How to Cheat at the Lottery
(or, Massively Parallel Requirements Engineering)

Ross Anderson

University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK
Ross.Anderson@cl.cam.ac.uk

Abstract. Collaborative software projects such as Linux and Apache have shown that a large, complex system can be built and maintained by many developers working in a highly parallel, relatively unstructured way.

In this note, I report an experiment to see whether a high quality system specification can also be produced by a large number of people working in parallel with a minimum of communication.

1 Introduction

Experienced software engineers know that perhaps 30% of the cost of a software product goes into specifying it, 10% into coding, and the remaining 60% on maintenance. This has profound effects on computer science. For example, when designing new programming languages the motive nowadays is mostly not to make coding easier, but to cut the costs of maintenance. There has also been massive interest in open source software products such as Linux and Apache, whose maintenance is undertaken by thousands of programmers working worldwide in a voluntary and cooperative way.

Open source software is not entirely a recent invention; in the early days of computing most system software vendors published their source code. This openness started to recede in the early 1980s when pressure of litigation led IBM to adopt an 'object-code-only' policy for its mainframe software, despite bitter criticism from its user community. The pendulum now seems to be swinging back, with Linux and Apache gaining huge market share.

In his influential paper 'The Cathedral and the Bazaar' [1], Eric Raymond compares the hierarchical organisation of large software projects in industry ('the cathedral') with the more open, unstructured approach of cooperative developers ('the bazaar'). He makes a number of telling observations about the efficiency of the latter, such as that "Given enough eyeballs, all bugs are shallow". His more recent paper, 'The Magic Cauldron' [2], explores the economic incentives that for-profit publishers have found to publish their source code, and concludes that IBM's critics were right: where reliability is paramount, open source is best, as users will cooperate in finding and removing bugs.

There is a corollary to this argument, which I explore in this paper: the next priority after cutting the costs of maintenance should be cutting the costs of specification.

Specification is not only the second most expensive item in the system development life cycle, but is also where the most expensive things go wrong. The seminal study by Curtis, Krasner and Iscoe of large software project disasters found that failure to understand the requirements was mostly to blame [3]: a thin spread of application domain knowledge typically led to fluctuating and conflicting requirements which in turn caused a breakdown in communication. They suggested that the solution was to find an 'exceptional designer' with a deep understanding of the problem who would assume overall responsibility.

But there are many cases where an established expert is not available, such as when designing a new application from scratch or when building a competitor to a closed, proprietary system whose behaviour can only be observed at a distance.

There are also some particular domains in which specification is well known to be hard. Security is one example; the literature has many examples of systems which protected the wrong thing, or protected the right thing but using the wrong mechanisms. Most real life security failures result from the opportunistic exploitation of elementary design flaws rather than 'high-tech' attacks such as cryptanalysis [4]. The list of possible attacks on a typical system is long, and people doing initial security designs are very likely to overlook some of them. Even in a closed environment, the use of multiple independent experts is recommended [5].

Security conspicuously satisfies the five tests which Raymond suggested would identify the products most likely to benefit from an open source approach [2]. It is based on common engineering knowledge rather than proprietary techniques; it is sensitive to failure; it needs peer review for verification; it is business critical; and its economics include strong network effects. Its own traditional wisdom, going back at least to Auguste Kerkhoffs in 1883, is that cryptographic systems should be designed in such a way that they are not compromised if the opponent learns the technique being used. In other words, the security should reside in the choice of key rather than in obscure design features [6].

It therefore seemed worthwhile to see if a high quality security specification could be designed in a highly parallel way, by getting a lot of different people to contribute drafts in the hope that most of the possible attacks would be considered in at least one of them.

2 Experimental design

The opportunity to test this idea was provided by the fact that I teach courses in cryptography and computer security to second and third year undergraduates at Cambridge. By the third year, students should be able to analyse a protection problem systematically by listing the threats, devising a security policy and then recommending mechanisms that will enforce it. (The syllabus and lecture notes are available online at [7].)

By a security policy, we mean a high level specification which sets out the threats to which a system is assumed to be exposed and the assurance properties which are to be provided in response. Like most specifications, it is a means of communication between the users (who understand the environment) and the system engineers (who will have to implement the encryption, access control, logging or other mechanisms). So it must be clearly comprehensible to both communities; it should also be concise.

