. It has to realize two properties: The attacker should not be able to detect the existence of the honeywall and for the operator of the honeynet it is an efficient analyzing, filtering and controlling tool.

From the attackers point of view the honeywall acts like a bridge which means that the honeywall is nearly "invisible" to the attacker. There is no decrement of the time-to-live (TTL) field in the IP–Header, there is no packet routing (bridges operate on layer two of the ISO/OSI model) and there are no MAC addresses to identify. Bridges usually use the spanning tree protocol to detect loops. This protocol is deactivated at the honeywall during operation. The only way to recognize the bridge respectively another system between the attacker and a honeypot is to capture the network traffic on one of the honeypots. If the attacker exceeds the outgoing connection limit or starts new attacks which get dropped by the controlling component of the honeywall he might wonder why his attacks do not reach the destination system. However in this case the attacker has already gained access to a honeypot and left behind lots of information about his attack. Even if he realizes that he has been bluffed, his traces can be used to reconstruct the way the attacker compromised the honeypot.

From the honeynet operators point of view the honeywall works like an extended firewall. All of the data passing the honeywall can be captured, and filtered. As the firewall component does not analyze any content of packets an Intrusion Detection System (IDS) and a controlling component is installed. The intrusion detection system is realized with snort

[snoa]. Snort controls the traffic from the Internet to the honeypots. Known attacks (especially attacks on web servers) can be easily detected: snort compares each packet with its internal database of signatures. A signature describes a kind of a patten of a known attack. If a packet matches one of these signatures an alert is generated or an action can be triggered. To simplify the analysis self defined signatures can be added. This feature has been used during the analysis phase to separate data noise from interesting traffic and new attacks. Data noise in this case is traffic which is already known and analyzed, e.g. probes which can be seen regularly, known mass attack–attempts from certain sources or senseless attacks like testing Windows IIS exploits on a Linux honeypot.

Snort_inline [snob] is a modified version of snort. It is intended as a protection for foreign hosts. It must be prevented that a honeypot is used as a platform for further attacks on other systems in the Internet. Therefore the firewall forwards outgoing packets from a honeypot to snort_inline (via the queue target of iptables [ipta]). The signatures are similiar to snort and all recognized attacks from one of the honeypots are dropped by snort_inline. Even if a worm (e.g. like Blaster or MS Slammer) or another kind of "automatic" attack reaches one of the honeypots this will not harm other systems. One deficiency of snort is that this kind of signature based IDS is unable to detect new attacks which are not known and therefore not represented in the signature database. In consequence a concept is necessary for protecting the "rest of the world" from unknown attacks.

For that purpose a controlling component has been installed to restrict the number of outgoing connections. The firewall which counts all outgoing connections could be used as such a controlling component on the honeywall. In our case each honeypot is allowed to have 15 outgoing connections per day (in each case for TCP, UDP and ICMP). This number is quite restrictive and the asymmetry between inbound and outbound traffic bandwidth might give an attacker a hint that he is within an honeynet. We are aware of that problem, however our most important requirement is to protect foreign systems from damage caused by our honeynet.

Besides this protection and control system a data capturing concept has been implemented. Data capture is done on all systems and on several layers:

1. All network traffic (inbound and outbound) will be dumped on the honeywall. To log binary dumps of all packets tcpdump has been used.

2. Firewall logfiles are more general than network dumps. Several tools exist to analyze these files. Analyzing the tcpdump log files is very time-consuming so it's much easier to get an overview of the events by firewall logfiles and select the most interesting binary dumps with this information.

3. A mechanism to dump attacker's keystrokes on each honeypot is required. Only with shell inputs the attacker's approach could be analyzed. Members of the Honeynet Project developed some tools to dump the keystrokes: for Windows platforms exists a tool called ComLog [com] on Unix systems Sebek [seb] is used. This tool is a modified rootkit based on the Adore rootkit [ado]. It forwards keystrokes and even SSH connections to a specified host. In this case these messages are forwarded to the honeywall. The messages can't be recognized by the attacker because Sebek

puts the data directly to the network device driver and not via the socket interface and the TCP/IP stack of the kernel [Mc03]. The attacker does not see any suspicious network connections. Even if he puts the network interface in promiscuous mode he is not able to see related outgoing packets.

