Mobile phone platforms have also acquired DRM mechanisms. The Open Mobile Alliance DRM version 1 supports the download of ringtones, while version 2 supports music and video download with more general mechanisms. Version 1 is widely used, but version 2 has been held up by an interesting dispute. The main OMA promoter, Nokia, would like to become the distribution channel of choice for mobile music, just as Apple has become for PCs with iTunes. However the network operators are extremely reluctant to let Nokia have a business relationship with their customers directly, and this has held up deployment.

In general, security is made more difficult by the long and complex supply chain that delivers mobile phone service to customers. IP companies like ARM own the chip designs; foundries such as Infineon make the chips; handset designers like Samsung manufacture the actual mobiles; Symbian provides the operating system; Nokia may provide some more software for a download interface; a network operator such as Vodafone provides the national infrastructure; and a local operating company then bills the customer. There has been a tendency for everyone in this chain to see security as a problem to be tossed over the fence to the companies on either side. On top of this there might be apps provided by games vendors, and corporate apps such as fleet management that are provided by third-party software houses. Add the next generation of location-based services and offerings from the likes of Google and eBay, and the whole thing becomes complex beyond belief.

One final aspect of mobile phone platforms is locking. In many countries, handsets are subsidised out of future call revenue: you get a cheap phone in return for a year's contract with a network. The downside is that your phone is locked to the network. Even some prepaid phones are locked, and carry an implicit subsidy from expected future token sales. However, in some countries, this business model isn't permitted. There has thus arisen a brisk trade in unlocking tools and services, some of which are devised by finding exploits in the phone software, and others using the kind of hardware reverse-engineering techniques I described in Chapter 16. Legal skirmishing between the phone companies and the unlocking services came to a head after Apple launched the iPhone, which had a twist on this business model: the network for which you could buy it were those that had paid Apple the most for the privilege. This annoyed iPhone purchasers as the U.S. network of choice, AT&T, was fairly slow. The iPhone was duly hacked, and AT&T sent its lawyers after the unlockers. It also shipped a software upgrade that disabled unlocked phones.

Laws on locking and unlocking vary widely. In the USA, the Digital Millennium Copyright Act (DMCA), which prohibits interference with any technical mechanism that enforces copyright, has a specific exemption for cellphone unlocking, in order that copyright law shouldn't be abused to stifle competition in the mobile phone market. It's now being argued that

this exemption covers only people who unlock their own phones, but doesn't extend to the sale of unlocking tools or services [821]. So in America, unlocking is legal if you're a customer (though the provider may brick your phone), and may be open to challenge if you do it commercially. At the other end of the scale, courts in France and Germany have held mobile phone locking to be illegal: Apple will have to offer unlocked models for sale there if it offers the product at all. Such variations in law and business practice have led to the development of a thriving grey market whereby phones are shipped from one country to another; this in turn has driven secondary markets for unlocking tools and even for assorted frauds.

20.3.5 So Was Mobile Security a Success or a Failure?

Whether mobile-phone security has been a success or a failure depends on whom you ask.

From the point of view of cryptography, it was a failure. Both the Comp128 hash function and the A5 encryption algorithm were broken once they became public. In fact, GSM is often cited as an object lesson in Kerckhoffs' Principle — that cryptographic security should reside in the choice of the key, rather than in the obscurity of the mechanism. The mechanism will leak sooner or later and it's better to subject it to public review before, rather than after, a hundred million units have been manufactured. (GSM security wasn't a disaster for most cryptographers, of course, as it provided plenty opportunities to write research papers.)

From the phone companies' point of view, GSM was a success. The shareholders of GSM operators such as Vodafone have made vast amounts of money, and a (small) part of this is due to the challenge-response mechanism in GSM stopping cloning. The crypto weaknesses were irrelevant as they were never exploited (at least not in ways that did significant harm to call revenue). There are one or two frauds that persist, such as the long conference call trick; but on balance the GSM design has been good to the phone companies.

From the criminals' point of view, GSM was also fine. It did not stop them stealing phone service: the modus operandi merely changed, with the cost falling on credit card companies or on individual victims of 'identity theft' or street robbery. It did not stop calls from anonymous phones; the rise of the prepaid phone industry made them even easier. (The phone companies were happy with both of these changes.) And of course GSM did nothing about theft through fraud.

From the point of view of the large-country intelligence agencies, GSM was fine. They have access to local and international traffic in the clear anyway, and the weakened version of A5 facilitates tactical signint against developing countries. And the second wave of GSM equipment is bringing some juicy features, such as remote control of handsets by the operator [1061]. If you can

subvert (or masquerade as) the operator, then there seems to be nothing to stop you quietly turning on a target's mobile phone without his knowledge and listening to the conversation in the room.

From the point of view of the police and low-resource intelligence agencies, things are not quite so bright. The problem isn't the added technical complexity of GSM networks: court-ordered wiretaps can be left to the phone company (although finding the number to tap can be a hassle if the suspect is mobile). The problem is the introduction of prepaid mobile phones. This not only decreases the signal to noise ratio of traffic analysis algorithms and makes it harder to target wiretaps, but also encourages crimes such as extortion and stalking.

From the customer's point of view, GSM was originally sold as being completely secure. Was this accurate? The encryption of the air link certainly did stop casual eavesdropping, which was an occasional nuisance with analog phones. (There had been some high-profile cases of celebrities being embarrassed, including one case in Britain where Prince Charles was overheard talking to his mistress Camilla Parker-Bowles before his divorce from Princess Diana, and one in the USA involving Newt Gingrich.) But almost all the phone tapping in the world is done by large intelligence agencies, to whom the encryption doesn't make much difference.

Things are even less positive for the subscriber when we look at billing. Cryptographic authentication of handsets can't stop the many frauds perpetrated by premium rate operators and phone companies. If anything it makes it harder to wriggle out of bogus charges, as the phone company can say in court that your smartcard and your PIN must have been used in the handset that made the call. The same will apply to 3rd generation phones. The one minor compensation is that GSM facilitated the spread of prepaid phones, which can limit the exposure.

So the security features designed into GSM don't help the subscriber much. They were designed to provide 'security' from the phone company's point of view: they dump much of the toll fraud risk, while not interrupting the flow of premium rate business — whether genuine or fraudulent.

In the medium term, the one ray of comfort for the poor subscriber is that the increasing complexity of both handsets and services may create regulatory pressure for transparent mechanisms that enable the customer to control the amount she's billed. There are a number of factors pushing for this, such as the growing vulnerability of platforms to malware; and a number of factors pushing in the other direction, such as the phone companies' desire to keep pricing opaque so that they can rip customers off. I mean this not just in the strict sense that phone companies often defraud their customers, but also in the colloquial sense that confusion pricing is a mainstay of phone company economics. I'll go into this in more detail in the next section, where I'll look at the economics of telecomms and how they relate to fraud and abuse.