The students see, as textbook examples of security policy:

- the Bell-LaPadula model, which is commonly used by governments to protect classified information and which states that information can only flow up the classification hierarchy, and never down. Thus a civil servant cleared to 'Secret' can read files at 'Secret' or below, but not 'Top Secret', while a process running at 'Secret' can write at the same level or above, but never down to 'Unclassified';
- The Clark-Wilson model, which provides a reasonably formal description of the double-entry bookkeeping systems used by large organisations to detect fraud by insiders;
- The Chinese Wall model, which models conflicts of interest in professional practice. Thus an advertising account executive who has worked on one bank's strategy will be prevented from seeing the files on any other banking client for a fixed period of time afterwards;
- The British Medical Association model, which describes how flows of personal health information must be restricted so as to respect the established ethical norms for patient privacy. Only people involved directly in a patient's care should be allowed to access their medical records, unless the patient gives consent or the records are de-identified effectively.

The first three of these are documented in [8] and the fourth in [9]. Further examples of security policy models are always welcome, as they help teach the lesson that 'security' means radically different things in different applications. However, developing a security policy is usually hard work, involving extensive consultation with domain experts and successive refinement until a model emerges that is compact, concise and agreed by all parties.

Exceptions include designing a policy for a new application, and for a competitor to a closed system. In such cases, the best we can do may be to think long and hard, and hope that we will not miss anything important.

I therefore set the following exam question to my third year students:

> **You have been hired by a company which is bidding to take over the National Lottery when Camelot's franchise expires, and your responsibility is the security policy. State the security policy you would recommend and outline the mechanisms you would implement to enforce it.**

3 The UK National Lottery

For the benefit of overseas readers, I will now give a simplified description of our national lottery. (British readers can skip the next two paragraphs.)

The UK's national lottery is operated by a consortium of companies called Camelot which holds a seven year licence from the government. This licence is up for renewal, which makes the question topical; and presumably Camelot will refuse to share its experience with potential competitors. A large number of franchised retail outlets sell tickets. The customer marks six out of 49 numbers on a form which he hands with his money to the operator; she passes it through a machine that scans it and prints a ticket containing the choice of numbers plus some further coded information to authenticate it.

Twice a week there is a draw on TV at which a machine selects seven numbered balls from 49 in a drum. The customers who have predicted the first six share a jackpot of several million pounds; the odds should be (49 choose 6) or 13,983,816 to one against, meaning that with much of the population playing there are several winners in a typical draw. (Occasionally there are no winners and the jackpot is 'rolled over' to the next draw, giving a pot of many millions of pounds which whips the popular press to a frenzy.) There are also smaller cash prizes for people who guessed only some of the numbers. Half the takings go on prize money; the other half gets shared between Camelot, the taxman and various charitable good causes[1].

The model answer I had prepared had a primary threat model that attackers, possibly in cahoots with insiders, would try to place bets once the result of the draw is known, whether by altering bet records or forging tickets. The secondary threats were that bets would be placed that had not been paid for, and that attackers might operate bogus vending stations which would pay small claims but disappear if a client won a big prize.

The security policy that follows logically from this is that bets should be registered online with a server which is secured prior to the draw, both against tampering and against the extraction of sufficient information to forge a winning ticket; that there should be credit limits for genuine vendors; and that there should be ways of identifying bogus vendors. Once the security policy has been developed in enough detail, designing enforcement mechanisms should not be too hard for someone skilled in the art – though there are some subtleties, as we shall see below.

The exam was set on the first of June 1999 [10], and when the scripts were delivered that evening, I was eager to find out what the students might have come up with.

[1] Appointing the members of the committees that dish out the money is a source of vast patronage for the Prime Minister and, according to cynics, is the real reason for the Lottery to exist.

4 Results

Thirty four candidates answered the question, and five of their papers were good enough to be kept as model answers. All of these candidates had original ideas which are incorporated in this paper, as did a further seven candidates whose answers were less complete. As the exam marking is anonymous, the 'co-authors' of this specification are a subset of the candidates listed in the ackowledgements below. The question was a 'good' one in that it divided the students up about equally into first, second and third class ranges of marks. Almost all the original ideas came from the first class candidates.

The contributions came at a number of levels, including policy goal statements, discussions of particular attacks, and arguments about the merits of particular protection mechanisms.

