DSV SU

SEC:I/IV2024
Information and Computer Security
Autumn Term 2010

Alan Davidson

Suggested Solutions to the Exam 2010-11-27 and
Comments on the Marking

The answers suggested here may contain more aspects or discussion than would normally be expected from students to achieve the highest grade. The intention is to give students not only explanations of why their answers were graded as they were, but also to give students plenty of food for thought (if applicable, for future exam sittings).

Please note that sources are quoted and referenced in these suggested answers. This is to provide accurate documentation only. In no way are students ever expected to remember and provide accurate quotes in exam papers for this course.

Problem 1

> Give motivated arguments that updates are both good and bad for system security.

It may be effective to avoid many software vulnerabilities during the design phase, given that many such vulnerabilities are relatively easily predicted. Nevertheless, trying to predict *all* possible environmental situations and threats at the design phase is increasingly difficult as the software or its environment grows in complexity. In such situations the most effective security measure for dealing with newly discovered vulnerabilities can be to develop and distribute updates that provide protection from those vulnerabilities.

Speed will be of the essence in developing, distributing and installing updates. It will not normally be possible for those who are responsible for the security of a system to analyse the content and effects of an update before installing it. This means that systems administrators have little choice but to trust the update and install it. The integrity of every system is therefore at risk with every update. There have been cases where important system updates have caused greater problems than the problems they were intended to solve, before they were retracted.

It is safe to assume that fixes to problems will involve more code that the version with a problem. In that case, the complexity of the software is increased with an update. According to Saltzer and Schroeder's Principle of Economy of Mechanism complexity is best avoided in the design of security mechanisms. In this respect updates that cause added complexity can be said to have an adverse effect on security.

When security updates are published they are often documented in terms of the vulnerabilities that they correct. This documentation could give pointers towards understanding how that vulnerability could be utilised[1].

One answer made the argument that updates fill disks and use CPU resources, which can be seen as a threat against availability. This is an example of how a novel answer can gain marks for a reasonably structured argument (even though this does not seem to be a very important factor in itself).

Problem 2

> Imagine that a person intends to implement their own symmetric encryption algorithm in order to keep large amounts of data safe on a hard drive. To ensure that the decryption and encryption processes are quick and effective she has decided to keep to a fairly simple substitutional

1 Though many times documentation of that vulnerability will in fact have been published before the update has been released.

method. The security and practicality of this method will to a significant degree be dependent on certain qualities of the keys that are used. Suggest and motivate what such qualities can be.

The two principle factors of importance for such relatively simple keys is length and randomness.

The longer the key the larger is the keyspace, and a perfectly random key means that no one key within that keyspace is more likely than another. The larger the keyspace the more difficult it is to execute a brute force attack with reasonable resources and within reasonable time.

Some answers suggested that longer keys will mean less repetition of patterns in the cryptotext. We have seen during the course that such repeating patterns are a problem for such methods as the Vigenere cipher. However, we have also seen such methods as Cipher Block Chaining that can be employed to reduce the occurrence of repeated patters. We see by this that the frequency of repeating patterns is not only dependent on the keysize, but can be dependent on the substitutional algorithm employed.

Simply stating the longer the key the better is ignoring practical issues. If the key was as long as the data on the hard drive then the encryption would be very strong, but totally impractical since it would take another hard drive to keep the key, and keeping the key safe would be just as difficult as protecting the original data. The key should be so long as to prohibit a brute force attack while at the same time being of a practically manageable size.

A number of answers clouded things by discussing issues like avoiding the reuse of keys, or being careful to transmit them carefully. These are not issues that are especially relevant to the problem of encrypting a hard-drive. Such answers suggest that the writer has difficulties differentiating encryption of messages from other encryption situations.

Some answers brought in discussions of properties of passwords. Unless the answer also explained the assumption that the key should be chosen by a user (a very unnecessary and unlikely assumption) such answers were assumed to be confused as to the difference between passwords and cryptographic keys.

Problem 3

Why is it important to change passwords after a period of time? Give well motivated arguments based on explanations of the kind of threats that passwords that are not regularly changed can suffer from.

