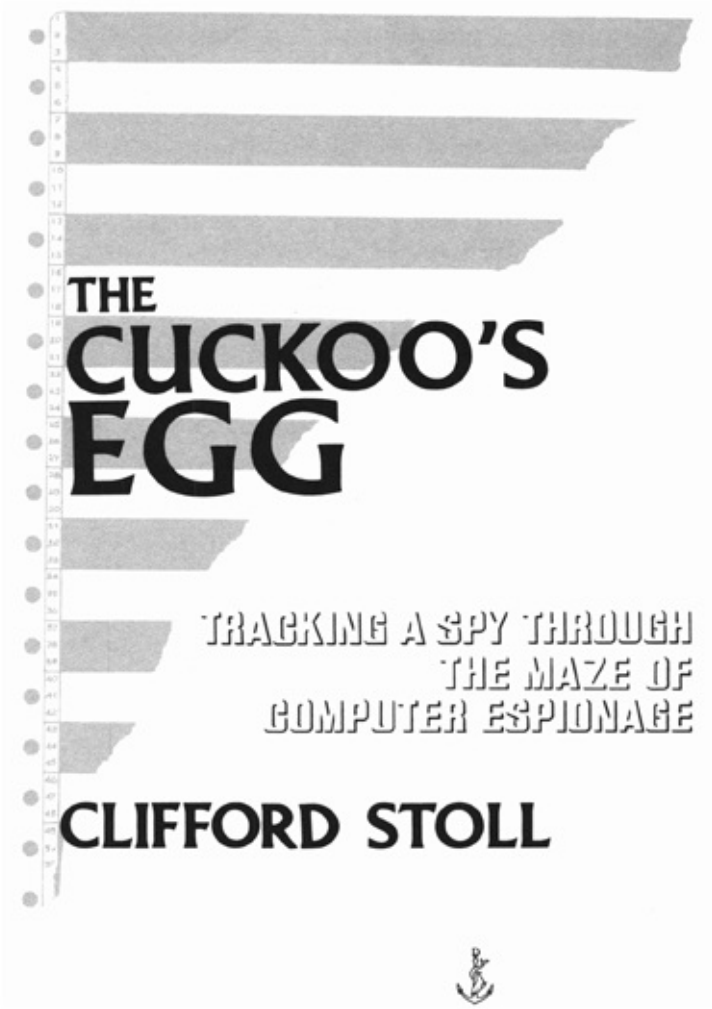
THE CUCKOO'S EGG

Tracking a Spy through the Maze of Computer Espionage

Clifford Stoll

Doubleday



THE CUCKOO'S EGG

TRACKING A SPY THROUGH THE MAZE OF COMPUTER ESPIONAGE

CLIFFORD STOLL

DOUBLEDAY

NEW YORK LONDON TORONTO SYDNEY AUCKLAND

PUBLISHED BY DOUBLEDAY

a division of Bantam Doubleday Dell Publishing Group, Inc.
666 Fifth Avenue, New York, New York 10103

DOUBLEDAY and the portrayal of an anchor with a dolphin are trademarks of Doubleday, a division of Bantam Doubleday D
Publishing Group, Inc.

Library of Congress Cataloging-in-Publication Data

Stoll, Clifford.
The cuckoo's egg : tracking a spy through the maze of computer espionage / by Clifford Stoll. — 1st ed.
p. cm.
1. Hess, Marcus. 2. Stoll, Clifford. 3. Espionage, Soviet—
United States. 4. Espionage, Soviet—Germany (West)—
Hannover. 5. Defense information, Classified—United States—Data
bases. 6. Computer crimes—United States. 7. Computer crimes—
Germany (West)—Hannover. I. Title.
UB271.R92H477 1989
364.1′68′0973—dc20 89-7808
364.1′68′0973—dc20

eISBN: 978-0-307-81942-0

v3.1

Contents

Acknowledgments

○ ○ ○ How do you spread the word when a computer has a security hole? Some sa nothing, fearing that telling people how to mix explosives will encourage them to mak bombs. In this book I've explicitly described some of these security problems, realizing th people in black hats are already aware of them.

I've tried to reconstruct this incident as I experienced it. My main sources are my logbool and diaries, cross-checked by contacting others involved in this affair and comparing repor from others. A few people appear under aliases, several phone numbers are changed, an some conversations have been recounted from memory, but there's no fictionalizing.

For supporting me throughout the investigation and writing, thanks to my friend colleagues, and family. Regina Wiggen has been my editorial mainstay; thanks also to Joche Sperber, Jon Rochlis, Dean Chacon, Donald Alvarez, Laurie McPherson, Rich Muller, Ger Spafford, Andy Goldstein, and Guy Consolmagno.

I posted a notice to several computer networks, asking for title suggestions. Sever hundred people from around the world replied with zany ideas. My thanks to Karen Anderso in San Francisco and Nigel Roberts in Munich for the title and subtitle.

Doubleday's editors, David Gernert and Scott Ferguson, have helped me throughout. T them, as well as my agent, John Brockman, thanks for your continued encouragement an wise advice.

To each of these people, I'm indebted; I owe most of them boxes of cookies as well.

Lawrence Berkeley Laboratory supported me throughout this quest; the people Smithsonian Astrophysical Observatory—especially Joe Schwarz and Steve Murray—hav been most gracious and supportive while I've been writing this book. My deep thanks go my friends at both institutes, and my hopes that I'll now be able to return to astronomy.

I was ten years old when Ernst Both of the Buffalo Museum of Science invited me to loo through a telescope, opening up a universe of astronomy. I wonder if I'll ever be able thank him properly.

I needn't thank my sweetheart and wife, Martha Matthews. She's been as much a part writing this book as she was in the story. I love her with all my heart.

—Cliff Stoll-Matthev
cliff@cfa.harvard.ed

1

Me, a wizard? Until a week ago, I was an astronomer, contentedly designing telescope optics. Looking back on it, I'd lived in an academic dreamland. All these years, never planning for the future, right up to the day my grant money ran out.

Lucky for me that my laboratory recycled used astronomers. Instead of standing in th unemployment line, I found myself transferred from the Keck Observatory at the Lawrenc Berkeley Lab, down to the computer center in the basement of the same building.

Well, hell, I could fake enough computing to impress astronomers, and maybe pick it u fast enough that my co-workers wouldn't catch on. Still, a computer wizard? Not me—I'm a astronomer.