4.1 Policy goal statements

On sorting out the high level policy statements from the more detailed contributions, the first thing to catch the eye was a conflict reminiscent of the old debate over who should pay when a 'phantom withdrawal' happens via an automatic teller machine – the customer or the bank [4].

One of the candidates assumed that the customer's rights must have precedence: *'All winning tickets must be redeemable! So failures must not allow unregistered tickets to be printed.'* Another candidate assumed the contrary, and thus the *'worst outcome should be that the jackpot gets paid to the wrong person, never twice.'* Ultimately, whether systems fail in the shop's favour or the customer's is a regulatory issue. However, there are consequences for security. In the context of cash machine disputes, it was noted that if the customer carries the risk of fraud while only the bank is in a position to improve the security measures, then the bank may get more and more careless until an epidemic of fraud takes place. We presumably want to avoid this kind of 'moral hazard' in a national lottery; perhaps the solution is for disputed sums to be added back to the prize fund, or distributed to the 'good causes'.

As well as protecting the system from fraud, the operator must also convince the gaming public of this. This was expressed in various ways: *'take care how you justify your operations;' 'don't forget the indirect costs of security failure such as TV contract penalties, ticket refund, and publicity of failure leading to bogus claims;' 'at all costs ensure that there is enough backup to prevent unverifiable ticket problems.'* The operator can get some protection by signs such as *'no winnings due unless entry logged'* but this cover is never total.

Next, a number of candidates argued that it was foolish to place sole reliance on any single protection mechanism, or any single instance of a particular type of mechanisms. A typical statement was: *'Don't bet the farm on tamper-resistance'.* For example, if the main threat is someone forging a winning ticket after tapping the network which the central server uses to send ticket authenticator codes to vending machines, we might not just encrypt the line but also delay paying jackpots for several days to give all winners a chance to claim. (Simply encrypting the authentication codes would not be enough, if a technician who dismantled the encryption device at the server could get both the authentication keys and the encryption keys.) Translated into methodology, this suggests a security matrix approach which maps the threats to the protection mechanisms, and makes it easy for us to check that at least two independent mechanisms constrain every serious threat.

Various attempts were made to reuse existing security policies, and particularly Clark-Wilson. These were mostly by weak candidates and not very convincing. But three candidates did get some mileage; for example, one can model the lottery terminal as a device that turns an unconstrained data item (the customer selection) into a constrained data item (the valid lottery ticket) by registering it and printing an authentication code on it. Such concepts can be useful in designing separation-of-duty mechanisms for ticket redemption and general financial control, but do not seem to be enough to cover all the novel and interesting security problems which a lottery provides.

Some candidates wondered whether a new franchisee would want to extend the existing lottery's business model, such as by allowing people to buy tickets over the phone or the net. In that case, one should try to design the policy to be extensible to non-material sales channels. (Internet based lottery ticket sales have since been declared to be a good thing by the government [11].)

Finally, some attention needs to be paid to protecting genuine winners. The obvious issue is safeguarding the privacy of winners who refuse publicity; less obvious issues include the risk that winners might be traced, robbed and perhaps even murdered during the claim process. For example, the UK has some recent history of telephone technicians abusing their access to win airline tickets and other prizes offered during phone-in competitions; one might be concerned about the risk that a technician, in cahoots with organised crime, would divert the winners' hotline, intercept a jackpot claim, and dispatch a hit squad to collect the ticket. In practice, measures to control this risk are likely to involve the phone company as much as the lottery itself.

4.2 Discussions of particular attacks

This leads to a discussion of attacks. There were several views on how the threat model should be organised; one succinct statement was *'Any attack that can be done by an outsider can be done at least as well by an insider. So concentrate on insider attacks'*. This is something that almost everyone knows, but which many system designers disregard in practice. Other candidates pointed out that no system can defend itself against being owned by a corrupt organisation, and that senior insiders should be watched with particular care[2].

[2] One of the companies that originally made up the Camelot consortium had to leave after its chief executive was found by the High Court to have tried to bribe a competing consortium during the bidding for the original lottery franchise.

Moving now to the more technical analysis, a number of interesting attack scenarios were explored.