4. On each honeypot the local logfile entries are forwarded to a central log server. This prevents from modifications of local logs concealing the attacker if a honeypot has been taken over. To ensure the forwarding to the log server and to camouflage forwarding a second syslog daemon (with a hidden configuration file) is installed and configured on the Linux honeypot. The Windows eventlog is forwarded with Eventlog to syslog [eve] to this log server.

Especially for forwarding of local logs a central log server is needed. The first idea was to place it together with the honeypots within the honeynet, but in this configuration the log server becomes another honeypot and might also be a potential aim for an attack which is not intended. For this reason the honeywall is equipped with a fourth network interface card where the log server is connected to.

The resulting architecture consists of a honeywall two honeypots (operating systems Windows 2000 and SuSE Linux 8.0) which are connected to the honeywall (cf. figure 1). Two interfaces on the honeywall are used for inbound and outbound honeynet traffic (eth0 and eth1), one is a dedicated management interface to administrate the architecture remotely (eth2) and the fourth is needed for the log server (eth3). The IP address of the management interface is placed in a different and additionally protected subnet which can not be reached directly from the honeynet. The log interface is protected by the firewall.

The Windows honeypot is installed without any Service Packs or patches for IIS or Windows on the Linux honeypot a default installation with X environment (also without any patches) was chosen.

The honeypots must be configured in a way that attackers should not notice the existence of the honeynet and the honeypots must be as interesting that they will be selected as a target for an attack. For this reason several services are set up and "attractive" names for the systems must be found to suggest productive systems. One honeypot is named internal.lrz-muenchen.de to indicate a host where confidential information is stored. The other one got the name tivoli.lrz-muenchen.de to pretend a running network management host.

The following services are configured:

- Web servers: On both honeypots are web servers installed, an Internet Information Server (IIS) on the Windows honeypot, which is included with Microsoft Windows 2000 Professional. There are no web pages configured to pretend the installation was done by a careless administrator who forgot this installed service. An Apache web server with PHP is configured on the Linux honeypot. To generate content every day two statistics of the daily traffic amount are generated by the accounting server of the LRZ and copied to the Linux honeypot. A Perl script randomizes the IP-addresses and stores the timestamp of the generation in a MySQL database to implement a history function.

Name	Function	CPU/RAM	Operating System
gway.lrz-muenchen.de	Honeynet sensor	PIII 500 MHz 128 MB RAM	SuSE Linux 8.3
tivoli.lrz-muenchen.de	Honeypot I	Pentium 200MHz 256 MB RAM	Microsoft Windows 2000 (without any Service Pack)
internal.lrz-muenchen.de	Honeypot II	Pentium 200MHz 64 MB RAM	SuSE Linux 8.0 (default installation)
logging.lrz-muenchen.de	Log server	Pentium 200MHz 128 MB RAM	SuSE Linux 8.3

Table 1: Systems building the honeynet

- FTP servers: Both honeypots have a FTP server installed. Download-able data (a Knoppix distribution) is available on the Windows honeypot.
- SSH server: A SSH server is installed per default on the Linux honeypot.
- Database server: The MySQL database which is used for the generation of web pages is also directly accessible from the Internet.

In addition, users with weak passwords have been created on both honeypots to provoke password guessing and account cracking. Table 1 summarizes the installed systems within the honeynet.

2.2 Alarming

Operation of a honeynet is a time consuming and critical task. During operation it is necessary to keep the administrator informed about the ongoing incidents in the honeynet. Therefore different notification mechanisms were set up on the honeypots and on the gateway. Notifications via email was one way to inform the administrator.