Now what? As I apathetically stared at my computer terminal, I still thought of planetar orbits and astrophysics. As new kid on the block, I had my choice of a cubicle with a windo facing the Golden Gate Bridge, or an unventilated office with a wall of bookshelve Swallowing my claustrophobia, I picked the office, hoping that nobody would notice when slept under the desk. On either side were offices of two systems people, Wayne Graves an Dave Cleveland, the old hands of the system. I soon got to know my neighbors through the bickering.

Viewing everyone as incompetent or lazy, Wayne was crossthreaded with the rest of th staff. Yet he knew the system thoroughly, from the disk driver software up to the microwav antennas. Wayne was weaned on Digital Equipment company's Vax computers and woul tolerate nothing less: not IBM, not Unix, not Macintoshes.

Dave Cleveland, our serene Unix buddha, patiently listened to Wayne's running stream o computer comparisons. A rare meeting didn't have Wayne's pitch, "Vaxes are the choice o scientists everywhere and helps build strong programs twelve ways." Dave retorted, "Loo you keep your Vax addicts happy and I'll handle the rest of the world." Dave never gave hi the satisfaction of getting riled, and Wayne's complaints eventually trailed off to a mutter.

Great. First day on the job, sandwiched between two characters who were already ruinin my daydreams with their periodic disputes.

At least nobody could complain about my appearance. I wore the standard Berkele corporate uniform: grubby shirt, faded jeans, long hair, and cheap sneakers. Manage occasionally wore ties, but productivity went down on the days they did.

Together, Wayne, Dave, and I were to run the computers as a lab-wide utility. We manage a dozen mainframe computers—giant workhorses for solving physics problems, togeth worth around six million dollars. The scientists using the computers were supposed to see simple, powerful computing system, as reliable as the electric company. This meant keepin the machines running full time, around the clock. And just like the electric company, w charged for every cycle of computing that was used.

Of four thousand laboratory employees, perhaps a quarter used the main computers. Eac of these one thousand accounts were tallied daily, and ledgers kept inside the computer. Wit an hour of computing costing three hundred dollars, our bookkeeping had to be accurate, s we kept track of every page printed, every block of disk space, and every minute o processor time. A separate computer gathered these statistics and sent monthly bills t laboratory departments.

And so it happened that on my second day at work, Dave wandered into my offic mumbling about a hiccup in the Unix accounting system. Someone must have used a fe

seconds of computing time without paying for it. The computer's books didn't quite balanc
last month's bills of $2,387 showed a 75-cent shortfall.

Now, an error of a few thousand dollars is obvious and isn't hard to find. But errors in th
pennies column arise from deeply buried problems, so finding these bugs is a natural test fo
a budding software wizard. Dave said that I ought to think about it.

"First-degree robbery, huh?" I responded.

"Figure it out, Cliff, and you'll amaze everyone," Dave said.

Well, this seemed like a fun toy, so I dug into the accounting program. I discovered ou
accounting software to be a patchwork of programs written by long-departed summe
students. Somehow, the hodgepodge worked well enough to be ignored. Looking at th
mixture of programs, I found the software in Assembler, Fortran, and Cobol, the most ancie
of computer languages. Might as well have been classical Greek, Latin, and Sanskrit.

As with most home-brew software, nobody had bothered to document our accountin
system. Only a fool would poke around such a labyrinth without a map.

Still, here was a plaything for the afternoon and a chance to explore the system. Dav
showed me how the system recorded each time someone connected to the computer, loggin
the user's name, and terminal. It timestamped each connection, recording which tasks th
user executed, how many seconds of processor time he used, and when he disconnected.

Dave explained that we had two independent accounting systems. The ordinary Un
accounting software just stored the timestamped records into a file. But to satisfy som
bureaucrat, Dave had built a second accounting system which kept more detailed records o
who was using the computer.

Over the years, a succession of bored summer students had written programs to analyze a
this accounting information. One program collected the data and stashed it into a file. A
second program read that file and figured how much to charge for that session. Yet a thir
program collected all these charges and printed out bills to be mailed to each departmen
The last program added up all user charges and compared that total to the result from th
computer's internal accounting program. Two accounting files, kept in parallel by differen
programs, ought to give the same answer.

For a year, these programs had run without a glitch, but weren't quite perfect this week
The obvious suspect was round-off error. Probably each accounting entry was correct, bu
when added together, tenths of a penny differences built up until an error of 75 cen
accumulated. I ought to be able to prove this either by analyzing how the programs worke
or by testing them with different data.

Rather than trying to understand the code for each program, I wrote a short program t
verify the data files. In a few minutes, I had checked the first program: indeed, it proper
collected the accounting data. No problem with the first.

The second program took me longer to figure out. In an hour I had slapped together enoug
makeshift code to prove that it actually worked. It just added up time intervals, the
multiplied by how much we charge for computer time. So the 75-cent error didn't come fro
this program.

And the third program worked perfectly. It looked at a list of authorized users, found the
laboratory accounts, and then printed out a bill. Round-off error? No, all of the program
kept track of money down to the hundredths of a penny. Strange. Where's this 75-cent erro

coming from?

Well, I'd invested a couple hours in trying to understand a trivial problem. I got stubbor dammit, I'd stay there till midnight, if I had to.

Several test programs later, I began actually to have confidence in the mishmash of locall built accounting programs. No question that the accounts didn't balance, but the program though not bulletproof, weren't dropping pennies. By now, I'd found the lists of authorize users, and figured out how the programs used the data structures to bill differer departments. Around 7 P.M. my eye caught one user, Hunter. This guy didn't have a val billing address.

Ha! Hunter used 75 cents of time in the past month, but nobody had paid for him.

Here's the source of our imbalance. Someone had screwed up when adding a user to ou system. A trivial problem caused by a trivial error.

Time to celebrate. While writing this first small triumph into the beginning pages of m notebook, Martha, my sweetheart, stopped by and we celebrated with late-night cappuccino at Berkeley's Cafe Roma.

A real wizard would have solved the problem in a few minutes. For me, it was unknow territory, and finding my way around hadn't been easy. As a consolation, I'd learned th accounting system and practiced a couple obsolete languages. Next day, I sent an electron mail message to Dave, preening my feathers by pointing out the problem to him.