1. A number of candidates remarked that in the absence of enforceable limits on ticket sales per machine, an operator could issue large numbers of tickets without any intention of paying for them. In extremis, he might issue all 13,983,816 tickets required to win a jackpot. The obvious fix is to have a value counter to enforce a system of credit limits – but where? If the terminal cannot be completely tamperproof, we need an online solution. But this is not enough: three candidates warned about possible traffic insertion attacks at the server end, so having synchronised value counters at both the terminal and the server might be a good idea[3]. So would banking industry style batch controls and totals.
2. Three candidates discussed tricking genuine terminals into attaching to a fake server. The goal might be fraud (after the draw, forge tickets with authenticators calculated using the fake server key) or denial of service (undermine the lottery's credibility by causing vendors to print tickets which cannot be redeemed if they win). The obvious fix is to have the terminals authenticate the server.
3. There was concern about the prospect of a winning ticket being claimed simultaneously at several shops. The general consensus was that an online operation with guaranteed commit-abort semantics and strong authentication of the terminal should be required to pay a winning ticket.
4. Candidates disagreed about the threat from refunds. If refunds are allowed, then someone might get a refund on a forged ticket and later present the original if it wins. (Historically, refund mechanisms have been a source of fraud with systems such as prepaid electricity meters [5]). The simplest solution is not to allow any refunds at all; and alternative is to allow them only in very restricted circumstances (only for data entry errors, only while the customer is still in the shop, only up to close of play, only while the terminal is online, and subject to collection and audit of all refunded tickets along with all locally paid winning tickets).
5. Although tamper resistance cannot be relied on completely, it can still be helpful. But should we protect the whole vending machine or just an embedded crypto module? If the latter, there is a risk that vendors will tamper with the rest of the system so that it reports only a proportion of their takings, in effect competing with the lottery by issuing the other tickets on their own account. So it is probably a good idea to make the whole vending machine tamper resistant, except for those components such as the receipt printer where user access is unavoidable.
6. It may be a good idea to allow small claims to be cashed anywhere in the system. This way, any bogus tickets should be spotted as quickly as possible. This will also help the operator detect any rogue merchants running completely bogus vending operations with unauthorised equipment.

[3] but see section 4.3 below on the problems of redundancy

7. This will not help, however, with another possible attack on the vending machine's tamper resistance. This is where a wiretap is used to reveal which machine sold a winning ticket (whether directly, or from published information about where a prizewinner lives); the attacker then burgles the shop, steals the machine and digs the authentication keys or logs out of it. So vending machines should not contain enough information to forge a ticket, except in the instant that a genuine ticket is being printed.
   There are some secondary design concerns here. How will the machines validate the lower-value tickets that are paid out locally – only online? Or will some of the authenticator code be kept in the vending station? But in that case, how do we cope with the accidental or malicious destruction of the machine that sold a jackpot winning ticket, and how do we pay small winnings when the machine that sold the ticket is offline?
8. Close attention has to be paid to failure modes. If random errors and system failures can lead to individual gain then, as with some burglar alarm systems [12], deliberate attempts to cause failure can be expected. They may lead not just to occasional frauds but also to more widespread service denial.
9. Some attention has to be paid to whether the system should collect evidence with a view to resolving possible disputes with franchisees, and if so what form it should take. The naive approach is to ask for everything to carry a digital signature, but this is largely irrelevant to the kind of attack one expects from the experience of electricity token vending [5] – namely that a vendor sells a large number of tickets and then reports the machine stolen. The solution is likely to involve contractual obligations, insurance, and monitoring of vending machines by the central server.
10. There should be enough privacy protection to prevent punters learning the pattern of bets; even if the draw is random, other people's choice of numbers will not be, and this will skew the odds. The published history of jackpots gives some information on this (it is already extensively analysed) but one should not give out any more information, unless the operator decides to as a matter of policy. If it were believed that insiders had an advantage by knowing the popularity of each number, this could seriously erode confidence. (There is no realistic way to stop a clever vending agent rigging up some means of collecting local statistics, but at least the authentic national statistics should be protected.)
11. Some candidates suggested using the BBC's broadcast radio clock signal as an authentication input to the vending terminals; but one candidate correctly pointed out that this signal could be jammed without much difficulty. This was a highly effective suggestion, in the sense that when it was mentioned to a colleague who had recently done an audit of a different online gaming system, his response was 'Oh s***!'