20.3.6 VOIP

The latest development in telephone is voice over IP (VOIP), in which voice traffic is digitised, compressed and routed over the Internet. This had experimental beginnings in the 1970s; products started appearing in the 1990s; but decent call quality requires bandwidth of about 80 kbit/sec, more than can be got from a dial-up modem. As a result, VOIP only really took off once a critical mass of households had broadband connections. Since about 2005, it has become big business, with eBay purchasing Skype in 2006 for $2.6bn. In fact, most normal phone calls are digitized and sent over IP networks belonging to the phone companies, so in a technical sense almost all phone calls are 'VOIP', but in line with common usage I'll use the term only for those calls made by customers over an IP service that they access directly, via their ISP.

The VOIP market is still fragmented, and suffers many of the problems you'd expect. Most products were shipped quickly in a race to market, with security an afterthought at best. There was the usual crop of stack overflows and other vulnerabilities at the implementation level. The most popular VOIP protocol, the Session Initiation Protocol (SIP), turns out to have vulnerabilities that enable third parties to wrongly bill calls to subscribers [1375]. These could give rise to difficult disputes between VOIP services and their customers.

There are many issues with VOIP as the market shakes down. The leading system, Skype, is a closed system using proprietary protocols while its many competitors are fragmented. One business opportunity is 'click-to-call' whereby an advertisement could contain a VOIP link enabling a prospective customer to click to speak to a representative; early efforts have run up against compatibility problems. Technical problems range from differing protocols through how to deal with the jitter (time delay variation) caused by the variable quality of service on the Internet, and how to deal with network address translation are firewalls.

The interaction with security is complex. Corporate security policies can result in firewalls refusing to pass VOIP traffic. Phone calls can be more secure if made over VOIP, as encryption is easy to add and with some services (such as Skype) it comes turned on by default. For some time at least, it may be more difficult for police and intelligence services to get access to call contents and to records of who called whom; many services have not yet got round to engineering a law-enforcement interface, and in the case of Skype, its peer-to-peer nature might make that difficult. The FBI is currently pushing for the CALEA regulations to be applied rigorously to VOIP providers, while the industry and civil liberties groups are resisting. (I'll have more to say about this in Part III.) Wiretaps are not the only issue; click-to-call clearly will have other security, privacy and safety issues in the context of social networking sites, especially

A more high-profile regulatory issue is that governments want emergency calls made through VOIP services to work reliably, and provide information about the location of the caller. This is hard; an IP packet stream can be coming from anywhere, and no-one owns enough of the Internet to guarantee quality of service. At a deeper level than that, the dispute over dependability is just the latest tussle in the forty years' war between computer companies and phone companies, which I've alluded to from time to time. Computer companies are innovative and entrepreneurial, racing to market with products that are just about good enough; phone companies are slow-moving and heavily regulated, with a fifteen-year product cycle and services engineered for 99.999% availability. Ever since people started using modems on a large scale in the 1960s, there has been one battle after another — which the computer companies pretty well always win. A glimpse into the future may have been provided by a two-day Skype outage in August 2007, caused by poorly-engineered protocols. After a large number of computers rebooted following Microsoft's 'patch Tuesday' security update, tens of millions of Skype clients simultaneously tried to contact the network. The congestion caused the network to fail, yet the clients kept on trying. We can probably expect a lot more failures. Although a VOIP handset looks like a phone and works like a phone, it's more like email in terms of reliability. If the power goes off, so does your service.

The main problems beyond that have to do with the incompatibility of the phone companies' business model and the nature of the Internet. Phone companies make their money by charging you vastly different rates for different types of content: their typical rates work out at a hundredth of a cent per megabyte for cable TV, eight cents per megabyte for wireline phone, three dollars a megabyte for mobile phone, and a whopping three thousand dollars a megabyte for text messages. The opportunity exploited by VOIP is to arbitrage these, and it's hardly surprising that the phone companies do what they can to get in the way. ISPs who are also phone companies deliberately cause problems for VOIP to stop it competing with their phone service; this is particularly pronounced with mobile IP service. For these reasons, we'd better look at the economics of phone companies and the interaction with security.

20.4 Security Economics of Telecomms

Phone companies are classic examples of a business with extremely high fixed costs and very low marginal costs. Building a nationwide network costs billions and yet the cost of handling an additional phone call is essentially zero. As I discussed in Chapter 7 on Economics, this has a couple of implications.

First, there's a tendency towards dominant-firm markets in which the winner takes all. Indeed for many years telephone service was considered a

most countries to be a 'natural monopoly' and operated by the government; the main exception was the USA where the old AT&T system was heavily regulated. After the breakup of AT&T following an antitrust case, and Margaret Thatcher's privatisation of BT, the world moved to a different model, of regulated competition. The details vary from one country to another but, in general, some sectors (such as mobile phones) had a fixed number of allowed competitors; others (such as long-distance provision) were free for companies to compete in; and others (such as local loop provision) remained de facto monopolies but were regulated.

Second, because the marginal cost of service provision is zero, the competitive sectors (such as long-distance calling) saw prices drop quickly to a very low level — in many cases below the level needed to recoup the investment. (The large investments made during the dotcom bubble end up being treated as sunk costs.)

In such a market, firms will try to maintain price discrimination by whatever means they can. In many telecomms markets, the outcome is *confusion pricing* — products are continually churned, with new offerings giving generous introductory discounts to compete with the low-cost providers, but with rates sneakily raised afterwards. The effect is to discriminate between 'people who will spend money to save time, and people who will spend time to save money'. If you can be bothered to continually check prices, you can get really good deals, but often at the cost of indifferent service. If you don't have the time to keep scrutinising your phone bills, and the latest emails you get from your ISP advising you of the latest service changes, you can find that the call home you made from your mobile while on business in France just ate up all that month's savings. In the end, you can end up paying about the same amount per month that you did in the past. Andrew Odlyzko, a scholar of phone-company economics, suggests the eventual way forward will be fixed-price contracts: for a certain amount per month you'll get a home phone, ISP service, a couple of mobiles and a decent allowance of air minutes and texts [982]. In the meantime, telecomms pricing remains murky, contentious and far from transparent. This leads directly to abuse.

20.4.1 Frauds by Phone Companies

One of the steadily growing scams is the unscrupulous phone company that bills lots of small sums to unwitting users. It collects phone numbers in various ways. (For example, if you call an 800 number, then your own number will be passed to the far end regardless of whether you tried to block caller line ID.) The wicked phone company then bills you a few dollars. Your own phone company passes on this charge and you find there's no effective way to dispute it. Sometimes the scam uses a legal loophole: if you call an 800 number in the USA, the company may say 'Can we call you right back?' and if [illegible]

then you're deemed to have accepted the charges, which are likely to be at a high premium rate. The same can happen if you respond to voice prompts as the call progresses. These practices are known as *cramming*.

I was myself the victim on an attempt at cramming. On holiday in Barcelona, my wife's bag was snatched, so we called up and cancelled the phone that she'd had in it. Several months later, we got a demand from our mobile provider to pay a few tens of dollars roaming charges recently incurred by that SIM card in Spain. In all probability, the Spanish phone company was simply putting through a few charges to a number that they'd seen previously, in the knowledge that they'd usually get away with it. My mobile service provider initially insisted that even though I'd cancelled the number, I was still liable for calls billed to it months afterwards and had to pay up. I got out of the charges only because I'd met the company's CEO at an academic seminar and was able to get his private office to fix the problem. Customers without such access usually get the short end of the stick. Indeed, UK phone companies' response to complaints has been to offer its customers 'insurance' against fraudulent charges. That they can get away with this is a clear regulatory (and indeed policing) failure.