Around noon, Dave stopped by to drop off a pile of manuals, and casually mentioned tha he had never added a user named Hunter—it must have been one of the other syste managers. Wayne's curt response: "It wasn't me. RTFM." Most of his sentences ended wi acronyms, this one meaning, "Read the fucking manual."

But I'd read the manuals. Operators weren't supposed to add a new user without a account. At other computer centers, you just log into a privileged account and tell the syste to add a new user. Since we also had to make several bookkeeping entries, we couldn't ru such a vanilla system. Ours was complex enough that we had special programs whic automatically did the paperwork and the systems juggling.

Checking around, I found that everyone agreed the automatic system was so superior tha nobody would have manually added a new user. And the automatic system wouldn't mak this mistake.

Well, I couldn't figure out who had made this goof. Nobody knew Hunter, and there wasn an account set for him. So I erased the name from the system—when he complained, w could set him up properly.

A day later, an obscure computer named Dockmaster sent us an electronic mail message. I system manager claimed that someone from our laboratory had tried to break into h computer over the weekend.

Dockmaster's return address might have been anywhere, but signs pointed to Marylan The e-mail had passed through a dozen other computers, and each had left a postmark.

Dave answered the message with a noncommittal "We'll look into it." Uh, sure. We'd loo when all our other problems disappeared.

Our laboratory's computers connect to thousands of other systems over a dozen network Any of our scientists can log into our computer, and then connect to a distant computer. Onc connected, they can log into the distant computer by entering an account name and passwor

In principle, the only thing protecting the networked computer is the password, since accou names are easy to figure out. (How do you find account names? Just use a phone book—mo people use their names on computers.)

Dockmaster's electronic mail message was a curiosity, and Dave passed it to Wayn attaching a question, "Who's Dockmaster?" Wayne forwarded it to me with his gue —"Probably some bank."

Eventually, Wayne bounced the message to me. I guessed Dockmaster was some Nav shipyard. It wasn't important, but it seemed worth spending a few minutes looking into.

The message gave the date and time when someone on our Unix computer tried to log in Dockmaster's computer. Since I'd just mucked around the accounting system, scrabble around the files, looking for records from Saturday morning at 8:46. Again, the tw accounting systems disagreed. The stock Unix accounting file showed a user, Sventek, loggin in at 8:25, doing nothing for half an hour, and then disconnecting. No timestamped activity between. Our home-brew software also recorded Sventek's activity, but it showed him usin the networks from 8:31 until 9:01 A.M.

Jeez. Another accounting problem. The time stamps didn't agree. One showed activi when the other account said everything was dormant.

Other things seemed more pressing, so I dropped the problem. After wasting an afternoo chasing after some operator's mistake, I wasn't about to touch the accounting system again.

Over lunch with Dave, I mentioned that Sventek was the only one connected whe Dockmaster reported the break-in. He stared and said, 'Joe Sventek? He's in Cambridg Cambridge, England. What's he doing back?' Turned out that Joe Sventek had been th laboratory's Unix guru, a software wizard who built a dozen major programs over the pa decade. Joe had left for England a year ago, leaving behind a glowing reputation throughou the California computer community.

Dave couldn't believe Joe was back in town, since none of Joe's other friends had hear from him. 'He must have entered our computer from some network,' Dave said.

'So you think Joe's responsible for this problem?' I asked Dave.

"No way," Dave replied. "Joe's a hacker of the old school. A smart, quick, capab programmer. Not one of those punks that have tarnished the word 'hacker.' In any cas Sventek wouldn't try to break into some Maryland computer. And if he did try, he'd succee without leaving any trace."

Curious: Joe Sventek's been in England a year, yet he shows up early Saturday mornin tries to break into a Maryland computer, disconnects, and leaves behind an unbalance accounting system. In the hallway I mention this to Wayne, who's heard that Joe's o vacation in England; he's hiding out in the backwoods, far away from any computers. "Forg that message from Dockmaster. Sventek's due to visit Berkeley RSN and he'll clear it up."

RSN? Real Soon Now. Wayne's way of saying, "I'm not sure when."

My worry wasn't Sventek. It was the unbalanced accounts. Why were the two accountin systems keeping different times? And why was some activity logged in one file witho showing up in the other?

Back to the accounting system for an afternoon. I found that the five minute tim difference between the time stamps came from our various computers' clocks drifting ov the months. One of our computer's clocks lost a few seconds every day.

But all of Sventek's activities should have appeared in both tallies. Was this related to la
week's accounting problem? Had I screwed things up when I poked around last week? Or wa
there some other explanation?

2

That afternoon, I sat through an impressively boring lecture on the structure of galaxies. The learned professor not only spoke in a monotone, but filled the chalkboard with a snake's nest of mathematical equations.

Trying to stay awake, I tossed around the problems I'd bumped into. Someone screwed u when adding a new account. A week later, Sventek logs in and tries to break into som computer in Maryland. The accounting record for that event seems garbled. Sventek unavailable. Something's amiss. It's almost as if someone's avoiding the accounting program.

What would it take, I wondered, to use our computers for free? Could someone have foun a way around our accounting system?

Big computers have two types of software: user programs and systems software. Program that you write or install yourself are user programs—for example, my astronomy routine which analyze a planet's atmosphere.

Alone, user programs can't do much. They don't talk directly to the computer; rather, the call upon the operating system to manipulate the computer. When my astronomy progra wants to write something, it doesn't just slap a word on my screen. Instead, it passes th word to the operating system, which, in turn, tells the hardware to write a word.

The operating system, along with the editors, software libraries, and language interpreter make up the systems software. You don't write these programs—they come with th computer. Once they're set up, nobody should tamper with them.

The accounting program is systems software. To modify or bypass it, you have to either b system manager, or somehow have acquired a privileged position within the operatin system.

OK, how do you become privileged? The obvious way is to log onto our computer with th system manager's password. We hadn't changed our password in months, but nobody woul have leaked it. And an outsider would never guess our secret password, "wyvern"—ho many people would think of a mythological winged dragon when guessing our password?

But even if you became system manager, you wouldn't fool with the accounting softwar It's too obscure, too poorly documented. Anyway, I'd seen that it worked.

Wait—our home-brew software worked properly. Someone had added a new accou without using it. Perhaps they didn't know about it. If someone had come in from the col they'd be unaware of our local wrinkles. Our system managers and operators knew this. Jo Sventek, even in England, surely would know.