Each time a password is used it is to some degree exposed and is therefore at risk from eavesdropping. For example, a password is typed in at a keyboard allowing others the opportunity to see what keys are pressed. The password may unavoidably be transmitted though insecure media on its way between the authenticated and authenticating nodes, allowing it to be sniffed on the way. If a password is written down on some medium as a necessary aid to remember large numbers of them, then whenever that medium is referred to the location of the password may be revealed. Sometimes one might accidentally type a password to one system as input to another, possibly revealing that password to a less secure system. Sometimes, one might even accidentally type a password in the wrong text field, causing it to be revealed on a screen. All these situations go to show that the more a password is used, the greater is the risk that it can be revealed to others. Note how this is a problem that is largely unrelated to the actual strength of the password, i.e. how well formed it is[2]. Regularly changing passwords reduces the risk that a password is with time discovered, or that a discovered password can be utilised for an extended period.

Even if a password is not discovered directly through eavesdropping, there may be possibilities to discover a password through other means. A system that allows multiple login attempts could be fooled by directly attempting large numbers of likely passwords until one proves to work. It may be possible to gain access to hash codes of passwords (such as are used in the authentication process on unix systems, albeit modified with a salt value), in which case it can be possible to test hashing likely passwords until one that produces the same hash is discovered. There are commonly

2 Though admittedly a password like *qwerty* would be more prone in an over-the-shoulder eavesdrop.

available tools that implement such attacks on passwords, either based on a dictionary attack or a pure brute force attack. Processes to discover passwords in this way can be computationally expensive, but given enough computing resources and enough time even moderately strong passwords could be discovered. Regularly changing passwords assures that the password is changed before a long running process has a reasonable chance to discover it.

Losing your password is not like losing your keys, i.e. if you lose your keys you will soon discover the problem, whereas it may be possible to make illicit use of a compromised password without your knowledge. It is a safer strategy to assume that all passwords will with time be compromised, and therefore subject to regular change.

Password authentication methods are known to be a relative weak spot in system security insofar as they rely on the security awareness of the individuals who create and keep their own passwords. There any many methods by which a password can be discovered, including those mentioned above. Changing a password regularly is at least a method of damage containment should a password be revealed, i.e., illicit use of that password is limited until such time as it is next changed.

Many student arguments were only vaguely related to the motivation to change passwords. For example, it might be argued that users might accidentally give away there passwords to social engineers, and for that reason it is a good idea to change the password. I regard this to be a very weak line of argumentation since in such cases it is clear that the most important measure is to not give away passwords. Arguments that show the unavoidable weakness of old passwords were regarded as the best.

Problem 4

> Describe each of the following IT security related terms. Also, for each of these terms further illustrate the concept by choosing a closely connected IT security concept and explaining the relationship between the concepts. Furthermore, give an example of an application of these tools/threats/concepts. Give concrete examples wherever possible. Structure each of your answers with headings *description*, *relationship to [your chosen related concept]*, and *example.*
>
> Some students may find it helpful to use the pre-printed problem 4 answer sheet for their answer. Those who choose not to should take care to follow the above instructions extra carefully.
>
> - Covert channel
> - Non-repudiation
> - Worm
> - Pseudo-anonymous remailer

Covert channel

Description

The Bishop definition is “A *covert channel* is a path of communication that was not designed to be used for communication” [Bishop05, p288]. One form of this is as a secret path of communication that piggy-backs on another, well understood and known communication path, such as hiding messages in the output of a web site.

Relationship to steganography

Steganography is the collective name for methods by which the fact that a message is being sent is obscured. The similarity between the two concepts is so great as to be confusing. They can be seen as two ways to describe the same phenomenon; whereas steganography is a collective name for one form of secret communication methods, covert channel is more concerned with the communication channel perspective. Steganography is used in the implementation of covert channels. A systems administrator who wants to check that there is no possible information leakage from one system to another might audit the system from the perspective of what possible communication channels

exist, and therefore look for possible covert channels, rather than considering the dangers of steganography.