Another problem is *slamming* — the unauthorized change of a subscriber's long distance telephone service provider without their consent. The slammers tell your local phone company that you have opted for their service; your phone company routes your long distance calls through them; they hope you don't notice the change and dispute the bill; and the telephone charges can then be jacked up. Some local phone companies, such as Bell Atlantic, allow their customers to freeze their chosen long distance carrier [22].

It would be a mistake to assume that cramming and slamming are just done by small fly-by-night operators. AT&T is one of the worst offenders, having been fined $300,000 not only for slamming, but for actually using forged signatures of subscribers to make it look as if they had agreed to switch to their service. They got caught when they forged a signature of the deceased spouse of a subscriber in Texas [390]. As for the UK, slamming wasn't even made illegal until 2005.

Another problem is the fly-by-night phone company. As anyone in the USA is legally entitled to set up a phone company, it is straightforward to set one up, collect some cash from subscribers, and then vanish once the invoices for interconnect fees come in. Companies also advertise sex lines with normal phone numbers to trap the unwary, then send huge bills to the subscriber at his residential addresses and try to intimidate him into paying. The regulation of premium-rate providers and their business practices is a widespread problem.

And it's not just the small operators that indulge in sharp practice. An example that affects even some large phone companies is the short termination of international calls

Although premium-rate numbers are used for a number of more or less legitimate purposes such as software support, many of them exploit minors or people with compulsive behaviour disorders. So regulators have forced phone companies in many countries to offer premium-rate number blocking to subscribers. Phone companies get round this by disguising premium rate numbers as international ones. I mentioned scams with Caribbean numbers in section 20.2.1 above. Now many other phone companies from small countries with lax regulators have got into the act, and offer sex line operators a range of numbers on which they share the revenue.

Often a call made to a small-country phone company doesn't go anywhere near its ostensible destination. One of the hacks used to do this is called *short termination*, and here's an example that surfaced a few years ago. Normally calls for the small Pacific country of Tuvalu went via Telstra in Perth, Australia, where they were forwarded by satellite. However, the sex line numbers were marked as invalid in Telstra's system, so they were automatically sent via the second-choice operator — a company in New Zealand. (The girls — or to be more precise, the elderly retired hookers who pretend to be young girls — were actually in Manchester, England.) Technically, this is an interesting case of a fallback mechanism being used as an attack vehicle. Legally, it is hard to challenge as there is an international agreement (the Nairobi Convention) that stops phone companies selectively blocking international destinations. So if you want to stop your kids phoning the sex line in Tuvalu, you have to block all international calls, which makes it harder for you to phone that important client in Germany.

Problems like these are ultimately regulatory failures, and they are increasingly common. (For example, in the Moldova scam I mentioned earlier, the calls didn't go to Moldova but to Canada [251].) They are continuing to get worse as technology makes new, complex services possible and the regulators fail to keep up.

20.4.2 Billing Mechanisms

Billing mechanisms are a growing source of problems, as the economic forces discussed above lead to ever-more-complex rate cards. Even the phone companies themselves sometimes fall foul of the growing complexity. Sometimes their rates get so complex that people can arbitrage against them; there has been more than one case in which an entrepreneur found he could set up an international premium-rate service and be paid more per minute for calling it, than it cost him to call it using the best discount rate. Phone companies have tried to recover such trading profits through the courts, claiming fraud — with mixed success.

The security of the billing mechanisms covers a much wider range of issues. Present arrangements are inadequate for a number of reasons

- A *call detail record* (CDR) is only generated once the calling phone goes on-hook. This was a long-established feature of wireline networks, but once the environment changed to mobile it became a serious problem. As I mentioned above, someone running a call-sell operation can set up a long conference call using a stolen or prepaid mobile, which clients join and leave one after the other. The phone stays off-hook continuously for hours. As soon as it goes on-hook, a CDR for several thousand dollars is generated, and the alarm goes off. The operator throws the phone in the river and starts using the next one. By 1996, this had become so serious that Vodafone introduced a six hour limit on all mobile calls. But it won't be acceptable to just drop all 3gpp calls after six hours. Many users are expected to have always-on internet connections (such as from their laptops) with relatively light packet traffic most of the time.
- More seriously, the back-end accounting system was designed in the days when phone companies were sleepy government departments or national monopolies, and there were no premium-rate services through which real money could be extracted from the system. So it has little in the way of audit and non-repudiation. In effect, phone company A tells phone company B, 'please debit your customer no. X the sum of $Y and send me the money' — and it does. Even when these debits are mistaken or fraudulent, phone company B has no incentive to quibble, as it gets a cut. The result, as we saw with the cramming cases above, is that fraud slowly rises as insiders abuse the system. This is no longer fit for purpose in a world with many phone companies, quite a few of which are unscrupulous. The regulators aren't effective, and the only real backward pressure comes from the growing number of prepay customers.
- The phone companies also want to be able to charge for relatively high-value product and service delivery, extending the current premium services through location-based services ('give me a map showing me how to drive to the nearest McDonalds') to music and video downloads and extra services such as the Finnish ferry tickets, cans of coke from vending machines, and (most recently) parking meters in London. The accounting system will have to become a lot more robust, and dispute resolution mechanisms fairer and more transparent, if this potential is to be realised.
- All this interacts with platform security. If malware becomes widespread on mobile phones, then the botnet herders who control subverted phones will be able to pay for all sorts of goods and services by getting infected machines to send text messages. Recent history suggests that any exploit that can be industrialised to make money on a large scale, will be.

So how can phone payment systems be improved?

One proposed way of implementing this is to incorporate a micropayment mechanism [92]. The idea is that the phone will send regular *tick payments* to each of the networks or service providers which are due to be paid for the call. The tick payments can be thought of as electronic coins and are cryptographically protected against forgery. At the start of the call, the handset will compute a number of phone ticks by repeated hashing: $t_1 = h(t_0)$, $t_2 = h(t_1)$, and so on, with t_k (for some credit limit k, typically 2^{10} units) being signed by the phone company. The phone will then release ticks regularly in order to pay for the services as the call progresses. It starts by releasing t_k, then t_{k-1}, then t_{k-2} and so on. If a charge is subsequently disputed — whether by a subscriber or a network operator — the party claiming an amount of (say) j ticks must exhibit the ticks t_{k-j} and t_k, the latter with a certificate. As the hash function h is one-way, this should be hard to do unless the handset actually released that many ticks. The tick t_{k-j} can how be checked by applying the hash function to it j times and verifying that the result is t_k[4]. One advantage of tick payments is that as well as protecting the phone companies from conference call frauds, it could protect the subscriber from many more. It could enable phone application designers to empower users in new ways: for example, you might by default decide to accept calls, or play games, or do other functions, only so long as they cost less than a dollar, and require user intervention for anything more expensive. At present, that kind of functionality just isn't available, except via the rather clunky mechanism of only loading so much airtime into your phone.