In the honeynet a program called swatch [swa] watches the firewall logfile. If a new entry was added to the logfile swatch sends out an email to predefined addresses. To restrict the traffic and prevent a mailbox overflow limits were set: Maximum ten emails were sent within an hour. Additionally all entries within one hour were counted. If the value of this counter exceeds more than 25 an additional email informed the administrator. Besides notifications via email Short Message Service (SMS) was used as a further alarming mechanism which allows the administrator greater mobility. As SMS is pretty expensive the notification via SMS was highly restricted. A maximum of two SMS per hour and only if outgoing connections were registered were sent.

2.3 Analysis Tools

During and after operation the collected data must be analyzed. There are three main tasks: Logfile Analysis, binary packet analyzing and investigation of the source of the attack and the tools used. Some tools are suitable to do an offline analysis; especially web based tools are better for online analysis. The following tools were used:

- Logfile Analysis:
  The honeynet administrator had to investigate snort logs, firewall logs and honeypot logs. To have efficient searching and querying possibilities all logfile entries were additionally stored in a database (e.g., MySQL). To analyze snort logs ACID [aci] was used, which is a PHP based web frontend for snort. ACID allows, for example, to generate charts, summarize alerts, select time frames or to take a look at the content of selected packets. All logs during the whole operation time of the honeynet have to be accessible, therefore ACID has to cope with huge amounts of data (cf. figure 2). The performance of the hosting system is the determining factor for ACID response times.

  The firewall was implemented using Linux iptables. To evaluate the logfiles, iptables log, a PHP/MySQL based web frontend was used [iptb]. This tool stores every entry of the iptables logfile in a database. With the database approach entries are easier to read, easier to group and easier to analyze than the logfile itself.

  Snort_inline [snob] logs all outgoing attacks and drops them. These logs are important if a successful attack hit one of the honeypots because it's possible to gain information about further propagation of the attack. Also ACID could be used here.

  This kind of analysis is very suitable for online analysis. If the administrator gets informed about an incident he can easily take a look at ACID or iptables log via a web browser.

- Binary packet analysis:
  All inbound and outbound traffic was dumped via tcpdump [tcp]. Packetyzer [pac] and Ethereal [eth] are efficient and handy tools to cope with tcpdump logfiles and support decoding of several protocols. Packetyzer (for Windows platforms only) is easier to handle and allows searching for patterns within captured packets. Ethereal is recommended for UNIX platforms. Normally binary packet analysis is done offline.

- Investigation of the source of the attack:
  One of the interesting questions during an analysis is the source (subnetwork or domain) of an attack. An IP address doesn't contain any information about the geographical position of the system the attacker uses. If nslookup returns a hostname of the IP address a rough guess of the location is possible. With traceroute transit systems between the honeynet and the attacker's system might be determined. Unfortunately traceroute doesn't determine the country where the system is located. Visualroute [vis] shows results of traceroute on a world map using known locations

of a lot of transit systems. This approach is far from being perfect, however in quite a lot of cases it reveals helpful and useful information.

By these tools it is only possible to determine the name and the rough location of the attacker's system. P0f [p0f] allows to identify the operating system of the attacker's host by using a technique called passive fingerprinting. Unlike active fingerprinting where packets are sent to the foreign host, passive fingerprinting uses the passing network traffic and analyzes the TCP options [Do].

3 Results

The honeynet was in operation between July 15th and September 12th 2003. At no time existence of the new subnet was propagated actively and no connections were made from the honeynet to other sites, e.g. there was no mail, news or www traffic. The honeynet was brought online at 8:55 am (GMT+1) on July 15th and two minutes later the first successful attack — a CodeRed2 on Microsoft IIS — hit the Windows honeypot. The honeynet was frequented quite often enabling us to collect a high amount of data (cf., figure 2).