But what about someone from the outside—a hacker?

The word hacker has two very different meanings. The people I knew who calle themselves hackers were software wizards who managed to creatively program their way ou of tight corners. They knew all the nooks and crannies of the operating system. Not du software engineers who put in forty hours a week, but creative programmers who can't leav the computer until the machine's satisfied. A hacker identifies with the computer, knowing like a friend.

Astronomers saw me that way. "Cliff, he's not much of an astronomer, but what computer hacker!" (The computer folks, of course, had a different view: "Cliff's not much o a programmer, but what an astronomer!" At best, graduate school had taught me to kee both sides fooled.)

But in common usage, a hacker is someone who breaks into computers.* In 1982, after group of students used terminals, modems, and long distance telephone lines to break in computers in Los Alamos and the Columbia Medical Center, the computing people sudden became aware of the vulnerability of our networked systems.

Every few months, I'd hear a rumor about someone else's system being invaded; usual this was at universities, and it was often blamed on students or teenagers. "Brilliant hig school student cracks into top security computer center." Usually it was harmless and writte off as some hacker's prank.

Could the movie *War Games* actually happen—might some teenage hacker break into Pentagon computer and start a war?

I doubted it. Sure, it's easy to muck around computers at universities where no securi was needed. After all, colleges seldom even lock the doors to their buildings. I imagined th military computers were a whole different story—they'd be as tightly secured as a militar base. And even if you did get into a military computer, it's absurd to think you could start war. Those things just aren't controlled by computers, I thought.

Our computers at Lawrence Berkeley Laboratory weren't especially secure, but we wer required to keep outsiders away from them and make an effort to prevent their misuse. W weren't worried about someone hurting our computers, we just wanted to keep our fundi agency, the Department of Energy, off our backs. If they wanted our computers painte green, then we'd order paintbrushes.

But to make visiting scientists happy, we had several computer accounts for guests. With a account name of "guest" and a password of "guest," anyone could use the system to solv their problems, as long as they didn't use more than a few dollars of computing time. hacker would have an easy time breaking into that account—it was wide open. This wou hardly be much of a break-in, with time limited to one minute. But from that account, yo could look around the system, read any public files, and see who was logged in. We felt th minor security risk was well worth the convenience.

Mulling over the situation, I kept doubting that a hacker was fooling around in my syste Nobody's interested in particle physics. Hell, most of our scientists would be delighted anyone would read their papers. There's nothing special here to tempt a hacker—no snazz supercomputer, no sexy trade secrets, no classified data. Indeed, the best part of working Lawrence Berkeley Labs was the open, academic atmosphere.

Fifty miles away, Lawrence Livermore Labs did classified work, developing nuclear bomb and Star Wars projects. Now, that might be a target for some hacker to break into. But wi no connections to the outside, Livermore's computers can't be dialed into. Their classifie data's protected by brute force: isolation.

If someone did break into our system, what could they accomplish? They could read an public files. Most of our scientists set their data this way, so their collaborators can read i Some of the systems software was public as well.

Though we call this data public, an outsider shouldn't wander through it. Some of it proprietary or copyrighted, like our software libraries and word processing programs. Oth databases aren't for everyone's consumption—lists of our employees' addresses an incomplete reports on work in progress. Still, these hardly qualify as sensitive material, an it's a long way from classified.

No, I wasn't worried about someone entering our computer as a guest and walking off wit somebody's telephone number. My real concern centered on a much bigger problem: could stranger become a super-user?

To satisfy a hundred users at once, the computer's operating system splits the hardwa resources much as an apartment house splits a building into many apartments. Eac apartment works independently of the others. While one resident may be watching T another talks on the phone, and a third washes dishes. Utilities—electricity, phone servic and water—are supplied by the apartment complex. Every resident complains about slo service and the exorbitant rents.

Within the computer, one user might be solving a math problem, another sendi electronic mail to Toronto, yet a third writing a letter. The computer utilities are supplied b the systems software and operating system; each user grumbles about the unreliab software, obscure documentation, and the exorbitant costs.

Privacy within the apartment house is regulated by locks and keys. One resident can't ent another's apartment without a key, and (if the walls are sturdy), one resident's activity wo bother another. Within the computer, it's the operating system that ensures user privacy. Yo can't get into someone's area without the right password, and (if the operating system is fa about handing out resources), one user's programs won't interfere with another's.

But apartment walls are never sturdy enough, and my neighbor's parties thunder into m bedroom. And my computer still slows down when there's more than one hundred peop using it at one time. So our apartment houses need superintendents, and our computers nee system managers, or super-users.

With a passkey, the apartment house superintendent can enter any room. From a privilege account, the system manager can read or modify any program or data on the compute Privileged users bypass the operating system protections and have the full run of th computer. They need this power to maintain the systems software ("Fix the editor!"), to tu the operating system's performance ("Things are too slow today!"), and to let people use th computer ("Hey, give Barbara an account.")

Privileged users learn to tread lightly. They can't do much damage if they're on privileged to read files. But the super-user's license lets you change any part of the system- there's no protections against the super-user's mistakes.

Truly, the super-user is all-powerful: he controls the horizontal, he controls the vertica When daylight savings time comes around, she resets the system clock. A new disk drive He's the only one who can graft the necessary software into the system. Different operati systems have various names for privileged accounts—super-user, root, system manager—b these accounts must always be jealously guarded against outsiders.

What if an outside hacker became privileged on our system? For one thing, he could ad new user accounts.

A hacker with super-user privileges would hold the computer hostage. With the master ke to our system, he could shut it down whenever he wishes, and could make the system unreliable as he wishes. He could read, write, or modify any information in the computer. N user's file would be protected from him when he operates from this privileged high groun The system files, too, would be at his disposal—he could read electronic mail before it delivered.

He could even modify the accounting files to erase his own tracks.

The lecturer on galactic structure droned on about gravitational waves. I was sudden awake, aware of what was happening in our computer. I waited around for the questio period, asked one token question, then grabbed my bike and started up the hill to Lawrenc Berkeley Labs.

A super-user hacker. Someone breaks into our system, finds the master keys, grants himse privileges, and becomes a super-user hacker. Who? How? From where? And, mostly, why?