The industry's proposed solution is to redesign the call data record to contain a lot more information. In addition to time, duration and called number, it will have fields for data quantity, location and quality-of-service. This is not just to support possible future differential charging for quality of service, but also to help with emergency call tracking, and to comply with a 2002 European directive on telecomms requires all mobile operators retain location information on mobile phones for at least a year, for law enforcement purposes. There was a proposal for an online cost-control mechanism to limit the charges incurred for each user [901], but this appears to have stalled. The cost-control mechanisms are not being standardized but can involve forwarding charging data from either the local network or the gateway to the home environment which will be able to have the call terminated if the available credit is exhausted (as with a prepaid SIM card) or if the use appears to be fraudulent. It's tempting to draw a parallel with NATO's failure to make IFF work properly across different countries' armed forces, which I discussed in Chapter 19. The public good (in this case, transparent and dependable cost

[4] This protocol is an example of multiple simultaneous discovery, having been invented by our group at Cambridge, by Pedersen, and by Rivest and Shamir, independently in 1995 [40, 1013,

control) isn't high on the agendas of the stakeholders who'd have to work together to make it happen. Indeed, it's even worse. It would be hard to find a general anywhere in North America or Europe who didn't agree that decent IFF would be a Good Thing; but many of the major stakeholders depend for their existence on confusion pricing and would see a decent charging system as a serious threat.

The extreme difficulty of engineering a global solution has left the market open to a variety of local ones. In the USA, there is a move to integrate RFID-based credit cards with NFC-compatible mobile phones. In the UK, a scheme called PayForIt was launched in 2007 by the main mobile operators that aims to replace premium SMS services with a WAP-based protocol that 'provides a uniform payment experience' and requires 'clear pricing, access to terms and conditions and merchant contact details before a purchase is confirmed'. O2, one of the big networks, required all its users to switch from June 2007. They said there would be 'reduced customer care issues'.

Personally, I am sceptical, because of the lack of bankable guarantees for the customer; the terms and conditions state that 'any queries or complaints regarding Goods and Services must be referred to the Supplier,' while firms advertising PayForIt transaction acquisition quote fixed prices and seem ready to accept all comers, respectable or otherwise. Perhaps the scheme will simply hold up the growth of phone-based payments by making them more fiddly and restricting them to more capable handsets, while not resolving the underlying trust problem. A law that enforced customer rights would be better for all. In its absence, the phone companies appear to be setting up a banking system but without the regulations and controls that long and bitter experience has shown necessary for bank stability and trustworthiness. In civilised countries, mafiosi are not allowed to run banks. But gangsters have a basic human right (in America at least) to own a phone company. So phone companies should not run banks.

20.5 Summary

Phone fraud is a fascinating case study. People have been cheating phone companies for decades, and recently the phone companies have been vigorously returning the compliment. To start off with, systems were not really protected at all, and it was easy to evade charges and redirect calls. The mechanisms adopted to prevent this — out-of-band signalling — proved inadequate as the rapidly growing complexity of the system opened up many more vulnerabilities. These range from social engineering attacks on users through poor design and management of terminal equipment such as PBXes to the exploitation of various hard-to-predict feature interactions.

On the mobile front, the attempts to secure GSM and its third generation successor make an interesting case study. Their engineers concentrated on communications security threats rather than computer security threats, and they concentrated on the phone companies' interests at the expense of the customers'. Their efforts were not entirely in vain but did not give a definitive solution.

Overall, the security problems in telecomms are the result of environmental changes, including deregulation, which brought in many new phone companies. But the main change was the introduction of premium rate numbers. While previously phone companies sold a service with a negligible marginal cost of provision, suddenly real money was involved; and while previously about the only serious benefit to be had from manipulating the system was free calls, or calls that were hard for the police to tap, suddenly serious money could be earned. The existing protection mechanisms were unable to cope with this evolution. However, the major phone companies are so threatened by price competition that their business models are now predicated on confusion pricing. So the incentives for an overhaul of the billing mechanisms are just not there.

Ultimately, I suspect, the regulator will have to step in. The best solution in the USA could be the extension of Regulation E, which governs electronic banking, to phone companies — as they have become de facto banks. When you can use your mobile phone to buy ferry tickets and songs, and feed parking meters, it's performing all the functions of an electronic purse, which if issued by a traditional payment service provider such as VISA or Mastercard would fall squarely under Reg E. Also, I believe that either the FCC or the Federal Reserve should have the right to ban known criminals from owning or managing regulated phone companies. Europe is even further behind, but there is some action in the regulatory pipeline: a draft Directive that will impose a Europe-wide duty on companies to disclose security breaches in telecomms to affected customers, similar to the breach notification laws in force in over 30 U.S states in 2007. This at least will be a start.

Research Problems

Relatively little research is done outside phone company and intelligence agency labs on issues related specifically to phone fraud and wiretapping. There is growing interest in traffic analysis, which I'll discuss later; and in the likely effects of next-generation value added services, which are bound to introduce new feature interactions and other vulnerabilities. The interaction between all communications (especially mobile), platform security, and the mechanisms used to protect distributed systems security, also looks like fertile ground for both interesting

expects greater resilience and availability from the phone system than from the Internet — for example, to get through to emergency services — and as the two systems converge there will be some interesting assurance problems.

Further Reading

There are a lot of scattered articles about phone fraud, but nothing I know of that brings everything together. The underlying technologies are described in a number of reference books, such as [1061] on GSM, and more can be found on websites such as [1190]. An overview of UMTS can be found in [642], and a number of relevant research papers at [92]. To keep up with phone fraud, a useful resource is the Discount Long Distance Digest [390]. NIST has a guide on how to evaluate your PBX for vulnerabilities [943]. Finally, there's a survey of threats to mobile payment systems by the Mobile Payment Forum that gives a summary of the state of play as of 2002 [897].

CHAPTER

21

Network Attack and Defense

Whoever thinks his problem can be solved using cryptography, doesn't understand his problem and doesn't understand cryptography.

— Attributed by Roger Needham and Butler Lampson to Each Other

If you spend more on coffee than on IT security, then you will be hacked. What's more, you deserve to be hacked.

— Richard Clarke, Former U.S. Cybersecurity Tsar

21.1 Introduction

So far we've seen a large number of attacks against individual computers and other devices. But attacks increasingly depend on connectivity. Consider the following examples.