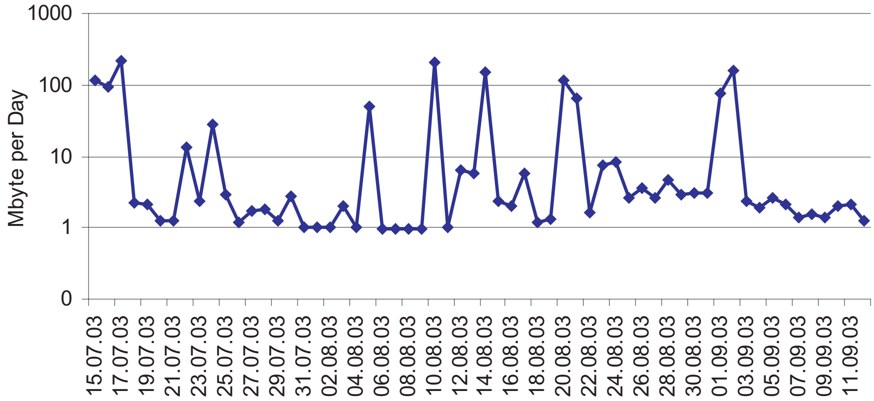

Figure 2: Honeynet traffic

Most attacks targeted the Windows systems. Regarding the services being attacked (cf., figure 3) one can observe that more than half of all attacks (57 % in sum) tried to exploit one of the plenty Windows NetBIOS vulnerabilities. The second biggest group, 36 % of the traffic, targeted Web servers. Incidents presented in the following can be classified in four main categories: Attacks against web servers, worm attacks, spoofing attacks and noise traffic.

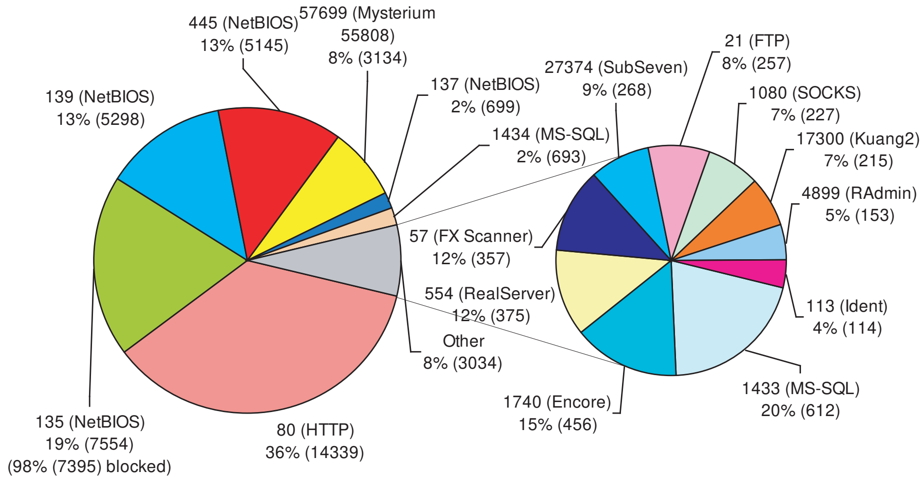

Figure 3: Attacked ports (services) and frequency

3.1 General Observations

At the start–up of the honeynet (July 15th), additional port filters had already been enabled on the DFN-Gateway. This filters block destination ports especially for avoiding well-known and easy exploitable bugs in different Windows NetBIOS system (i.e. ports 135, 137, 138, 139, 445 and 593 are blocked; see [lrz]). During the operation phase it was decided to open these ports to observe all attacks. On August 12th these filters were disabled for the honeynet which resulted in a heavy rise of traffic per day (figure 4).

In figure 5 the number of attacks against port 80 and port 139 for both honeypots are shown. The lines are mostly "parallel" for both systems. Even if a honeypot is not vulnerable the attackers tried to probe both systems. This suggest heavy tool usage and broad scans of Script–Kiddies. Often tools have been seen which probe the kind of OS and than stop on systems which are not vulnerable. This behavior explains the heavy differences in the number of attacks regarding both honeypots.

An analysis on source addresses of packets arriving at the honeynet revealed interesting aspects regarding the distribution of source domains (cf., figure 6). The transformation used were reverse DNS lookups directly on the firewall. The vast majority of attacks were carried out from `t-dialin.net`. Addresses of digital subscriber line (DSL) of "Deutsche Telekom" are bound to this domain. For the second largest group (36%) a reverse DNS lookup (promptly triggered by iptables) did not succeed, giving that group the name "unknown". IP–spoofing or IP–addresses from private networks may be possibly the reason for this fact, although some IP–addresses simply might have no reverse lookup defined by intention.