---

* What word describes someone who breaks into computers? Old style software wizards are proud to be called hackers, a resent the scofflaws who have appropriated the word. On the networks, wizards refer to these hoodlums of our electronic a as "crackers" or "cyberpunks." In the Netherlands, there's the term "computervredebreuk"—literally, computer pea disturbance. Me? The idea of a vandal breaking into my computer makes me think of words like "varmint," "reprobate," a "swine."

3

It's only a quarter mile from the University of California to Lawrence Berkeley Labs, but Cyclotron Road is steep enough to make it a fifteen-minute bike ride. The old ten-speed didn't quite have a low enough gear, so my knees felt the last few hundred feet. Our computer center's nestled between three particle accelerators: the 184-inch cyclotron, where Ernest Lawrence first purified a milligram of fissionable uranium; the Bevatron, where the anti-proton was discovered; and the Hilac, the birthplace of a half-dozen new elements.

Today, these accelerators are obsolete—their mega-electron volt energies long surpassed b giga-electron volt particle colliders. They're no longer winning Nobel prizes, but physicis and graduate students still wait six months for time on an accelerator beamline. After all, ou accelerators are fine for studying exotic nuclear particles and searching out new forms matter, with esoteric names like quark-gluon plasmas or pion condensates. And when th physicists aren't using them, the beams are used for biomedical research, including canc therapy.

Back in the heyday of World War II's Manhattan project, Lawrence's cyclotron was th only way to measure the cross sections of nuclear reactions and uranium atoms. Naturall the lab was shrouded in secrecy; it served as the model for building atomic bomb plants.

During the 1950s, Lawrence Berkeley Laboratory's research remained classified, unt Edward Teller formed the Lawrence Livermore Laboratory an hour's drive away. All th classified work went to Livermore, while the unclassified science remained in Berkeley.

Perhaps to spread confusion, both laboratories are named after California's first Nob Laureate, both are centers for atomic physics, and both are funded by the Atomic Energ Commission's offspring, the Department of Energy. That's about the end of the similarity.

I needed no security clearance to work in the Berkeley Lab—there's no classified researc not a military contract in sight. Livermore, on the other hand, is a center for designin nuclear bombs and Star Wars laser beams. Hardly the place for a long-haired ex-hippie. Whi my Berkeley Lab survived on meager scientific grants and unreliable university fundin Livermore constantly expanded. Ever since Teller designed the H-bomb, Livermore classified research has never been short of funds.

Berkeley no longer has huge military contracts, yet openness has its rewards. As pu scientists, we're encouraged to research any curious phenomena, and can always publish ou results. Our accelerators might be peashooters compared to the behemoths at CERN Switzerland, or Fermilab in Illinois; still, they generate huge amounts of data, and we ru some respectable computers to analyze it. In fact, it's a source of local pride to find physicis recording their data at other accelerators, then visiting LBL to analyze their results on ou computers.

In raw number-crunching power, Livermore's computers dwarfed ours. They regular bought the biggest, fastest, and most expensive Crays. They need 'em to figure out wh happens in the first few nanoseconds of a thermonuclear explosion.

Because of their classified research, most of Livermore's computers are isolated. Of cours they have some unclassified systems too, doing ordinary science. But for their secret work- well, it's not for ordinary mortal eyes. These classified computers have no connections to th outside world.

It's just as impossible to import data into Livermore from the outside. Someone designin

nuclear bomb triggers using Livermore's classified computers has to visit the lab in perso bringing his data in on magnetic tape. He can't use the dozens of networks crossing th country, and can't log in from home, to see how his program is running. Since the computers are often the first ones off the production line, Livermore usually has to writ their own operating systems, forming a bizarre software ecology, unseen outside of the laboratory. Such are the costs of living in a classified world.

While we didn't have the number-crunching power of Livermore, our computers were n slouches. Our Vax computers were speedy, easy to use, and popular among physicists. W didn't have to invent our own operating systems, since we bought Digital's VMS operatin system, and grabbed Unix from campus. As an open lab, our computers could be networke anywhere, and we supported scientists from around the world. When problems developed i the middle of the night, I just dialed the LBL computer from my home—no need to bicyc into work when a phone call might solve it.

But there I was, bicycling up to work, wondering if some hacker was in our system. Th just might explain some of my accounting problems. If some outsider had picked the locks o our Unix operating system and acquired super-user privileges, he'd have the power t selectively erase the accounting records. And, worse, he could use our network connections t attack other computers.

I ducked my bike into a corner and jogged over to the cubicle maze. By now it was we past five, and the ordinary folks were at home. How could I tell if someone was hackin inside our system? Well, we could just send an electronic mail message to the suspiciou account, saying something like, "Hey, are you the real Joe Sventek?" Or we could disab Joe's account, and see if our troubles ended.

My thoughts about the hacker were sidetracked when I found a note in my office: th astronomy group needed to know how the quality of the telescope's images degraded if the loosened the specifications for the mirrors. This meant an evening of model building, a inside the computer. I wasn't officially working for them anymore, but blood's thicker tha water … by midnight, I'd plotted the graphs for them.

The next morning, I eagerly explained my suspicions about a hacker to Dave Clevelan "I'll bet you cookies to doughnuts it's a hacker."

Dave sat back, closed his eyes, and whispered, "Yep, cookies for sure."

His mental acrobatics were almost palpable. Dave managed his Unix system with a lai back style. Since he competed for scientists with the VMS systems, he had never screwe down the security bolts on his system, figuring that the physicists would object and take the business elsewhere. By trusting his users, he ran an open system and devoted his time t improving their software, instead of building locks.

Was someone betraying his trust?

Marv Atchley was my new boss. Quiet and sensitive, Marv ran a loose group that someho managed to keep the computers running. Marv stood in contrast to our division head, Ro Kerth. At fifty-five, Roy looked like Rodney Dangerfield as a college professor. He did physi in the grand style of Lawrence Laboratory, bouncing protons and antiprotons togethe looking at the jetsam from these collisions.

Roy treated his students and staff much as his subatomic particles: keep them in lin energize them, then shoot them into immovable objects. His research demanded heav

number crunching, since his lab generated millions of events each time the accelerator wa
turned on. Years of delays and excuses had soured him on computer professionals, so when
knocked on his door, I made sure we talked about relativistic physics and ignored computing

Now, Dave and I could guess Roy's reaction to our problem: "Why the hell did you leav
our doors wide open?"