1. An office worker clicks on an attachment in email. This infects her PC with malware that compromises other machines in her office by snooping passwords that travel across the LAN.
2. The reason she clicked on the attachment is that the email came from her mother. The malware had infected her mother's machine and then sent out a copy of a recent email, with itself attached, to everyone in mum's address book.
3. Her mother in turn got infected by an old friend who chose a common password for his ISP account. When there are many machines on a network, the bad guys don't have to be choosy; rather than trying to guess the password for a particular account, they just try one password over and over for millions of accounts. Given a webmail account, they can send out bad email [illegible]

4. Another attack technique that makes sense only in a network context is *Google hacking*. Here, the bad guys use search engines to find web servers that are running vulnerable applications.
5. The malware writers infect a whole lot of PCs more or less at random using a set of tricks like these. They then look for choice pickings, such as machines in companies from which large numbers of credit card numbers can be stolen, or web servers that can be used to host phishing web pages as well. These may be auctioned off to specialists to exploit. Finally they sell on the residual infected machines for under a dollar a time to a *botnet herder* — who operates a large network of compromised machines that he rents out to spammers, phishermen and extortionists.
6. One of the applications is *fast-flux*. This changes the IP address of a web site perhaps once every 20 minutes, so that it's much more difficult to take down. A different machine in the botnet acts as the host (or as a proxy to the real host) with each change of IP address, so blocking such an address has at most a temporary effect. Fast-flux hosting is used by the better phishing gangs for their bogus bank websites.

There are many attacks, and defenses, that emerge once we have large numbers of machines networked together. These depend on a number of factors, the most important of which are the protocols the network uses. A second set of factors relate to the *topology of the network*: is every machine able to contact every other machine, or does it only have direct access to a handful of others? In our example above, a virus spreads itself via a social network — from one friend to another, just like the flu virus.

I've touched on network aspects of attack and defense before, notably in the chapters on telecomms and electronic warfare. However in this chapter I'm going to try to draw together the network aspects of security in a coherent framework. First I'm going to discuss networking protocols, then malware, then defensive technologies, from filtering and intrusion detection to the widely-used crypto protocols TLS, SSH, IPsec and wireless LAN encryption. Finally I'll discuss network topology. The most immediate application of this bundle of technologies is the defence of networks of PCs against malware, however as other devices go online the lessons will apply there too. In addition, many network security techniques can be used for multiple purposes. If you invent a better firewall, then — like it or not — you've also invented a better machine for online censorship and a better police wiretap device as well. Conversely, if mobility and virtual private networks make life tough for the firewall designer, they can give the censor and the police wiretap department a hard time, too.

21.2 Vulnerabilities in Network Protocols

This book isn't an appropriate place to explain basic network protocols. The telegraphic summary is as follows. The *Internet Protocol* (IP) is a stateless protocol that transfers packet data from one machine to another; IP version 4 uses 32-bit *IP addresses*, often written as four decimal numbers in the range 0–255, such as 172.16.8.93. People have started to migrate to IP version 6, as the 4 billion possible IPv4 addresses will have been allocated sometime between 2010 and 2015; IPv6 uses 128-bit addresses. Most modern kit is ready to use IPv6 but the changeover, which companies will probably do one LAN at a time, will no doubt throw up some interesting problems. The *Domain Name System* (DNS) allows mnemonic names such as `www.ross-anderson.com` to be mapped to IP addresses of either kind; there's a hierarchy of DNS servers that do this, ranging from thirteen top-level servers down through machines at ISPs and on local networks, which cache DNS records for performance and reliability.

The core routing protocol of the Internet is the *Border Gateway Protocol* (BGP). The Internet consists of a large number of *Autonomous Systems* (ASs) such as ISPs, telcos and large companies, each of which controls a range of IP addresses. The routers — the specialized computers that ship packets on the Internet — use BGP to exchange information about what routes are available to get to particular blocks of IP addresses, and to maintain routing tables so they can select efficient routes.

Most Internet services use a protocol called *transmission control protocol* (TCP) that is layered on top of IP and provides virtual circuits. It does this by splitting up the data stream into IP packets and reassembling it at the far end, automatically retransmitting any packets whose receipt is not acknowledged. IP addresses are translated into the familiar Internet host addresses using the *domain name system* (DNS), a worldwide distributed service in which higher-level name servers point to local name servers for particular domains. Local networks mostly use ethernet, in which devices have unique ethernet addresses (also called MAC addresses) that are mapped to IP addresses using the *address resolution protocol* (ARP). Because of the growing shortage of IP addresses, most organisations and ISPs now use the *Dynamic Host Configuration Protocol* (DHCP) to allocate IP addresses to machines as needed and to ensure that each IP address is unique. So if you want to track down a machine that has done something wicked, you will often have to get the logs that map MAC addresses to IP addresses.

There are many other components in the protocol suite for managing communications and providing higher-level services. Most of them were developed in the good old days when the net had only trusted hosts and security wasn't a concern. So there is little authentication built in. This is

a particular problem with DNS and with BGP. For example, if a small ISP mistakenly advertises to a large neighbour that it has good routes to a large part of the Internet, it may be swamped by the traffic. Here at least there are sound economic incentives, in that ISPs either swap, or pay for, routes with their peers; BGP security is in effect bodged up using manual intervention. DNS is trickier, in that disruptions are often malicious rather than mistaken. A DNS server may be fed wrong information to drive clients to a wicked website. This can be done either wholesale, by an attack on the DNS servers of a large ISP, or at the local level. For example, many homes have a wireless router attached to a broadband connection, and the router contains the address of the DNS server the customer uses. In an attack called *Drive-By Pharming*, the villain lures you to view a web page containing javascript code that sets your router's DNS server to one under his control [1213]. The effect is that next time you try to go to `www.citibank.com`, you may be directed to a phishing site that emulates it. For this reason it's a really good idea to change the default password on your home router.

21.2.1 Attacks on Local Networks

Suppose the attacker controls one of your PCs. Perhaps one of your employees was careless; or maybe he's gone bad, and wants to take over an account in someone else's name to defraud you, or to do some other bad thing such as downloading child porn in the hope of framing someone. There are several possibilities open to him.

1. He can install packet sniffer software to harvest passwords, get the root password, and thus take over a suitable account. Password-sniffing attacks can be blocked if you use challenge-response password generators, or a protocol such as Kerberos or ssh to ensure that clear text passwords don't go over the LAN. I described Kerberos in Chapter 3, and I'll describe SSH later.
2. Another approach is to masquerade as a machine where the target user — say the sysadmin — has already logged on. It is often possible for the attacker simply to set his MAC address and IP address to those of the target. In theory, the target machine should send 'reset' packets when it sees traffic to its IP address that's not in response to its own packets; but many machines nowadays have personal firewalls, which throw away 'suspicious' packets. As a result, the alarm doesn't get raised [300].
3. There's a whole host of technical address-hijacking attacks that work fine against old-fashioned LANs. An example I gave in the first edition of my book was that the attacker gives wrong answers to ARP messages, claiming to be the target, and may stop the target machine noticing and

raising the alarm by sending it a false subnet mask. Another possibility is to send bogus DHCP messages. Attacks like this may or may not work against a modern switched ethernet, depending on how it's configured.