4.3 Reasoning about particular protection mechanisms

The third type of contribution from the candidates can be roughly classed as reasoning about particular mechanisms.

1. Five candidates discussed the kind of authenticator needed to validate the ticket. One suggested a digital signature; one reasoned that a MAC[4] would do; three pointed out that even a random number generated by the central server would be enough (though a MAC might be more convenient).
2. There was some discussion of how one should eliminate single points of vulnerability such as the encryption devices that would generate authenticators if this were done algorithmically. There was also some reasoning about separation of duty, such as how to prevent any single individual from being able to validate a jackpot win. One might, for example, have 'orange' and 'blue' encryption boxes (if encryption were used to generate authenticators) or databases (if the authenticators were randomly generated) and have the call centre send out an orange manager and a blue manager to visit the winner and check the claim.
3. There was also discussion of the nature of the Trusted Computing Base[5]. Is this all of the central server or just part of it? How much protection can you get by separating function across replicated hardware, such as multiple databases or crypto boxes at the centre, or by having part of the authenticator computed centrally and part by the vending machine? In the latter case, do you need to have all the vending machines online when claims are paid, or do you upload winning authenticators – in which case what did you gain by decentralising part of the codes before the draw? The efficacy of replication is well known to be bounded by common mode errors (particularly specification errors [13]). And in any case, how do you prevent yourself being laid open to service denial attacks? There are similar tradeoffs involving security and resilience when we consider whether to put the value counters at the server, in the terminals, or in both. Managing these tradeoffs may involve several iterations of a detailed design, with criticism from a number of bright people in parallel.
4. Some candidates discussed the level of reliance that could be placed on physical ticket security technologies, such as holograms; the general consensus was that the stock is bound to be stolen. Thus the primary protection should be digital not physical. However, having a printed serial number on the ticket costs little and may do some good if it is also an input to the MAC or other authentication process. This way, a crook has to do some physical forgery as well. Serial numbers might also provide a second level control against wiretap attacks, as one might transmit only the first few digits of the serial number to the server and arrange matters so that the remaining digits were a MAC computed with a key known only to the ticket printing company.

[4] For the benefit of readers without a security background, a MAC – or message authentication code – is a cryptographic checksum computed on data using a secret key and which can only be verified by principals who also possess that secret key. By comparison, a digital signature can in principle be verified by anybody. See [8] for more detail

[5] the set of hardware, software and procedural components whose failure could lead to a compromise of the security policy

5. The candidates came up with quite a number of checklist items of the kind that designers often overlook – e.g. 'tickets must be associated with a particular draw'. This might seem obvious, but a protocol design which used a purchase date, ticket serial number and server-supplied random challenge as input to a MAC computation might appear plausible to a superficial inspection. The evaluator might not check to see whether a shopkeeper could manufacture tickets that could be used in more than one draw. Experienced designers appreciate the value of such checklists.
6. The user interface design also needs some care. We mentioned above that one should ask for telephone claims of big wins after the draw, then delay payment for a week or two in the hope that any duplicated winning ticket will become evident. This delay can be used for (and excused by) due diligence activities such as getting a sworn statement from each jackpot winner that the ticket is theirs, and that they are not cheating on a partner or a syndicate with an equity stake in the win – an activity which has given rise to most of the publicised disputes over the years.
7. As the company will want to convince outsiders that it is not cheating, it might veer towards involving third parties in many of the protection mechanisms. For example, in order to secure the database of bets before the draw, it would be natural to use a third party timestamping service rather than simply having a spare copy of a CD of the database; if a spare database were preferred, then one might leave it with a bank rather than at an in-house backup site.
8. How much audit effort is needed? Certainly, one should collect both winning and refunded tickets for examination. Key staff should be watched; a Jaguar in the car park should sound an alarm more quickly than it did in the Aldrich Ames case. There are many other details, such as:
   - what will be the controls on adding vending machines to network (and for that matter adding servers);
   - how long should logs be kept;
   - how to deal with refunded tickets;
   - how to deal with tickets that are registered but not printed (these will exist if you insist that unregistered tickets are never printed);
   - what system will be used to transfer takings from merchants to the operator (we don't want a fake server to be able to collect real money);
   - what audit requirements the taxman will impose;
   - what sort of 'intrusion detection' or statistical monitoring system will be incorporated to catch the bugs and/or attacks that we forgot about or which crept in during the implementation. E.g., we might have a weird bug which enables a shopkeeper to manufacture occasional medium-sized winners which he credits against his account. If this is significant, it should turn up in long term statistical analysis.