Our boss's reaction might be predictable, but how should we react? Dave's first thoug
was to disable the suspect account and forget about it. I felt we ought to send a nastygram t
whoever was breaking in, telling him to stay away or we'd call his parents. After all,
someone was breaking in, it was bound to be some student from down on campus.

But we weren't certain that someone was breaking into our system. It might explain som
of our accounting problems—someone learns the system manager's password, connects to ou
machine, creates a new account, and tampers with the accounting system. But why woul
someone use a new account if they already had access to the system manager account?

Our boss never wanted to hear bad news, but we swallowed hard and called a lunchtim
meeting. We had no clear proof of a hacker, just circumstantial pointers, extrapolated fro
trivial accounting errors. If there was a break-in, we didn't know how far it extended, no
who was doing it. Roy Kerth blasted us. "Why are you wasting my time? You don't kno
anything and you haven't proven a whit. Go back and find out. Show me proof."

So how do you find a hacker? I figured it was simple: just watch for anyone using Sventek
accounts, and try to trace their connection.

I spent Thursday watching people log into the computer. I wrote a program to beep m
terminal whenever someone connected to the Unix computer. I couldn't see what each use
was doing, but I could see their names. Every couple minutes my terminal beeped, and I'd se
who had logged in. A few were friends, astronomers working on research papers or gradua
students plugging away on dissertations. Most accounts belonged to strangers, and
wondered how I could tell which connection might be a hacker.

At 12:33 on Thursday afternoon, Sventek logged in. I felt a rush of adrenaline and then
complete letdown when he disappeared within a minute. Where was he? The only pointer le
for me was the identifier of his terminal: he had used terminal port tt23.

Sitting behind a computer terminal, fingers resting on his keyboard, someone wa
connecting into our lab. My Unix computer gave him the address of port tt23.

Well, that's a start. My problem was to figure out which physical wires corresponded to th
logical name tt23.

Terminals from our laboratory and modems from dial-in telephones are all assigned "t
labels, while network connections show up as "nt." I figured that the guy must be either fro
our laboratory or dialing in on a phone line over a modem.

For a few seconds, I'd sensed a hesitant feeler into our computer. Theoretically, it must b
possible to trace the path from computer to human. Someone must be at the far end of th
connection.

It would take six months to track that path, but my first step was to trace the connectio
out of the building. I suspected a dial-in modem, connected from some telephone line, but
conceivably might be someone at the laboratory. Over the years, well over five hundre
terminals had been wired in, and only Paul Murray kept track. With luck, our homegrow
hardware connections were documented better than the home-brew accounting software.

Paul's a reclusive hardware technician who hides in thickets of telephone wire. I found hi behind a panel of electronics, connecting some particle detector to the lab-wide ethern system. Ethernets are electronic pipelines connecting hundreds of small computers. A fe miles of orange ethernet cable snaked through our lab, and Paul knew every inch of it.

Cursing me for surprising him in the middle of soldering a wire, he refused to give me ar help until I proved that I had a legitimate need to know. Aw, hell. Hardware technician don't understand software problems, and software jockeys know nothing about hardware.

Years of ham radio had taught me to solder, so Paul and I had at least one commo denominator. I picked up his spare soldering iron and earned his grudging respect after a fe minutes of burning my fingers and squinting. Finally, he disentangled himself from th ethernet cables and showed me around the LBL communications switchyard.

In this roomful of wires, the telephones, intercoms, radios, and computers were a interconnected by a tangle of cables, wires, optical fibers, and patch panels. The suspiciou port tt23 entered this room and a secondary computer switched it to one of a thousan possible terminals. Anyone dialing into the lab would be randomly assigned to a Unix por The next time I saw a suspicious character, I'd have to run over to the switchyard and unwin the connection by probing the switching computer. If he disappeared before I disentangle the connection, well, tough. And even if I did succeed, I'd only be able to point to a pair o wires entering the laboratory. I'd still be a long way from the hacker.

By lucky accident, though, the noontime connection had left some footprints behind. Pa had been collecting statistics on how many people used the switchyard. By chance he ha recorded the port numbers of each connection for the past month. Since I knew the tim when Sventek was active on port tt23, we could figure out where he came from. The printo of the statistics showed a one-minute 1200-baud connection had taken place at 12:33.

1200 baud, huh? That says something. The baud rate measures the speed that data flow through a line. And 1,200 baud means 120 characters per second—a few pages of text ever minute.

Dial-up modems over telephone lines run at 1200 baud. Any lab employee here on the hi would run at high speed: 9600 or 19,200 baud. Only someone calling through a mode would let their data dribble out a 1200-baud soda straw. And the anonymity and convenienc of these dial-in lines are most inviting to strangers. So pieces were beginning to fit together. couldn't prove that we had a hacker in the system, but someone dialed into our lab and use Sventek's account.

Still, the 1200-baud connection was hardly proof that a hacker entered our system. A incomplete trace, especially one that went no farther than my building, would never convinc my boss that something was up, something weird. I needed to find incontrovertible evidenc of a hacker. But how?

Roy Kerth had shown me the high-energy particle detectors attached to the Bevatron: the find jillions of subatomic interactions, and 99.99 percent are explainable by the laws o physics. Spending your time exploring each particle trail will lead you to conclude that all th particles obey known physics, and there's nothing left to discover. Alternatively, you cou throw away all the explainable interactions, and only worry about those that don't qui satisfy the canonical rules.

Astronomers, distant cousins of high-energy physicists, work along similar lines. Most sta

are boring. Advances come from studying the weirdies—the quasars, the pulsars, th gravitational lenses—that don't seem to fit into the models that you've grown up wit Knowing cratering statistics on the planet Mercury tells you how often the planet wa bombarded in the early solar system. But study the few craters intersected by scarps an ridges and you'll learn how the planet shrank as it cooled during its first billion years. Colle raw data and throw away the expected. What remains challenges your theories.

Well, let's apply this way of thinking to watching someone visiting my computer. I've got terminal on my desk, and can borrow a couple others. Suppose I just watched the traff coming into the computer center. There's about five hundred lines entering the system. Mo of these lines run at 9600 baud, or around one hundred fifty words per second. If half th lines are used at any time, I'd have to read well over ten thousand pages every minute. Righ No way could I monitor that kind of traffic on my terminal.