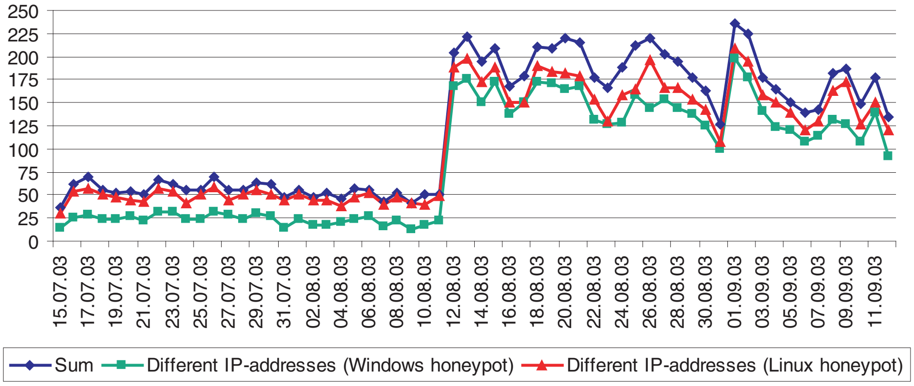

Figure 4: Number of different registered IP–addresses per day

3.2 Web attacks

Web servers (especially Microsoft IIS based ones) frequently become victims of worm code because of plenty and well known (but mostly unfixed) bugs in the software, e.g. vulnerability to buffer overflows.

The IIS webservers in our honeynet was successfully hit by an attack identified as Code Red II [dfn] only two minutes after being online. Even on the Linux honeypot, this attack was recognized eight times during the two month of operation. This fact raises the assumption that script kiddies are randomly trying to attack systems without determining the type of the running web server software in advance.

Some attackers tried to harm the web servers by buffer overflow attacks sending large HTML requests. In most cases the request length was 4096 characters.

3.3 Worm attacks

During the honeynet's operation time the Blaster worm (also known as Lovsan or W32Blaster [cer]) spread out over the globe. It's first appearance at the honeynet was recognized on August 11th at 10:56 pm. A client within the Munich Research Network (MWN) — most likely a notebook — sent requests to port 135 over the whole network, also hitting the Windows honeypot. Further dissemination of the worm was circumvented by snort_inline blocking outgoing TFTP request. Such a connection would have been necessary for the worm to retrieve a file called `msblaster.exe` which was needed to spread further.

The worm's impact on the Windows honeypot was to kill the remote procedure call subsystem (RPC), causing messages in the eventlog (one approximately every 2 minutes).

