1. Formally state the two rules of the Bell-Lapadula (BLP) security policy model and then re-state them infomrally in terms of a single rule about the direction of information flow.

   Formal definitions:

   - No Read-Up

     A process running at security clearance α is not allowed to read from any files requiring a security clearance greater than α.

   - No Write-Down

     A process running at security clearance β is not allowed to write to any files requiring a security clearance less than β.

   Informally:

   Under the Bell-LaPadula policy, information is only allowed to flow "up" into higher security clearances.

2. Consider a distributed system in which A is a `TOP SECRET` process running on a machine Alice and B is a `CONFIDENTIAL` object residing on machine Bob.

   (a) Explain and justify whether A is allowed to read and/or write from B according to the BLP policy.

   According to the Bell-LaPadula Policy; A is allowed to read from B but not to write to B.

   The process A has `TOP SECRET` clearance. This is higher than `CONFIDENTIAL` and so A is allowed to read from any confidential objects. So A is allowed to read from B.

   Under the Bell-LaPadula policy, processes are not allowed to write to any objects requiring a lower security clearance than the running process. This prevents leaks by malicious users or processes. So A would not be allowed to write to B. Although this sounds counterintuitive, it prevents documents being leaked. Consider if A wanted to leak a `TOP SECRET` document C to a process D running on `CLASSIFIED`. A could do this by writing the contents of C into the file B. D could then request to read B and see the contents of C. Bell-LaPadula prevents this by disallowing write-down.

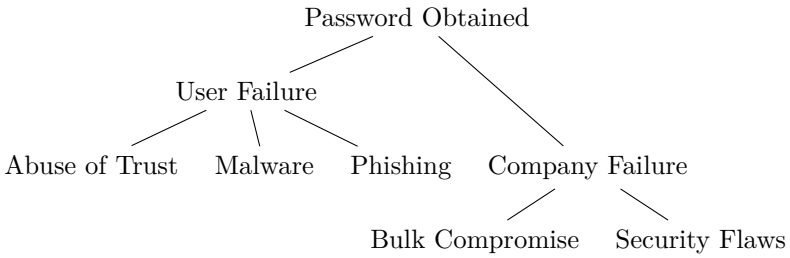
   (b) Discuss the claim made by some researchers that this scenario highlights a fundamental problem with the BLP policy.

   The Bell-LaPadula policy is designed with the idea that malicious insiders are the opponent. This matches the information-sensitive cold-war environment in which it was made. However, some researchers may call it overly-restrictive since it totally disallows all decreases in classification level. Consider a document about targets to bomb during the Second World War. On it's creation, this would be `TOP SECRET`. Under the Bell-LaPadula policy, this would remain `TOP SECRET` forever – long after the second world-war ended. This is not realistic and does not match the real-world.

   Consider also a public inquiry: during the inquiry, many documents will be classified. However, after the completion of the inquiry they would be made unclassified. Bell-LaPadula would again disallow this on the grounds that decreasing clearance is a potential security risk.

   While I accept that disallowing such reasonable actions is a flaw with Bell-LaPadula; I would argue that Bell-LaPadula's purpose is to prevent insider leaks of dangerous information; and it does that well. These situations are not fundamental flaws – rather edge cases for which Bell-LaPadula is not designed to deal with.

3. You're trying to steal my password. How would you go about it? Draw a fault tree and explain the options and their liklihood of success.



User Failure

These attacks involve exploiting user behaviour to gain their password.

Abuse of Trust

Trusted people are more able to get marks to do things they would not otherwise do. I give an example here, although there are many other methods of abusing trust.

As a supervisee, I'm in a position of relative trust (you expect that I'll send supervision work and have basic trust that I'm not going to try and scam you during supervisions).

An attack abusing trust would be to claim "KuDoS didn't work" and submit my supervision work using Google Docs or Dropbox or another online repository. The submissions for the first supervision would be legitimate and the second or third would be link to a spoof website impersonating the repository which required a login. I expect you'd be unlikely to blindly input your password on the first email from me. However, by the second or third supervision the chances would be increased to a meaningful level. With sufficient effort to make a good spoof website, the probability of success in this attack is moderately high.

Phishing

Phishing is one of the most common forms of scam. However, a generic "I'm the crown prince of Atlantis and I need your help" message is very unlikely to succeed. To have any hope of a successful scam, we would need targetted phishing (known as spearphishing). There are several methods of this.