4. A further set of attacks target particular platforms. For example, if the target company uses Linux or Unix servers, they are likely to use Sun's *Network File System* (NFS) for file sharing. This allows workstations to use a network disk drive as if it were a local disk, and has a number of well-known vulnerabilities to attackers on the same LAN. When a volume is first mounted, the client gets a *root filehandle* from the server. This is in effect an access ticket that refers to the root directory of the mounted filesystem, but that doesn't depend on the time, or the server generation number, and can't be revoked. There is no mechanism for per-user authentication: the server must trust a client completely or not at all. Also, NFS servers often reply to requests from a different network interface to the one on which the request arrived. So it's possible to wait until an administrator is using a file server and then masquerade as him to overwrite the password file. Filehandles can also be intercepted by network sniffing, though again, switched ethernet makes this harder. Kerberos can be used to authenticate clients and servers, but many firms don't use it; getting it to work in a heterogeneous environment can be difficult.

So the ease with which a bad machine on your network can take over other machines depends on how tightly you have the network locked down, and the damage that a bad machine can do will depend on the size of the local network. There are limits to how far a sysadmin can go; your firm might need to run a complex mixture of legacy systems for which Kerberos just can't be got to work. Also, a security-conscious system administrator can impose real costs. At our lab we argued with our sysadmins for years, trying to get access to the Internet for visiting guests, while they resisted on both technical protection and policy grounds (our academic network shouldn't be made available to commercial users). In the end, we solved the problem by setting up a separate guest network that is connected to a commercial ISP rather than to the University's backbone.

This raises a wider problem: where's the network boundary? In the old days, many companies had a single internal network, connected to the Internet via a firewall of some kind. But compartmentation often makes sense, as I discussed in Chapter 9: separate networks for each department can limit the damage that a compromised machine can do. There may be particularly strong arguments for this if some of your departments may have high protection requirements, while others need great flexibility. In our university, for example, we don't want the students on the same LAN that the payroll folks use; in fact we separate student, staff and administrative [illegible]

are also separated by college and department. Recently, mobility and virtual networks have made definition of clear network boundaries even harder. This debate goes by the buzz-word of *deperimiterisation* and I'll return to it later.

One final attack is worth mentioning under the heading of attacks on local networks, and that's the *rogue access point*. Occasionally one finds WiFi access points in public areas, such as airports, that have been deployed maliciously. The operator might sit in the airport lounge with a laptop that accesses the Internet via a paid WiFi service and advertises a free one; if you use it, he'll be able to sniff any plaintext passwords you enter, for example to your webmail or Amazon account, and if you tried to do online banking he might conceivably send you to a malicious site. So the effects can be somewhat like drive-by pharming, although more reliable and less scalable. In addition, rogue access points may also be devices that employees have installed for their own convenience in defiance of corporate policy, or even official nodes that have been misconfigured so that they don't encrypt the traffic. Whether unencrypted WiFi traffic is a big deal will depend on the circumstances; I'll discuss this in more detail later when we come to encryption.

21.2.2 Attacks Using Internet Protocols and Mechanisms

Moving up now to the Internet protocol suite, the basic problem is similar: there is no real authenticity protection in the default mechanisms. This is particularly manifest at the lower level TCP/IP protocols, and has given rise to many attacks.

Consider for example the 3-way handshake used by Alice to initiate a TCP connection to Bob and set up sequence numbers.

This protocol can be exploited in a surprising number of different ways. Now that service denial is becoming really important, let's start off with the simplest service denial attack: the *SYN flood*.

21.2.2.1 SYN Flooding

The attack is quite simply to send a large number of SYN packets and never acknowledge any of the replies. This leads the recipient (Bob, in Figure 21-4) to accumulate more records of SYN packets than his software can handle. This attack had been known to be theoretically possible since the 1980s but came to public attention when it was used to bring down Panix, a New York ISP, for several days in 1996.

A technical fix has been incorporated in Linux and some other systems. This is the so-called 'SYNcookie'. Rather than keeping a copy of the incoming SYN packet, *B* simply sends out as *Y* an encrypted version of *X*. That way, it's

A → B: SYN; my number is X
B → A: ACK; now X+1
SYN; my number is Y
A → B: ACK; now Y+1
(start talking)

Figure 21.1: TCP/IP handshake

not necessary to retain state about sessions which are half-open. Despite this, SYN floods are still a big deal, accounting for the largest number of reported attacks (27%) in 2006, although they were only the third-largest in terms of traffic volume (18%, behind UDP floods and application-layer attacks) [86].

There is an important general principle here: when you're designing a protocol that anyone can invoke, don't make it easy for malicious users to make honest ones consume resources. Don't let anyone in the world force your software to allocate memory, or do a lot of computation. In the online world, that's just asking for trouble.

21.2.2.2 Smurfing

A common way of bringing down a host in the 90s was *smurfing*. This exploited the *Internet control message protocol* (ICMP), which enables users to send an echo packet to a remote host to check whether it's alive. The problem was with broadcast addresses that are shared by a number of hosts. Some implementations of the Internet protocols responded to pings to both the broadcast address as well as the local address — so you could test a LAN to see what was alive. A collection of such hosts at a broadcast address is called a *smurf amplifier*. Bad guys would construct a packet with the source address forged to be that of the victim, and send it to a number of smurf amplifiers. These would then send a flurry of packets to the target, which could swamp it. Smurfing was typically used by kids to take over an *Internet relay chat* (IRC) server, so they could assume control of the chatroom. For a while this was a big deal, and the protocol standards were changed in August 1999 so that ping packets sent to a broadcast address are no longer answered [1144]. Another part of the fix was socio-economic: vigilante sites produced lists of smurf amplifiers. Diligent administrators spotted their networks on there and fixed them; the lazy ones then found that the bad guys used more and more of their bandwidth, and thus got pressured into fixing the problem too. By now (2007), smurfing is more or less fixed; it's no longer an attack that many people use.

But there's a useful moral: *don't create amplifiers*. When you design a network protocol, be extremely careful to ensure that no-one who puts one packet in can get two packets out. It's also important to avoid feedback and loops. A classic example was *source routing*, A feature of early IP that enabled the sender

of a packet to specify not just its destination but the route that it should take. This made attacks too easy: you'd just send a packet from A to B to C to B to C and so on, before going to its final destination. Most ISPs now throw away all packets with source routing set. (There was an alarm in early 2007 when it turned out that source routing had found its way back into the specification for IPv6, but that's now been fixed [417].)

21.2.2.3 Distributed Denial of Service Attacks

As the clever ways of creating service-denial attacks have been closed off one by one, the bad guys have turned increasingly to brute force, for example by sending floods of UDP packets from infected machines. The *distributed denial of service* (DDoS) attack made its appearance in October 1999 with the attack already mentioned on a New York ISP, Panix. In DDoS, the attacker subverts a large number of machines over a period of time and, or on a given signal, these machines all start to bombard the target with traffic [391]. Curiously, most of the machines in the first botnets around 1999–2000 were U.S. medical sites. The FDA insisted that medical Unix machines which were certified for certain clinical uses have a known configuration. Once bugs were found in this, there was a guaranteed supply of vulnerable machines; an object lesson in the dangers of monoculture.

Nowadays, botnets are assembled using all sorts of vulnerabilities, and a market has arisen whereby people who specialise in hacking machines can sell their product to people who specialise in herding them and extracting value. Compromised machines typically pass down a kind of value chain; they are first used for targeted attacks, then for sending spam, then (once they get known to spam filters) for applications like fast flux, and then finally (once they're on all the blacklists) for DDoS.