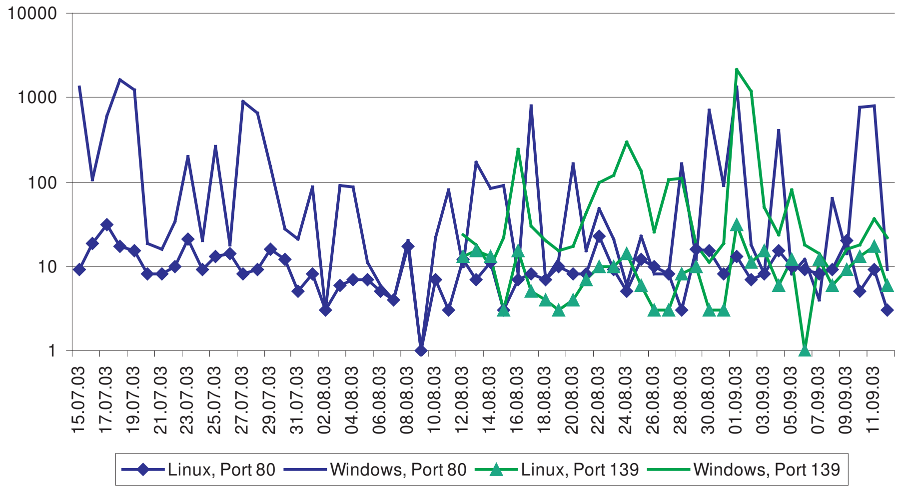

Figure 5: Number of attacks on ports 80 and 139 comparing both honeypots

The system was restored to a clean state by rebooting. Finally, on August 12th at 12:30 am a new rule for dropping requests to port 135/TCP had to be added to the honeywall, as at that time every two minutes the Windows honeypot was hit by a new attack.

On August 20th a variant of the Blaster worm (GaobotAA) [sym] hit the Windows honeypot. This worm also uses the RPC vulnerability but it connects via ports 139 and 445. The execution of the worm code (a file called `winhlpp32.exe` (about 58 KB) was dropped in %WindDir%\system32) inducing a connection to the worm's source system on destination port 22226 or 22227. Via that connection data could be exchanged. However GaobotAA was not able to download code because of the former mentioned new blocking rule for port 135. After September 7th no more attacks of this worm were recognized.

3.4 DoS-Attacks

Between August 26th and September 12th nameservers of different providers in the USA probably became victims of denial of service (DoS) attacks. The queries' source address was set to one of the honeypots' IP–address, the destination port was 53 or 80. The hosts replied to the given IP–address sending response packets with set SYN/ACK–Flag. The honeypots itself replied with a RST-Flag because the source port of the original query was 1740 and this port was closed by default. The honeypots received only the spoofed answers and were not among the intended victims.

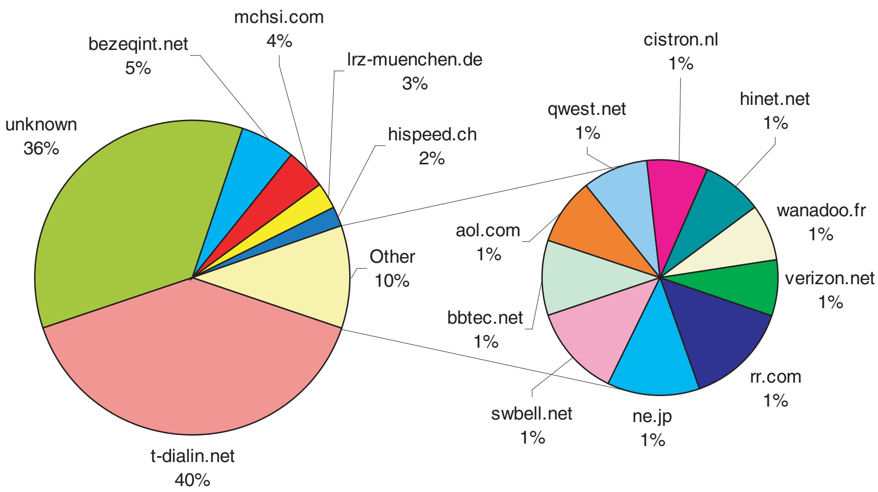

Figure 6: Distribution of DNS domains (more than 250 incoming connections)

3.5 Stumbler/"Mysterium 55808"

Since the beginning of operation, packets with a very high window size were recognized only on the Linux honeypot. The window size was set to 55808 and destination port was always set to 57669. Intrusec [int] and ISS [iss] called the causing trojan "Stumbler" and investigated these packets. The characteristic of Stumbler is its window size and a spoofed source IP–addresses. The trend of this activity is shown in figure 7.

3.6 "Noise"

Lots of connection requests were registered which tried to set up a connection to known trojan ports like Skydance respectively IRC (Port 4000), RAdmin (Port 4899), NetBus (Port 12345), Kuang 2 (Port 17300) and SubSeven (Port 27374). None of these ports were in listen state so all requests have been confirmed with RST. Nearly the same number of requests were noticed on both honeypots. This is also an evidence that script kiddies tried to find vulnerable systems by doing random scans. The same situation showed up with the classic proxy ports (3128, 8080) and the SOCKS port (1080).