The first is the simplest: find out who you're likely to receive an email from and send an email appearing to be from that person asking you to do something which requires logging into an account. We could buy a fake email domain (such as @camtab.co.uk for @cantab.ac.uk) and send an email from a legitimate looking address. The probability of success is low. You're is likely to notice the fake domain and perhaps would not fill in the form even if you thought the email legitimate. Note that if we sent an email pretending to be from a colleague such as a research partner we would be likely to use very different mannerisms and expose ourselves quickly. The probability of a direct spearphish success is moderate – nontrivial but we're likely to make some mistake.

Although the above scams may work, once you notice one failed attempt you'll be more wary for other scams. This would make it far harder to obtain your password. To maximise the chances of success, we would be better to find out who you are likely to receive an email from and then perform a targetted attack (as described above) on many of *those* people. Although the probability of individual success is small, if we make 50 such attacks we are likely to get some yield (we could use multiple layers of indirection although too large an attempt will mean we get noticed quickly). Once

we gain access to those people's emails, we can see their email history and send a legitimate seeming email (discussing topics they're likely to discuss and using their mannerisms) containing a link to our spoof website requiring a login. This comes from an email address they could reasonably expect to receive emails from and is the first time they're aware they're being scammed. I would suspect this would have a high chance of success (on a non-security professional).

Under this method, by infecting a person the mark doesn't even know, we could gain access to the internal system and send emails from legitimate addresses close to the mark. Emails from legitimate addresses are far more likely to be trusted than those from illegitimate addresses. As a side effect, we would also gain many of their colleagues passwords.

Malware

Infecting your computer with malware would be an effective way of obtaining your password. Malware can be spread in almost every way (in fact word documents are one of the most common ways of spreading malware and even PDF files can contain malware – although both need certain settings configured). An effective method would be to send a word document or pdf containing malware in an email discussing a seemingly legitimate topic.

Malware such as this is publically accessible. The liklihood of success is low since you're unlikely to read or open any files sent by a stranger – even if they're legitimate.

Company Failure

These methods are out of the users control and are as a result of failures of companies.

Bulk Compromise

Sometimes, companies face are hacked and the entire contents of their password databases are placed online. Under this attack, you would find databases of accounts and passwords from major data breaches and attempt to find the accounts which could belong to you. Since password reuse is high, a password you used on one account is likely to be used elsewhere and may be your password for more important things. We would then attempt logins using these passwords into the accounts we care about.

Security Flaws

Security flaws are rare – however when they are found, hacks are easy.

In 2012, journalist Mat Honan was hacked due to inconsistent security protocols between Google, Apple and Amazon leading to an emergent security flaw where publically available information could be used to reset passwords. Although this technically doesn't involve finding a password, emergent security flaws exist and can be used to hack people. However, the liklihood of finding one is low and the chance of success of it working against an individual is very low.

Social engineering company tech support for a password reset would have a very high probability of success. Unfortunately most security questions use publically available information. For example we could use Ancestry.com or a government database to find your mother's maiden name. With appropriate research, we could find the answer to many possible security questions and use those to password reset.

Larger websites and companies usually have good security protocols. I will not bother discussing possible attacks against them – they would have negligible chances of suc-

ceeding. However, some smaller websites have terrible security protocols; since they don't store sensitive information they think that investing in good security is pointless.

For example, I once attempted to set a C program that printed "Hello World" as my password to a site forcing me to create an account. The site then errored with a message saying a "field contains code" and gave a full description of the query which had failed – including all fields of the database and associated metadata. One of the fields was "txtPassword".

Security flaws in smaller companies (SQL injection, no cooldown or covert channels) lead to security flaws in larger companies. Many people reuse passwords and so a breach in a smaller company can expose passwords used for more important things. Consider a company who asks people to make an account using an email address and a password. There is a nontrivial probability these passwords are shared. If the mark had ever created an account with a security-oblivious small company, we could extract their password information. This relies on knowledge of which sites have bad security protocols and great patience.

The liklihood of success is very low against an individual target. This sort of attack is better-suited to "hacking any person".

4. (a) Why are Nonces useful?

   Nonces can be used to ensure that hashes are "fresh" (†). This protects against replay attacks.