Example

Two virtual machines running on the same hardware may be assumed to be well segregated and unable to transfer any information between each other. However, since they ultimately share the same cpu it may be possible to establish a covert channel. Run a process on one of those machines that loads the cpu in a predictable manner. Another process on the second virtual machine may be run that notes changes in cpu usage and can interpret them as messages from the first process.

Non-repudiation

Description

Non-repudiation is a general service that can be supplied by security systems. It ensures that subjects in a secure system can be shown to have taken certain actions, i.e. it cannot be denied that they have taken such actions.

Relationship to authentication

Authentication is another general security service where an entity in the real world is bound to a subject of a secure system. Authentication can be said to be a necessary pre-requisite to non-repudiation. Without a secure binding of entities of the real world, no actions of subjects within the secure system would be attributable to real agents.

Example

A teacher denies having received a student's hand-in and accuses the student of being mistaken about having sent it in on time. Luckily the message system used provides a copy of the sent mail for the student to keep, but also digitally signs the email together with a secure time-stamp of the time that the system received the message from message for delivery. When that signature is verified as having come from the system the teacher can no longer deny that the student fulfilled her obligations. The system, or the teacher may have misplaced the email, but it cannot be denied that the student sent it properly.

Worm

Description

According to Bishop's definition:

“A computer worm is a program that copies itself from one computer to another” [Bishop05, p373] We may assume that this definition (in common with other definitions from the same chapter on malware) assumes an element of ill intent. A worm is commonly programmed with the ability to exploit security vulnerabilities in order to spread itself from one computer to another.

Relationship to computer virus

Some definitions of the term computer virus will cover many kinds of malware, However, the more precise definitions will usually establish that a computer virus attaches itself to program hosts. In that case we can say that while viruses have programs as hosts, worms will not attach themselves to a program but to the system itself. Since worms have the ability to spread themselves from system to system, without relying on a a host program to first be activated, they will commonly spread much faster than the classic computer virus.

Example

The so called *Morris Worm* infected the Internet while it was still young in 1988. It exploited

known vulnerabilities of Unix systems, including a very rudimentary dictionary attack. Though it was not written to be harmful, accidental side effects caused it to become an effective denial of service attack for a significant portion of the Internet of the day, and an estimated 10% of Internet connected machines were infected.

Pseudo-anonymous remailer

Description

"A *pseudo-annoymous* (or *pseudonymous*) *remailer* is a remailer that replaces the originating electronic mail addresses (and associated data) of messages it receives before it forwards them, but keeps mappings of the anonymous identities and the associated origins." [Bishp05, p27]

This is therefore a means to anonymously send emails to a party, who can then reply to such an email. The reply is routed to the pseudo-anonymous remailer, which then maps the original sender's address into the message, and forwards it there.

Relationship to type 1 remailers

A *type 1* or *Cypherpunk remailer* also facilitates the sending of anonymous emails. In contrast to the pseudo-anonymous remailer it does not replace the sender address with a pseudonym but simply strips all sender information from the email header fields. The type 1 remailer therefore keeps no mappings between sender addresses and pseudonyms. It is not possible to reply to such anonymous messages.

Example

Possibly the most famous of pseudo-anonymous remailers was anon.penet.fi. During the 1990's it operated a well known service that was used for anonymous email services by senders from all around the globe. After several incidents where the site was subpoenaed to reveal identities of email senders by providing information of the mappings that it kept, those who ran the service decided that since anonymity could not in practice be guaranteed the service was discontinued.

Problem 5

> Saltzer and Schreoder's principles for the design and implementation of security mechanisms are summarised as:
>
> - The Principle of Least Privilege
> - The Principle of Fail-Safe Defaults
> - The Principle of Economy of Mechanism
> - The Principle of Complete Mediation
> - The Principle of Open Design
> - The Principle of Separation of Privilege
> - The Principle of Least Common Mechanism
> - The Principle of Psychological Acceptability
>
> Pick the three principles that you consider would be most important for the design of a secure and efficient firewall, and fully motivate the choice that you make including reasons why other principles are assumed to be less important.

Pass marks were given where answers showed an understanding of the principles, even if there was some minor confusion on which principle was which. Good marks were given where it was clear that there was a correct understanding of the principles and how they apply to firewalls. High marks were given where convincing arguments for the top three were given.