But the high speed lines come from people at LBL. We'd already traced one suspiciou connection to a 1200-baud line. There are fewer of them (we can't afford too many incomin phone lines), and they're slower. Fifty lines at 1200 baud might generate a hundred pages minute, still far too fast to watch on the screen of my terminal. I might not be able to watc fifty people running at once, but maybe I could print out all their interactive sessions, an read the piles of paper at my leisure. A paper printout would provide hard proof of someon messing around; if we found nothing suspicious, we could drop the whole project.

I'd record everything that happened during each 1200-baud connection. This would b technically challenging—since I didn't know which line the hacker was calling, I'd have t monitor four dozen. More worrisome was the ethical problem of monitoring ou communications. Did we have the right to watch the traffic running through our lines?

My sweetheart, Martha, was just finishing law school. Over a deep-dish pizza, we talke about the implications of someone breaking into a computer. I wondered how much troub I'd be in by listening to incoming traffic.

"Look," she mumbled, burning the roof of her mouth on the vulcanized mozzarella. "You' not the government, so you don't need a search warrant. The worst it would be is invasion o privacy. And people dialing up a computer probably have no right to insist that the system owner not look over their shoulder. So I don't see why you can't."

So with a clear conscience, I started building a monitoring system. We had fifty 1200-bau lines, and a hacker might be using any of them. I had no equipment designed to record th traffic.

But there's an easy way to record a hacker's activity. Modify the Unix operating system s that whenever a suspicious person logged in, the system records all the keystrokes. This wa tempting, because I only had to add some lines of code to the Unix daemon software.

The daemons themselves are just programs that copy data from the outside world into th operating system—the eyes and ears of Unix. (The ancient Greek daemons were inferio divinities, midway between gods and men. In that sense, my daemons are midway betwee the god-like operating system and the world of terminals and disks.)

I could split the daemon's output like a T-joint in a pipe, so the hacker's keystrokes woul simultaneously go to both the operating system and a printer. Software solutions are simp and elegant.

"Muck with the daemons at your own risk," Dave Cleveland said. "Just respect their timi

needs."

Wayne also warned me, "Look, if you goof up, you'll break the system for sure. It will tu the system into molasses, and there's no way you'll follow everything that happens. Just wa till you see the system console print out 'Panic kernel mode interrupt'—don't come crying o my shoulder!"

Dave chipped in, "Hey, if your hacker has any Unix experience, he's bound to notice change in the daemons."

That convinced me. A sharp systems person would notice that we'd changed the operati system. The moment the hacker knew someone was watching him, he'd trash our databas and scram. Our wiretaps had to be completely undetectable, even to an omnipotent supe user. Silent, invisible monitors to trap the hacker's activity.

Maybe just tape recording the telephone lines would work, but tape recorders didn't fe right, too much of a kludge. We'd have to play them back, and couldn't watch the keystrok until long after a hacker had disconnected. Finally, where would I find fifty tape recorders?

About the only other place to watch our traffic was in between the modems and th computers. The modems converted the tones of a telephone into electronic pulses, palatab to our computers and the daemons in their operating systems. These modem lines appeare as flat, twenty-five conductor wires, snaking underneath the switchyard's false floor. printer or personal computer could be wired to each of these lines, recording every keystrok that came through.

A kludge? Yes. Workable? Maybe.

All we'd need are fifty teletypes, printers, and portable computers. The first few were eas to get—just ask at the lab's supplies desk. Dave, Wayne, and the rest of the systems grou grudgingly lent their portable terminals. By late Friday afternoon, we'd hooked up a doze monitors down in the switchyard. The other thirty or forty monitors would show up after th laboratory was deserted. I walked from office to office, liberating personal computers fro secretaries' desks. There'd be hell to pay on Monday, but it's easier to give an apology tha get permission.

Strewn with four dozen obsolete teletypes and portable terminals, the floor looked like computer engineer's nightmare. I slept in the middle, nursing the printers and computer Each was grabbing data from a different line, and whenever someone dialed our system, I wake up to the chatter of typing. Every half hour, one of the monitors would run out paper or disk space, so I'd have to roll over and reload.

Saturday morning, Roy Kerth shook me awake. "Well, where's your hacker?"

Still in my sleeping bag, I must have smelled like a goat. I blinked stupidly and mumble something about looking at the fifty piles of paper.

He snorted, "Well, before you start poking around those printouts, return those printer You've been running around here like a maniac swiping equipment used by people who a getting work done. You've pissed off a dozen astronomers. Are *you* getting work done? N Whaddya think this place is, your own personal sandbox?"

Bleary-eyed, I dragged each printer back to its rightful owner. The first forty-nine showe nothing interesting. From the fiftieth trailed eighty feet of printout. During the nigh someone had sneaked in through a hole in the operating system.

4

For three hours, a hacker had strolled through my system, reading whatever he wished. Unknown to him, my 1200-baud Decwriter had saved his session on eighty feet of single-spaced computer paper. Here was every command he issued, every typing mistake, an every response from the computer.

This printer monitored the line from Tymnet. I didn't realize it, but a few of our 1200-bau lines weren't dial-in modem lines. Rather, they came from Tymnet, a communicatio company that interconnected computers around the world.

Back before divestment, the Bell system monopolized communications. AT&T was the on way to connect New York to Chicago. By using modems, the phone system could handle dat but the noise and expense of the long distance service made it unsuitable for computers. B the late '70s, a few other companies dipped their toes in the water, offering specialize services like data phones. Tymnet created a network to interconnect computers in majo cities.

Tymnet's idea was simple and elegant: create a digital communications backbone, l anyone connect to the backbone by making a local telephone call, then send the data to an computer on the network. Tymnet would compress dozens of users' data into a few packet and economically send these around the country. The system was immune to noise, and eac user could run as fast as he wished. Customers saved money because they could access distant computer by making a local call.

To satisfy scientists around the country, LBL subscribed to Tymnet. When a researcher Stonybrook, New York, wanted to connect to our computer, he dialed his local Tymn number. Once his modem was connected to Tymnet, he just asked for LBL and worked as he were in Berkeley. Physicists from far away loved the service, and we were delighted t find them spending their research dollars on our computers, rather than their home machine

Someone was breaking in, using the Tymnet line. Since Tymnet interconnected the who country, our hacker might be anywhere.