   If Alice sends a message containing an unpredictable Nonce N_A and receives a response from Bob containing the Nonce N_A, then that message must have been generated in response to her initial message. So this message cannot be a replay.

   (†) Provided that the generator for the nonces is not predictable.

   (b) Why are Salts useful?

   Salts ensure that under different circumstances, the same value will not hash to the same thing.

   This means that hashes are independent – so a compromise of one hash will not compromise other hashes.

   Consider a medium-sized company which does not salt hashes having a data breach. If this company has 100000 hashes leaked then we can see which of the users share passwords. We could then find the most popular passwords and break it (either by brute-force attacks [this is reasonable since we have a lot of users to test on and a frequently used password is likely to be easy to guess], spearphishing any one of the individuals who use this password or any other method). After breaking this one users password, we have many hundreds of users login details.

5. Consider the "car unlocking protocols" in the slides. What attacks can be made on each of them?

   - Static

     With the static car unlocking protocol, the transponder sends it's key: so the signal the transponder transmits never changes. If this key is intercepted then it can be relayed at a later date by any malicious party. This enables any party who is nearby to anyone who uses the key to permanently have acccess to that key. This is a serious security flaw and is very easy to exploit.

     Many transponders using static protocols are simple and only use 16 bit signals. There is insufficient entropy. This opens the system up to a brute force attack. Consider entering a large car park with 400 cars and trying random signals until success. This would take only an expected $\frac{2^{16}}{400} \approx 160$ codes. This brute force attack works for any car unlocking protocol which has insufficient entropy.

- Non-interactive

  Under this model, the transponder generates a nonce; and sends the transponder id salted with the nonce encrypted under the transponders key. The engine controller will then check the nonce against a list of previously used nonces to check that it is fresh (not a replay).

  However, in practice since storage is limited, the engine controller will not store all nonces which have ever been used. Instead it will maintain a list of length n containing "recently used" nonces. So, if the engine controller never hears a signal or if a signal was used sufficiently long ago that it's no longer in the engine controller's memory; then the signal will open the car.

  So we could attack this protocol by listening to a signal from the transponder and storing it until the engine controller has forgotten it. Or intercept a signal from the transponder before the car hears it.

- Interactive

  Interactive controllers are much more difficult to attack.

  However, if we know how the nonces are generated then we can launch an attack. Many nonces are not truly random numbers but counters or timestamps (imprecise timestamps are vulnerable). If the counter is sufficiently slow (or we can otherwise predict what the nonce would be). Then we can work out the challenge that the engine controller will send at a particular time in the future. We can then issue that challenge to the transponder and obtain the response required to turn on the car at a point in the future. We now wait until this point and then issue a request to the engine controller. The engine controller will issue the challenge and we can respond with the correct answer. This will turn on the engine.

6. (a) Discuss the lessons learned from the London Ambulance Servvice disaster from the following viewpoints.

   (i) Requirements engineering

   The London Ambulance Service Disaster resulted from attempting to change the service with minimal consultation with the users of the proposed CAD system. A crucial stage of requirements engineering should be a consultation with users (in the case of the London Ambulance Service, this would be ambulance drivers and call-takers), trade unions and management. This would ensure that the requirements actually match and will solve the problem – rather than just management's idea of what the problem is.

   The London Ambulance Service (and many other companies of that era) tried to implement Business Process Reorganisation. This method of top-down change through a new technical system disregards people in the organisation, how they work best and their skills. Attempting top-down change of a business leads to tensions and distrust between management and employees. Businesses work because the employees work effectively. Attempting to change how employees work and ignoring established working practices and staff skills can seriously damage the effectiveness of a business.

   Advice from consultants should be listened to very carefully – in the case of the London Ambulance Service Disaster, the consultants said that a system could be adapted from an off-the-shelf solution for £1.9m in 19 months and that an Automatic Vehicle Location System (AVLS) would cost more. The service management then decided to attempt to get a bespoke solution for £1.5m *with* AVLS. This was out of line with what the consultants had said was feasible.

   (ii) Human factors

LAS managment designed the project with hard, unmanageable and inflexible deadlines. This put undue pressure on contractors – who upon realising that the project was going to fail started covering themselves. This created a climate of tension, mistrust and obstructiveness – where nobody wanted to take as little responsibility as possible. This meant people were rushing to mean unmanageable deadlines, making more mistakes and caring less and less as the project became less and less likely to succeed.