DDoS attacks have been launched at a number of high-profile web sites, including Amazon and Yahoo, but nowadays the major sites have so much bandwidth that they're very hard to dent. The next development was extortionists taking out online horserace-betting sites just before popular race meetings that would have generated a lot of business, and demanding ransoms not to do it again. Some bookmakers moved their operations to high-bandwidth hosting services such as Akamai that are highly distributed and can cope with large packet volumes, and others to specialist ISPs with packet-washing equipment that filters out bad packets at high speed. However the real fix for extortion wasn't technical. First, the bookmakers got together, compared notes, and resolved that in future none of them would pay any ransom. Second, the Russian government was leant on to deal with the main gang; three men were arrested in 2004 and sent to prison for eight years in 2006 [791].

For a while, there was a technical arms race. Attackers started to spoof source IP addresses, and to reflecting packets off innocuous hosts [1011]. One

countermeasure was traceback: the idea was that whenever a router forwards a packet, it would also send an ICMP packet to the destination, with a probability of about 1 in 20,000, containing details of the previous hop, the next hop, and as much of the packet as will fit. Large-scale flooding attacks could thus be traced back to the responsible machines, even despite forged source IP addresses [154]. However, this arms race has largely fizzled out, and for a number of reasons. First, Microsoft changed their network stack to make it much harder for an infected machine to send a packet with a spoofed IP address; you now need to hack the operating system, not just any old application. Second, more and more equipment at the edges of the network won't accept spoofed packets, and now about half of broadband ISPs filter them out [86].

However, the main reasons for the arms race stopping had to do with economics. Tracing back packets didn't work across different autonomous systems; if you were a large telco, for example, you would not give network access to your competitors' technical staff. So the bad guys found that in practice nobody came after them, and stopped using spoofing. Also, once markets emerged round about 2004 for botnet machines to be bought and sold (along with credit card numbers, spam contracts and other criminal services), the price of compromised machines fell so low, and botnets started to become so large, that the whole game changed. Instead of using a handful of compromised machines to send out clever attacks via amplifiers using spoofed source addresses, the bad guys simply burn thousands of end-of-life botnet machines to send the bad packets directly. The rapier has been replaced with the Kalashnikov.

Most recently, in 2005–7, there have been attempts to target core services such as DNS and thus take down the whole Internet. DNS has now been expanded to thirteen servers (the maximum the protocol will support), and many of them use *anycast* — a protocol whereby when you ask to resolve the domain name of a host into an IP address, the result that you get depends on where you are. The net effect is that DNS has become a massively distributed system using a lot of very fast machines connected to very high-capacity networks. In the end, the brute force of the modern DDoS attack was simply answered by even more brute force. If a hundred peasants with Kalashnikovs are going to shoot at you, you'd better buy a tank with good enough armor to absorb the fire.

Large-scale DDoS attacks on critical services seem quiescent at present. There are a few residual worries, though.

- There's a rising tide of DDoS attacks that happen by accident rather than as a result of malice. For example, in 2003 the University of Wisconsin-Madison found itself receiving hundreds of thousands of packets per second requesting the time. It turned out that Netgear had sold some 700,000 routers that

it was, and to ask again a second later if no response was received. Netgear ended up paying the university to maintain a high-bandwidth time server for them. There have been dozens of similar incidents [303].

- There's a steady stream of DDoS attacks by spammers and phishermen on the websites of organisations that try to hinder their activities, such as Artists Against 419 and Spamhaus.
- There are continuing worries that DDoS attacks might come back on an even larger scale. As of September 2007, there are several botnets with over half a million machines [742]. The operators of such networks can send out packet floods that will disable all but the biggest sites. These worries have been amplified in some quarters by the 2007 attacks on Estonia (even although that attack would not have harmed a large commercial target like Microsoft or Google). The highest attack rate seen in 2006 was 24 Gbit/sec, compared with 10 Gbit/sec in 2004 and 1.2 Gbit/sec in 2002 [86]. A further order-of-magnitude increase could put all but the most distributed targets at risk.
- Even if we never see the Internet taken down by a monster botnet, attacks can still be carried out on smaller targets. Prior to the Estonia incident, there had been a DDoS attack on the servers of an opposition party in Kyrgyzstan, and these followed the site when it was relocated to North America [1081]. Certainly, DDoS puts a weapon in the hands of gangsters that can be rented out to various unsavoury people.
- That said, one mustn't forget online activism. If a hundred thousand people send email to the White House protesting against some policy or other, is this a DDoS attack? Protesters should not be treated as felons; but drawing legislative distinctions can be hard.

21.2.2.4 Spam

Spam is in some respects similar to a DDoS attack: floods of generally unwanted traffic sent out for the most part by botnets, and often with clear criminal intent. The technical aspects are related, in that both email and the web protocols (`smtp` and `http`) assume wrongly that the lower levels are secure. Just as DDoS bots may forge IP addresses, spam bots may forge the sender's email address.

Spam differs in a various ways, though, from packet-level DDoS. First, it's enough of an annoyance and a cost for there to be real pressure on ISPs to send less of it. If you're a medium-sized ISP you will typically peer with other ISPs as much as you can, and buy routes from large telcos only where you can't trade for them; if other ISPs see you as a source of spam they may drop your peering arrangements, which costs real money. As a result, some ISPs are starting to do *egress filtering*: they monitor spam coming out of their own networks and then quarantine the infected PCs into a 'walled garden' from

Second, unlike DDoS, spam does not seem to be tailing off. It does appear that spammers are consolidating, in that most spam comes from several dozen large gangs. This is apparent from the 'lumpiness' of spam statistics: if there were hundreds of thousands of mom-and-pop spam operations, you'd expect to see spam volumes pretty constant, but this is no longer what we see [305]. So rather than spending more money on spam filters, it might be cheaper to get the police to arrest the gangs. Trends do change over time, though. Between 2006 and 2007, we've seen a drop in *backscatter* — in messages sent to the people whose email addresses were forged by spammers. Quite a lot of this came from anti-virus products, and it was pointed that the vendors were often breaking the antispam laws by sending messages saying 'Product X found worm Y in the message you sent'. If worm Y was known to use address forgery, the only conceivable purpose of sending such a message to the party who hadn't sent the offending message was to advertise product X. At the same time, there's been a huge increase in mule recruitment spam.

21.2.2.5 DNS Security and Pharming

I've given two examples so far of attacks in which a user is duped by being directed to a malicious DNS server, with the result that when he tries to go to his bank website he ends up entering his password into a fake one instead. This is generally referred to as *pharming*. I mentioned drive-by pharming, in which people's home routers are reconfigured by malicious javascript in web pages they download, and rogue access points in which the attacker offers a WiFi service to the victim and then has complete control over all his unprotected traffic.