For the moment, though, I was fascinated not by where the hacker came from, but what h had done in three hours. My guess was right: Sventek's account was being used to break in our Unix computer.

Not just break in. This hacker was a super-user.

The hacker had sneaked through a hole in our system to become a super-user—he'd neve even logged into the system manager's account. He was like a cuckoo bird.

The cuckoo lays her eggs in other birds' nests. She is a nesting parasite: some other bir will raise her young cuckoos. The survival of cuckoo chicks depends on the ignorance of oth species.

Our mysterious visitor laid an egg-program into our computer, letting the system hatch and feed it privileges.

That morning, the hacker wrote a short program to grab privileges. Normally, Unix wor allow such a program to run, since it never gives privileges beyond what a user is assigne But run this program from a privileged account, and he'll become privileged. His proble was to masquerade this special program—the cuckoo's egg—so that it would be hatched b the system.

Every five minutes, the Unix system executes its own program named *atrun*. In turn, atru

schedules other jobs and does routine housecleaning tasks. It runs in a privileged mode, wi the full power and trust of the operating system behind it. Were a bogus atrun progra substituted, it would be executed within five minutes, with full system privileges. For th reason, atrun sits in a protected area of the system, available only to the system manage Nobody but the system manager has the license to tamper with atrun.

Here was the Cuckoo's nest: for five minutes, he would swap his egg for the system's atru program.

For this attack, he needed to find a way to move his egg-program into the protecte systems nest. The operating system's barriers are built specifically to prevent this. Norm copy programs can't bypass them; you can't issue a command to "copy my program in systems space."

But there was a wildcard that we'd never noticed. Richard Stallman, a free-lance comput programmer, loudly proclaimed that information should be free. His software, which he giv away for free, is brilliantly conceived, elegantly written, and addictive.

Over the past decade Stallman created a powerful editing program called Gnu-Emacs. B Gnu's much more than just a text editor. It's easy to customize to your personal preference It's a foundation upon which other programs can be built. It even has its own mail facili built in. Naturally, our physicists demanded Gnu; with an eye to selling more computi cycles, we installed it happily.

Just one problem: there's a bug in that software.

In the way it was installed on our Unix computer, the Gnu-Emacs editor lets you forward mail file from your own directory to anyone else in an unusual way. It doesn't check to se who's receiving it, or even whether they want the file. It just renames the file and changes i ownership label. You've just transferred ownership of the file from you to me.

No problem to send a file from your area to mine. But you'd better not be able to move file into the protected systems area: only the system manager is allowed there. Stallman software had better make sure this can't happen.

Gnu didn't check. It let anyone move a file into protected systems space. The hacker kne this; we didn't.

The hacker used Gnu to swap his special *atrun* file for the system's legitimate version. Fiv minutes later, the system hatched his egg, and he held the keys to my computer.

He had used this technique to fool the computer into giving him power. He planted h phony program where the system expected to find a valid one. The instant that Unix execute his bogus *atrun* program, he became super-user. The whole operation depended on his bei able to move a file anywhere he wished.

Gnu was the hole in our system's security. A subtle bug in an obscure section of som popular software. Installed blindly by our systems programmers, we'd never thought that might destroy our whole system's security.

Now I understood. Our friend must have entered a guest account, leveraged his privileg using Gnu's hole, and then added a new account to the computer's files.

In front of me, the first few feet of the printout showed the cuckoo preparing the nes laying the egg, and waiting for it to hatch. The next seventy feet showed the fledgling cucko testing its wings.

As super-user, he had the run of our system. First thing he did was erase his tracks: h

switched the good copy of *atrun* back where it belonged. Then he listed the electronic mail
all our users, reading news, gossip, and love letters. He learned of the past month's compute
changes, grant proposals, and new hires. He searched for changes in the system manager
files, and discovered that I'd just started work. He checked my salary and résumé. Mo
worrisome, he realized that I was a system manager, and my account name.

Why me? What did I do? At any rate, from now on, I'd better use a different name.

Every ten minutes, the hacker issued the command, "who," to list everyone logged onto th
computer. Apparently, he worried that someone might see him connected, or might b
watching. Later, he searched for any changes in the operating system—had I modified th
daemons to record his session, as I'd first planned to do, he would surely have discovered it.
felt like a kid playing hide-and-seek, when the seeker passes within inches of his hiding place

Within the first hour, he wrote a program to scan everyone's mail messages for an
mention of his activity. He searched for the word, "hacker," and "security."

One scientist had started a program that assembled data from an experiment over th
weekend. Running under the name "gather," this program innocuously collected informatio
every few minutes and wrote it to a file. The hacker saw this program, spent ten minute
trying to understand what it did, and killed it.

Yow! Here's someone looking over his shoulder every few minutes, checking to see
anyone's around. He kills any jobs that he thinks might monitor him. He opens my mai
checking to see if anyone's written about hackers. Wayne was right: if you stay in the ope
he'll know you're watching. From now on, we'd have to be subtle and invisible.

When he wasn't looking back over his shoulder, the hacker was reading files. By studyin
several scientists' command files and scripts, he discovered pathways into other la
computers. Every night, our computer automatically calls twenty others, to exchange ma
and network news. When the hacker read these phone numbers, he learned twenty ne
targets.

From the mail file of an engineer:

"Hi, Ed!

I'll be on vacation for the next couple weeks. If you need to get any of my data, just log into my account on the Vax computer. Account name is Wilson, password is Maryanne (that's my wife's name). Have fun!"

The hacker had fun, even if Ed didn't. He connected through our local area network int
that Vax, and had no problem logging into Wilson's account. Wilson wouldn't notice th
hacker reading his files, and probably wouldn't care. They contained numerical dat
meaningless to anyone but another nuclear physicist.

Our visitor knew about our lab's internal networks. Our dozen big computers were tied t
a hundred laboratory computers using ethernets, serial lines, and chewing gum. Whe
physicists wanted to get data from a computer at the cyclotron into our big compute
elegance meant nothing. They'd use any port, any line, any network. Over the yea
technicians had woven a web of cables around the lab, interconnecting most of the compute
with whatever seemed to work. This local area network reached into every office, connectin
PC's, Macintoshes, and terminals to our mainframe computers.

Often, these networked computers had been arranged to trust each other. If you're OK o