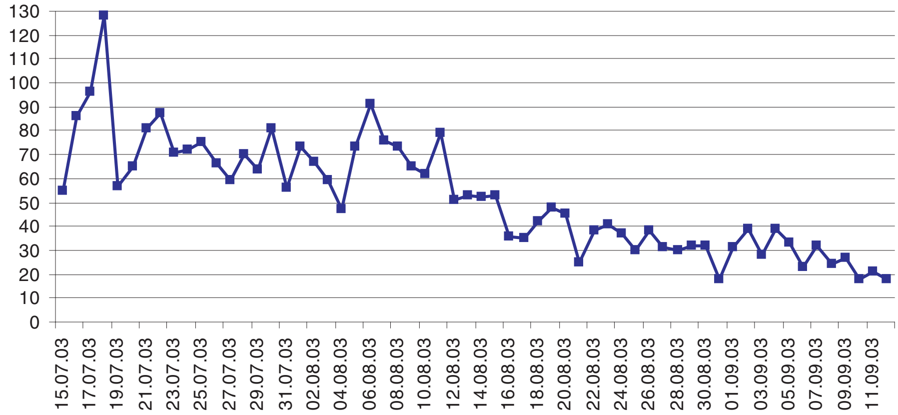

Figure 7: Received Packets per day with window size 55808

4 Lessons learned and future work

This work showed that computer systems which are placed in the Internet without publishing their exitance are under attack extremely fast. The honeynet got online and only two minutes later the first attack took place. “Unfortunately” no “real” or “clever” hostile take-over happened; neither to windows nor to linux. Nevertheless, it was very interesting to watch and analyze the ongoing events in the honeynet, especially the propagation of the Blaster worm.

The honeynet project presents the results and traces of clever attacks. However our experience was, that such attacks are quite seldom. This implies that more than 90 % of the attacks can be defended quite easily by applying a patch for vulnerable software.

The Windows platform was the favorite target of our attackers, 69 % of the attacks aim at this platform. On the Windows honeypot one third of all connections tried to attack the Web server. On the Linux honeypot however we could not detect any serious or successful attack.

A lot of our attackers can be categorized as Script–Kiddies. They are low skilled and mostly use scripts written by more sophisticated crackers. We infer a Script–Kiddie based on the following reasons: A lot of attacks were web attacks using tools which are available in the Internet e.g. FX Scanner [fxs]. These tools are suited for automated attacks over a wide range of IP-addresses and subnets (brute-force scans) without knowing anything about existing systems and their vulnerabilities. Scans have been seen, trying to exploit a certain OS–dependent bug, on both honeypots regardless the used operating system.
Another big group of unintentional “attackers” became infected by a worm or virus, which started to spread out and hit the honeynet casually.

Regarding the sources of the attacks: More than 40% could be assigned to systems which use the Deutsche Telekom as ISP (based on the IP-addresses and domain names). For the attacker such a big provider might be used as a kind of “anonymizer”. Each dial in results in a changed IP address. Without judicial authorization it is nearly impossible to find further informations about the “real” source of the attack.

To sum it up: Most attackers are low skilled, the favorite target of attacks was Windows especially the IIS. In most cases the security level could be improved quite easily. A firewall blocking services which are a cinch to exploit and a patch management which operates continuously, consequently and rapidly are the two most important building blocks for higher security.

Deploying a honeynet the way described here and especially its operation requires a lot of time (supervisioning and analyzing the collected data) and a lot of knowledge about operating systems. However, a honeynet can give hints to ongoing mass attacks and very important information about propagation and effects of such attacks (e.g. worms). A honeynet can slow down an attack. It shows which systems used in the production network are mostly attacked and which services are preferred. This insights can influence the security policy of an organization. Honeynets are very flexible and can be deployed for a multiplicity of scenarios.