Due to the poor communication with employees, the specification was poor and employees had little faith in the project. This lead to a poor user experience and difficulty using the system. This was exemplified by a lack of staff training before the rollout of the project – the report called the training staff received "incomplete and inconsistent". This helped lead to the total lack of confidence in the new system.

(iii) Testing

There are four main flaws with the testing of the London Ambulance Service from which lessons can be learnt:

Lack of proper testing

The whole system never underwent proper testing after individual components had been integrated together. This meant the system suffered from a large amount of "emergent bugs" due to unexpected or unintended interactions. For example the two-line summary of the awaiting attention list and the system which placed jobs on the awaiting attention list both worked. However, these components were not tested properly in conjunction and so nobody realised that in practice, jobs would rapidly be scrolled past and lost forever.

In particular, the backup server had not been tested properly. Systems Options Limited had been under a large amount of time pressure and so their focus had been on making the ctual system work. Very little effort was put into the backup server. In safety-critical systems such as the London Ambulance Service, the backup server is as important as the main server. When the main system did fail, the backup server was so poorly tested that it didn't kick in and the whole system crashed the first time it tried to switch to the backup server.

The Computer Aided Dispatch system failed to deal with was imperfect data. It worked adequately well with perfect data, however when the data became imperfect the quality of service dropped dramatically and totally unsuitable ambulances were sent to incidents. The poor quality of geolocation data was entirely predictable and this gross oversight meant the system performed significantly better in tests than in the real world. Real world systems must always have provisions for poor data quality.

Insufficient Quality Assurance

Due to the tight timescale, the project was written in such a way that would make it fast to develop. To help with this goal, the programmers decided to write the user interface in Visual Basic – using a new and untested development tool designed for prototyping and the development of small programs. The use of untested systems likely caused bugs. Visual Basic was unmanageably slow that Central Ambulance Control staff had to preload all screens they would use and use windows multi-tasking or face unnacceptable loading times. This resulted in a massive space requirement for the system, cluttered

up every users workstation and greatly reduced the overall performance of the system. The review states while in testing Visual Basic on Windows 3.0 (the operating system on which the software was deployed), they found lots of unexplained crashes.

Systems Options did not appreciate that writing in the most appropriate programming language is almost always preferable to the programming language that will allow for the fastest development and that experimental software should never be used in large projects.

Due to insufficient quality assurance, there were bugs which were unlikely to be picked up by testing – most notably a careless memory leakage bug which over the course of three weeks filled up large parts of the file server causing the whole system to crash. Bugs such as that are unlikely to be picked up unless you test for that particular bug or debug the code. Proper quality assurance would have meant that this bug did not happen.

Problems found during the trial were ignored

The partial trial in the North-East Division exposed many of the flaws in the system. However, these problems were never forwarded onto management and were never properly addressed. These bugs remained in the final system. The system was never ready for a proper trial – it was still in development and only parts of it were used in the trial. This meant that large parts of the system were never trialled. The point of a trial is to expose any subtle bugs in a completed system before deployment. The trial of the Computer Aided Dispatch system was totally inneffectual since the system was not complete and the results were totally ignored.

The project didn't undergo a rolling deployment

The London Ambulance Service opted for a full deployment (without any backup) across all centres on a single day. This meant that the whole service was dependent on the CAD system working flawlessly. Given the safety-critical nature and size of the system, such a choice was very naïve. The system was well-suited to a rolling deployment across the regions of London – each centre worked almost independently of the others! A rolling deployment gradually increases the load on the new system as it proves itself effectual without removing the capability of swapping back to the old system until the new system has been tested in-the-field. Choosing a rolling deployment instead of an all-or-nothing deployment can avert almost every disaster – all but the most subtle bugs are caught almost immediately and the new service can be taken offline and replaced by the old one almost instantly.

(iv) Project Management

Management in the LASCAD project was extremely poor. I believe the core reason was a lack of management training. This meant that the board was inexperienced and naïve leading to many of the other mistakes: although many of the mistakes made on this project were so amateur that management training cannot be exclusively to blame.

The most obvious failing of management was ignoring the advice of consultants. This happened on three occasions.