   As we work through these details, it becomes clear that for most of the system, 'Trusted' means not just tamper resistant but subject to approved audit and batch control mechanisms.

4.4 How complete are the above lists?

At the time I set the exam question, I had never played the lottery. I did not perform this experiment until after marking the exam scripts; this helped ensure an even playing field for the candidates. In fact, by the time I got round to buying a ticket, I had already written the first draft of this article and circulated it to colleagues. My description of the ticket purchase process in that draft had been based on casual observation of people ahead of me in Post Office queues, and was wrong in an unimportant but noticeable detail: I had assumed that the authentication code was printed on the form filled by the customer whereas in fact it appears on the receipt (which I have therefore called 'the ticket' in this version of the paper). None of my colleagues noticed, and none of them has since admitted to having ever played. Indeed, only one of the candidates shows any sign of having done so. I had expected a negative correlation between education and lottery participation (many churches already denounce the lottery as a regressive tax on the poor, the weak and the less educated) but the strength of this correlation surprised me.

So the above security analysis was done essentially blind – that is, without looking at the existing system. Subsequent observation of the procedures actually implemented by Camelot suggests only two further issues.

1. Firstly, the Camelot rules allow small franchisees to pay wins of up to £500, while the agencies in main Post Offices can pay up to £10,000. This seems a better idea than our 4.2.6; it makes it a lot harder to run a bogus vending operation. Wins in the £500–£10,000 range are much commoner than jackpots, and main Post Offices are much harder to 'forge' than corner shops.
2. Secondly, the tickets are numbered as suggested in 4.3.4, but printed on continuous stock. The selected bet numbers and authentication codes are printed on the front, while pre-printed serial numbers appear on the back. This may have both advantages and disadvantages. If a standard retail receipt printer is used, it can produce a paper audit roll with a copy of all tickets printed. This may well be more convincing to a judge than any cryptographic protection for electronic logs. On the other hand, the audit roll might facilitate ticket forgery as in 4.2.7, and there may be synchronisation problems (the sample ticket I purchased has two successive serial numbers on the back). When synchronising tickets with serial numbers, one will have to consider everything from ticket refunds to how operators will initialise a new roll of paper in the ticket printer, and what sort of mistakes they will make.

The final drafting of the threat model, security policy and detailed functional design is now left as an exercise to the reader.

5 Discussion and Conclusions

Linux and Apache prove that software maintenance can be done in parallel; the experiment reported in this paper shows that requirements engineering can too.

There has been collaborative specification development before, as with the 'set-discuss' mailing list used to gather feedback during the development of the SET protocol for electronic payments. However, such mechanisms tend to have been rather ad-hoc, and limited to debugging a specification that was substantially completed in advance by a single team. The contribution of this paper is twofold: to show that it is possible to parallelise right from the start of the exercise, and to illustrate how much value one can add in a remarkably short period of time. Our approach is a kind of structured brainstorming, and where a complete specification is required for a new kind of system to a very tight deadline, it looks unbeatable: it produced high quality input at every level from policy through threat analysis to technical design detail.

The bottleneck is the labour required to edit the contributions into shape. In the case of this paper, the time I spent marking scripts, then rereading them, thinking about them and drafting the paper was about five working days. A system specification would usually need less polishing than a paper aimed at publication, but the time saved would have been spent on other activities such as doing a formal matrix analysis of threats and protection mechanisms, and finalising the functional design.

Finally, there is an interesting parallel with testing. It is known that different testers find the same bugs at different rates – even if Alice and Bob are equally productive on average, a bug that Alice finds after half an hour will only be spotted by Bob after several days, and vice versa. This is because different people have different areas of focus in the testing space. The consequence is that it is often cheaper to do testing in parallel rather than series, as the average time spent finding each bug goes down [14]. The exercise reported in this paper strongly supports the notion that the same economics apply to requirements engineering too. Rather than paying a single consultant to think about a problem for twenty days, it will often be more efficient to pay fifteen consultants to think about it for a day each and then have an editor spend a week hammering their ideas into a single coherent document.