During the operation of this work several new ideas were developed to enhance and achieve new insights e.g. how honeynets can help to fight spam. Therefore faked open proxies (e.g. Bubblegum proxypot [pro]) are configured which pretend to anonymize the spammers' data but this data is captured. A possible way to fight relay spam with a special sendmail configuration is pointed out in [fig].

We are interested to compare a virtual honeynet (Honeyd [hona]) with a dedicated system. In the virtual honeynet the honeypots are not represented as physical hosts but emulated on a single host.

A honeynet might also be used as an Intrusion Response System (IRS) or even as an Intrusion Prevention System (IPS). Detection of an attack can be coupled with active defense operations (e.g., closing ports or filtering certain sources). We are aware that this can be double-edged sword. However defending against extremely fast automatic attacks only an automatic defense system may help.

Acknowledgment

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. The web server of the MNM Team is located at `http://wwwmnmteam.informatik.uni-muenchen.de`.

References

[aci] ACID. http://acidlab.sourceforge.net/.

[ado] Adore rootkit. https://www.team-teso.net/releases/adore-0.39b4.tgz.

[cer] CERT Advisory CA-2003-20 W32/Blaster worm. http://www.cert.org/advisories/CA-2003-20.html.

[com] ComLog. http://www.geocities.com/floydian_99/comlog.html.

[dfn] Code Red2 Analyse und Gegenmassnahmen. http://www.dfn-cert.de/dfn/bt/2001/bt-2001-codered.pdf.

[Do] Doyle, B. Passive Fingerprinting Utilizing the Telnet Protocol Negotiation data. http://www.sans.org/resources/idfaq/fingerp_telnet.php.

[eth] Ethereal. http://www.ethereal.com/.

[eve] Eventlog to Syslog Utility. https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys.

[fig] Fighting Relay Spam the Honeypot Way. http://www.tracking-hackers.com/solutions/sendmail.html.

[fxs] FX Scanner. http://www.der-klan.de/tools/scanner.html.

[HAN99] Hegering, H.-G., Abeck, S., und Neumair, B.: *Integrated Management of Networked Systems — Concepts, Architectures and their Operational Application*. Morgan Kaufmann Publishers, ISBN 1–55860–571–1. January 1999. 651 p.

[Ho01] Honeynet Project (Hrsg.): *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley. 2001.

[hona] Honeyd. http://www.honeyd.org.

[Honb] The Honeynet Project. www.honeynet.org.

[Honc] The Honeynet Project: Frequently Asked Questions. http://www.honeynet.org/misc/faq.html.

[int] Intrusec Alert: 55808 Trojan Analysis. http://www.intrusec.com/55808.html.

[ipta] iptables. http://www.netfilter.org/.

[iptb] iptables log. http://www.gege.org/iptables/.

[iss] "Stumbler" Distributed Stealth Scanning Network. http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=22441.

[lrz] Einschränkungen und Regeln im Netzbetrieb. http://www.lrz-muenchen.de/services/netz/einschraenkung/.

[Mc03] McCarty, B.: The Honeynet Arms Race. *IEEE Security and Privacy*. 1(6). December 2003.

[p0f] P0f. http://lcamtuf.coredump.cx/p0f.shtml.

[pac] Packetyzer. http://www.networkchemistry.com/products/packetyzer/.

[pro] Bubblegum proxypot. http://world.std.com/~pacman/proxypot.html.

[seb] Know Your Enemy: Sebek2. http://www.honeynet.org/papers/sebek.pdf.

[snoa] Snort - The Open Source Network Intrusion Detection System. http://www.snort.org.

[snob] Snort_inline. http://sourceforge.net/projects/snort-inline/.

[Sp03] Spitzner, L.: The Honeynet Project: Trapping the Hackers. *IEEE Security and Privacy*. 1(2). March 2003.

[swa] Swatch. http://swatch.sourceforge.net/.

[sym] W32.HLLW.Gaobot.AA. http://www.symantec.com/avcenter/venc/data/w32.hllw.gaobot.aa.html.

[tcp] tcpdump. http://www.tcpdump.org/.

[vis] Visualroute. http://www.visualroute.com/.