A complete analysis of how the principles apply to firewall design is far beyond the scope of an exam question. Therefore there are many possible ways to briefly reason on the subject, and if the

arguments were consistent, clear and reasonable, they were good enough for this problem. The following suggested answer is therefore to be regarded as only one possible way to reason.

Viewing a firewall from a functional point of view (rather than from e.g. its administration point of view) its main purpose is access control of network traffic, both incoming and outgoing. Let us begin by relating the principles that are more directly associated with access control.

Firewalls will not normally be endowed with enough information to make complex decisions on the nature of the processes that are communicating through them. They have relatively simple criteria to decide if a port should be open for a communication stream, or not. In this respect there does not seem to be enough room to apply the principle of least privilege (whereby a process should be given only those privileges need to complete its task). The principle would appear to be of lesser importance here.

The relative simplicity of the access control in a firewall also suggests that the principle of fail-safe defaults will less relevant. The principle says, in summary, that white lists are better for security than black lists. But it will be very difficult to characterise the access control only in terms of what kind of traffic is to be allowed in a white list. It will often make more sense to also describe what should be refused access in a black list.

The principle of least common mechanism requires that mechanisms used to access resources not be shared. This might be interpreted to mean that the firewall should not share hardware with other software, and this seems to be good general advice, except that for personal firewalls their design and purpose is to run on the same hardware that they protect. It seems to be difficult to hold this as a general principle for firewalls in practice.

The principle of separation of privilege states that one should not grant permissions on the basis of a single condition. It may be a good idea to grant access to a port based on several specific conditions, such as that a certain port is only open for a process that can show that it is from a trusted address, is not previously known to have executed a denial of service attack, and requests an SMTP port. However, in many situations the conditions for access will be simple, and complicating them for the sake of this principle does not seem motivated. This is once again a principle that is difficult to apply consequently for a firewall.

The one among the access control related principles that does seem generally applicable, and indeed the very essence of what a firewall is, is that of complete mediation. It is very important that a firewall filter all traffic without exception. There should be no loopholes in network communication that a firewall does not check. Admittedly there will be depths of complexity in dangerous traffic that it will be beyond the ability of the firewall to check, but that does not mean to say that it should not check all traffic within the limits of its abilities.

It is difficult to argue convincingly here for Open Design. The principle states that the security of a mechanism should not depend on the secrecy of its design or implementation [Bishop05, p204]. But surely there is little in a firewall design that could be enhanced by attempting to keep the design secret, so this becomes a non-issue[3]

From the point of view of the firewall user, psychological acceptability is surely paramount. If a firewall causes network to noticeably slow down, or else block traffic that is legitimate, then users will not stand for it, and ultimately (if they are able) disable the firewall. If we were to shift perspective to the administration of a firewall for a moment, even here the psychological acceptability factor is extremely important. A major factor in the failure of firewalls is from configuration errors by administrators. This indicates that if a firewall is desgined to ease correct configuration it will be a more secure tool.

This discussion leaves only the principle of economy of mechanism. A firewall should be an

3 Some answers held the line that open design is a good way to get free help in bug-hunting, but that is surely a very tenuous principle on which to base the design of a very vital security tool. The “many-eyes” principle has so far not been shown to be reliable enough (e.g. bugs are not infrequent in linuxes despite the code being entirely open). The principle of open design does not suggest that we should be open because “many-eyes” improves security. It says that trying to implement security by hiding things is generally not a good idea.

effective, efficient and bug-free security tool. This indicates that complexity of design and implementation should be kept to a minimum. Firewalls are not the kind of tool where one would like to see feature creep at the expense of well written, economic code[4].

From the above it is motivated that economy of mechanism, complete mediation and psychological acceptability are the three most important factors for the design of a secure and efficient firewall.

Reference

Bishop05 Matt Bishop, Introduction to Computer Security, Addison Wesley, 2005.

4 It is clear from a number of answers that some students are still equating economy of mechanism with simplicity of use. The two should not be confused. I claim that in practice the two ideas are in conflict – in order to make a program easy to use one will most likely have to introduce very complex coding.