The first during the project specification when management ignored the estimated cost and specifications of the project. This led management to believe they could get a solution for significatly cheaper than was realistic and with a significantly lower timescale.

When the project was proposed and IBM and many other companies advised that the timescale was unrealistic and the budget unreasonable management should have reconsidered. However, again management believed a solution could be obtained with their specifications at their price. They ignored the obvious sign that no company offered a solution for a sum close to that proposed by Systems Options Limited. Management's response to Systems Option's cheap offer was in-line with the advice from the Thames Regional Health Authority – which encouraged getting the best price at all costs rather than the best product.

Lastly, after the trial immediately before the launch the LAS Systems Manager carried out an independent review. This review highlighted key shortcomings of the current system claiming that the system was failing "almost daily", stressed that volume testing was essential and recommended reviewing the training of staff. This advice was neglectfully ignored: the senior manager who it was reported to never even presented it to the board.

LAS Management made a number of critical errors throughout the project: they hired a small company with no prior experience of a project such as CAD; they set impossible and un-challengeable deadlines – furthermore the review highlights that staff believed they risked "losing their jobs" if they challenged any deadlines. This environment of fear meant concerns could not be raised and were not listened to. The LAS board was also under political pressure for the project to be a success – and delivered on-time. This "need" for the project to work made the LAS board willing to accept Systems Option's misleading claims about their experience and assurances of a full and successful implementation. In summary: the LAS board appeared to be focussed on meeting deadlines rather than making the system functional and deceived themselves until they believed both would happen.

Lastly, management attempted to change the organisation from the top-down without communication with employees. This strategy is damaging to organisations. They underestimated the difficulties of changing the culture at the LAS and created a hostile environment. An example of this is alleged intentional misuse during trials of the new system – management distrusted crew enough to allege sabotage rather than a bad system; or crews disliked the new system enough to intentionally misuse it at risk of lives; or both. In all cases this highlights the toxic climate in the LAS.

(b) You have been hired by the Department of Energy and Climate Change (DECC) to manage a project to replace Britain's 47 million gas and electricity meters with "smart" meters that report energy use every 30 minutes. These reports go to a central service, and energy companies have access to their customers' readings. The goals of the system are to facilitate more flexible pricing, so customer demand can be brought more into line with supplly; to make it simpler for customers to switch energy companies; and to help DECC predict and manage energy demand.

Describe what measures you would take to reduce the liklihood of a project disaster.

Using the mistakes learnt from the LASCAD disaster and others like it; I'd take the following measures to avoid a project disaster:

- Prior to the start of such a large project, we should launch an inquiry and review into the expected benefits and risks of the program. This phase is important to evaluate the benefits of any project and will usually kill off any projects which have no benefits. Assuming the projects are not in government manifestos and companies aren't lobbying overly hard.

  For smart meters specifically, this review would conclude that customers would not change their behaviour significantly and that the energy savings

would be so negligible that almost any other investment (insulation, wind, solar, nuclear or others) would be preferable. At this point the project should be cancelled.

From this point onwards, I will assume that the project is a core point of a government manifesto and must go ahead regardless of the lack of perceivable benefits.

- Hire consultants before the project starts to review the expected cost and timescale. Then listen to and plan according to their advice. My knowledge of smart-meters and projects such as this is limited: consultants knowledge would not be. Many projects fail when management ignores the advice of professionals and solutions are purchased with unadvisable timescales or overly cheap costs. We should expect to pay no less than what the consultants recommend for no more than they say the solution will do.
- When the project to roll-out smart meters started in 2009, the smart meter technology was limited. I would propose postponing the start of the project until the technology is good enough.
- Installing smart-meters across the whole of the UK will take many years. If we do not install the modern smart-meters then we will be installing obselete devices by the end of the project. We should install modern smart-meters throughout the project – and at the start of the project we should make it clear to contractors that we plan to do so.
- A nationwide rollout should start only after a trial of installing smart-meters in a localised area (several towns and cities). We would use this to compare the change in energy usage and find out issues with the planned methodology and "iron out the bugs". This trial would expose most problems in the project early on before damage was done. Trials are essential in any large project such as this.
- The project should not have hard, enforced deadlines. While (soft) deadlines are important to ensure the project is proceeding as planned they should be flexible and approximate guides of performance rather than targets projects must meet even at the cost of quality.
- Projects such as LASCAD often fail when innapropriate or inexperienced contractors are hired. For a project as large as installing smart-meters throughout the country, I would recommend hiring only very experienced and very large contractors.
- A breakdown in communications between management and contractors (and between contractors) can lead to management becoming out of touch with the true progress of the project. I would recommend very regular updates from contractors on the progress of the project and to actively communicate throughout.
- To ensure the progress of the project is where we are reassured it is and is roughly on-track; we should have regular external reviews of progress. For a project as large and long as installing smart-meters across the UK, I would suggest *at least* annual reviews – and perhaps more during the early stages before the contractors have proven their capability.