Acknowledgements

I am grateful to the security group at Cambridge, and in particular to Frank Stajano, for a number of discussions. I also thank JR Rao of IBM for the history of the 'object code only' effort, and Karen Spärck Jones who highlighted those parts of the first draft that assumed too much knowledge of computer security for a general engineering audience, and also persuaded me to buy a ticket.

Finally, the students who contributed many of the ideas described here were an anonymous subset of our third year undergraduates for 1998–9, who were:

PP Adams, MSD Ashdown, JJ Askew, T Balopoulos, KE Bebbington, AR Beresford, TJ Blake, NJ Boultbee, DL Bowman, SE Boxall, G Briggs, AJ Brunning, JR Bulpin, B Chalmers, IW Chaudhry, MH Choi, I Clark, MR Cobley, DP Crowhurst, AES Curran, SP Davey, AJB Evans, MJ Fairhurst, JK Fawcett, KA Fraser, PS Gardiner, ADOF Gregorio, RG Hague, JD Hall, P Hari Ram, DA Harris, WF Harris, T Honohan, MT Huckvale, T Huynh, NJ Jacob, APC Jones, SR King, AM Krakauer, RC Lamb, RJP Lancaster, CK Lee, PR Lee, TY Leung, JC Lim, MS Lloyd, TH Lynn, BR Mansell, DH Mansell, AD McDonald, NG McDonnell, CJ McNulty, RD Merrifield, JT Nevins, TM Oinn, C Pat Fong, AJ Pearce, SW Plummer, C Reed, DJ Scott, AA Serjantov, RW Sharp, DJ Sheridan, MA Slyman, AB Swaine, RJ Taylor, ME Thorpe, BT Waine, MR Watkins, MJ Wharton, E Young, HJ Young, WR Younger, W Zhu.

References

1. The Cathedral and the Bazaar. Eric S. Raymond, `http://www.tuxedo.org/~esr/writings/cathedral-bazaar/`
2. The Magic Cauldron. Eric S. Raymond, `http://www.tuxedo.org/~esr/writings/magic-cauldron/`
3. A Field Study of the Software Design Process for Large Systems. Bill Curtis, Herb Krasner, Neil Iscoe, Comm ACM 31.11 (Nov 88) pp 1268-87
4. Why Cryptosystems Fail. Ross Anderson, Comm ACM 37.11 (Nov 1994) pp 32-40, `http://www.cl.cam.ac.uk/users/rja14/wcf.html`
5. On the Reliability of Electronic Payment Systems. Ross Anderson and S Johann Beduidenhoudt, IEEE Trans. Software Engineering 22.5 (May 1996) pp 294-301, `http://www.cl.cam.ac.uk/ftp/users/rja14/meters.ps.gz`
6. La Cryptographie Militaire. Auguste Kerkhoffs, Journal des Sciences Militaires, 9th series, IX (Jan 1883) pp 5–38 and (Feb 1883) pp 161-191; `http://www.cl.cam.ac.uk/~fapp2/kerckhoffs/`
7. Security. Ross Anderson, University of Cambridge Computer Laboratory, `http://www.cl.cam.ac.uk/Teaching/1998/Security/`
8. Computer Security. Dieter Gollmann, John Wiley and Sons, 1999; ISBN 0-471-978442-2
9. A Security Policy Model for Clinical Information Systems. Ross J Anderson, Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp 30–43, Oakland, CA, 1996; conference paper is `http://www.cl.cam.ac.uk/ftp/users/rja14/oakpolicy.ps.Z`; full BMA version is `http://www.cl.cam.ac.uk/users/rja14/policy11/policy11.html`
10. The 1999 papers are at `http://www.cl.cam.ac.uk/tripos/y1999.html`; see `http://www.cl.cam.ac.uk/tripos/y1999PAPER7.pdf` for paper 7 in which this question is number 6
11. New Age lottery will be played on the net. Rupert Steiner, Sunday Times 25 July 1999 p 3.3
12. Denial of Service: An Example. Roger M Needham, Comm ACM 37.11 (Nov 1994) pp 42–46
13. Safeware. Nancy Leveson, Addison-Wesley (1994).
14. Murphy's law, the fitness of evolving species, and the limits of software reliability. Ross J Anderson, Robert M Brady and Robin C Ball; `http://www.cl.cam.ac.uk/ftp/users/rja14/bab.ps.gz`.