There are a number of other variants on this theme, including feeding false DNS records to genuine servers. Older DNS servers would accept additional records without checking; if they asked your server where X was, you could volunteer an IP address for Y as well. This has been fixed but there are still older servers in use that are vulnerable. Such attacks are often referred to as *DNS cache poisoning* as they basically affect users who trust the information about the target that's cached by the attacked machine. They've been used not just for pharming but also for simple vandalism, such as replacing the web site of a target company with something offensive. They can also be used for censorship; China has used DNS spoofing against dissident websites for years, and by 2007 was also using it to make Voice of America news unavailable [1297].

A number of researchers have worked on a proposed upgrade to the security of DNS, but they have turned out to be hard to deploy for economic reasons; most of the things that secure DNS would do can be done by TLS without the need for new infrastructure, and individual network operators don't get enough benefit from DNS security until enough other operators have adopted

21.3 Trojans, Viruses, Worms and Rootkits

Computer security experts have long been aware of the threat from malicious code. The first such programs were *Trojan Horses*, named after the horse the Greeks left supposedly as a gift for the Trojans but which contained soldiers who opened the gates of Troy to the Greek army. The use of the term for malicious code goes back many years (see the discussion in [774] p 7). There are also *viruses* and *worms*, which are self-propagating malicious programs, and to which I've referred in earlier chapters. There is debate about their precise definitions; the common usage is that a Trojan is a program that does something malicious (such as capturing passwords) when run by an unsuspecting user, while a worm is something that replicates and a virus is a worm which replicates by attaching itself to other programs.

Finally, the most rapidly-growing problem is the *rootkit* — a piece of software that once installed on a machine surreptitiously places it under remote control. Rootkits can be used for targeted attacks (law enforcement agencies use them to turn suspects' laptops into listening devices) or for financial fraud (they may come with keyloggers that capture passwords). One of the most salient features of rootkits nowadays is stealth; they try to hide from the operating system so that they can't be located and removed using standard tools. But sooner or later rootkits are identified and tools to remove them are written. On the other side of the coin, most PCs infected in this way end up in botnets, so another way of framing the problem is how botnets are set up, maintained and used. The rootkit vendors now do after-sales-service, supplying their customers the botnet herders with the tools to upgrade rootkits for which removal tools are becoming available.

21.3.1 Early History of Malicious Code

Malicious code, or *malware*, seems likely to appear whenever a large enough number of users share a computing platform. It goes back at least to the early 1960's, when machines were slow and their CPU cycles were carefully rationed between different groups of users — with students often at the tail of the queue. Students invented tricks such as writing computer games with a Trojan inside to check if the program is running as root, and if so to create an extra privileged account with a known password. By the 1970s, large time-sharing systems at universities were the target of more and more pranks involving Trojans. All sorts of tricks were developed. In 1978, John Shoch and Jon Hupp of Xerox PARC wrote a program they called a *worm*, which replicated itself across a network looking for idle processors so it could assign them tasks. They discussed this in a paper in 1982 [1164].

In 1984, Ken Thompson wrote a classic paper 'On Trusting Trust', in which he showed that even if the source code for a system were carefully

inspected and known to be free of vulnerabilities, a trapdoor could still be inserted [1247]. Thompson's trick was to build the trapdoor into the compiler. If this recognized that it was compiling the login program, it would insert a trapdoor such as a master password that would work on any account. (This developed an idea first floated by Paul Karger and Robert Schell during the Multics evaluation in 1974 [693].) Of course, someone might try to stop this by examining the source code for the compiler, and then compiling it again from scratch. So the next step is to see to it that, if the compiler recognizes that it's compiling itself, it inserts the vulnerability even if it's not present in the source. So even if you can buy a system with verifiably secure software for the operating system, applications and tools, the compiler binary can still contain a Trojan. The moral is that vulnerabilities can be inserted at any point in the tool chain, so you can't trust a system you didn't build completely yourself.

1984 was also the year when computer viruses appeared in public following the thesis work of Fred Cohen. He performed a series of experiments with different operating systems in which he showed how code could propagate itself from one machine to another, and (as I mentioned in Chapter 8) from one compartment of a multilevel system to another. This caused alarm and consternation, and within about three years we started to see the first real live viruses in the wild[1]. Almost all of them were PC viruses as DOS was the predominant operating system. They spread from one user to another when users shared programs on diskettes or via bulletin boards.

One early innovation was the 'Christma' virus, which spread round IBM mainframes in December 1987. It was a program written in the mainframe command language REXX that had a header saying 'Don't read me, EXEC me' and code that, if executed, drew a Christmas tree on the screen — then sent itself to everyone in the user's contacts file. It was written as a prank, rather than out of malice; and by using the network (IBM's BITNET) to spread, it was ahead of its time.

The next year came the Internet worm, which alerted the press and the general public to the problem.

21.3.2 The Internet Worm

The first famous case of a service denial-attack was the Internet worm of November 1988 [421]. This was a program written by Robert Morris Jr that exploited a number of vulnerabilities to spread from one machine to another. Some of these were general (e.g. 432 common passwords were used in a guessing attack, and opportunistic use was made of `.rhosts` files), and others

[1] That's when I first came across them, as a security guy working in a bank; we now learn that the first ever computer virus in the wild [illegible]

were system specific (problems with `sendmail`, and the `fingerd` bug mentioned in section 4.4.1). The worm took steps to camouflage itself; it was called `sh` and it encrypted its data strings (albeit with a Caesar cipher).

Its author claimed that this code was not a deliberate attack on the Internet — merely an experiment to see whether code could replicate from one machine to another. It was successful. It also had a bug. It should have recognised already infected machines, and not infected them again, but this feature didn't work. The result was a huge volume of communications traffic that completely clogged up the Internet.

Given that the Internet (or more accurately, its predecessor the Arpanet) had been designed to provide a very high degree of resilience against attacks — up to and including a strategic nuclear strike — it was remarkable that a program written by a student could disable it completely.

What's less often remarked on is that the mess was cleaned up and normal service restored within a day or two; that it only affected Berkeley Unix and its derivatives (which may say something about the dangers of the Microsoft monoculture today); and that sites that kept their nerve and didn't pull their network connection recovered more quickly as they could find out what was happening and get the fixes.

21.3.3 How Viruses and Worms Work

A virus or worm typically has two components — a replication mechanism and a payload. A worm simply makes a copy of itself somewhere else when it's run, perhaps by breaking into another system (as the Internet worm did) or mailing itself as an attachment to the addresses on the infected system's address list (as many recent worms have done). In the days of DOS viruses, the commonest way for a virus to replicate was to append itself to an executable file and patch itself in, so that the execution path jumps to the virus code and then back to the original program.

Given a specific platform, there are usually additional tricks available to the virus writer. For example, if the target system was a DOS PC with a file called ACCOUNTS.EXE, one could introduce a file called ACCOUNTS.COM, which DOS will execute in preference. DOS viruses could also attack the boot sector or the partition table, and there are even printable viruses — viruses all of whose opcodes are printable ASCII characters, so that they can even propagate on paper. A number of DOS viruses are examined in detail in [817].

The second component of a virus is the payload. This will usually be activated by a trigger, such as a date, and may then do one or more of a number of bad things:

- make selective or random changes to the machine's protection state (this is what we worried about with multilevel secure systems);