7. "Security is made difficult by the inadequacies of users."

   To what extent do you agree with this. Write a short, bullet-point essay plan with factors for and against.

Introduction

I believe the statement is untrue, the majority of difficulty in security is caused by the competence of malicious attackers and inherant difficulty of the problem. User inadequacies can make security *more* difficult but are not the core reason.

Against

I believe that the majority of user-caused security breaches are due to difficult-to-use UI's and innapropriate defaults rather than incompetence. Consider a user with an easy-to-guess password which they re-use for many websites. Why were they allowed to use this password? The company should have recommended a better method of picking a password and prevented such a weak password. The way in which users misuse software is predictable and repeated: companies can and should take simple measures to prevent users making most common mistakes. While implementing these checks on passwords can be annoying, it is usually not inherantly difficult and is certainly not the core of the difficulty of security.

For

Conversely, consider a user who is phished into inputting their details into a spoof website. Although the company could attempt to invest heavily in anomaly detection, it would take dozens of irregular logins before the alarm was raised. By this time the scammer could make a new website. The fault here lies with the user – and mistakes such as this are the reason for numerous campaigns by banks and other companies never to click on links from emails and to always think before inputting passwords.

Case Study 1

Most security breaches are a combination of user and manufacturer failure. WannaCry was randomware distributed in 2017 which used a leaked software bug exposed by the NSA to encrypt files on infected computers and demand payment. One can argue WannaCry was created by manufacturer failure – there was a crucial vulnerability. However, since Microsoft released an update which patched the EternalBlue vulnerability quickly it's also possible to argue that users are to blame for not updating immediately.

There are many cases (such as the NHS) where updating is infeasable. The NHS runs specialist equipment which is not compatible with newer versions of Windows. This means that they cannot update (without massive expense and unnacceptable downtime in the health service). As a result the NHS was forced to be vulnerable by a bug in Microsoft's code.

Case Study 2

Consider the example of banks. Since the majority of bank security problems are due to users falling for obvious scams one can argue that user inadequacies are the cause of bank's security problems. I disagree: banking systems are inherantly complicated. Bank security systems are complicated that writing was invented for early banking and most civil research into cryptography is financed by banks. A bank which has an insecure system will rapidly lose public trust and rapidly go bust. Banks have a massive incentive to make their systems as secure as possible. It is only due to the large investment by banks into secure systems which has meant that in the 21st century; users are the weakest link. Banks have an incentive to make customers and the public believe that user inadequacies are the problem. There is much more publicity about

releasing card information to obvious scams than there is about bank vulnerabilities – such as the NoPIN attack which exposed a critical vulnerability in the worldwide chip-and-pin system.

Case Study 3

Next, consider an example where all users are highly trained and seldom have inadequacies: aviation. Every aeroplane user undergoes extensive training and as a result there are almost no accidents due to pilot misuse. In aviation, the cause of almost every crash is mechanical or extreme circumstances (such Malaysia Airlines Flight MH17 which was shot down in 2014, the 911 terrorist attacks or the bird strikes causing the "Miracle on the Hudson"). Consider the cause of the most recent aviation disasters: the Boeing 747 MAX MCAS bug. This software bug caused flights to turn downwards when an angle-of-attack sensor was damaged: the force required to trim the aircraft was unmanageable by pilots and this was not pilot error. Almost no aviation disasters are due to pilot inadequacies – although some such as flying in innapropriate conditions, flying over a warzone or not refueling correctly are debatably user fault.

Conclusion

Overall, the fault of most security issues is a mixture of both user inadequacies and flaws by manufacturers. I believe that security is often inherantly difficult and user inadequacies can make security *more* difficult but rarely make otherwise simple problems difficult.