DSV SU/KTH ***SEC:I***
Principles of Computer Security
Autumn Term 2012

Alan Davidson

Suggested Solutions to the Exam 2012-10-31 and Comments on the Marking

The answers suggested here may be briefer than would normally be expected from students. I have tried to summarise the most important aspects of problems so that students may compare the content their own answers to these. Lack of time prevents me from doing more.

Problem 1

Define the terms *Security Policy* and *Security Mechanism*, and motivate why it is important to differentiate between the two concepts.

According to the definitions from the course book [Bishop05]:

A security policy is a statement of what is, and what is not, allowed. [p7]

and also a later refinement of this definition

A security policy is a statement that partitions the states of the system into a set of authorised, or secure, states and a set of unauthorised of nonsecure, states. [p45]

and

A security mechanism is a method, tool, or procedure for enforcing a security policy. [p7]

I do not normally expect students to remember such definitions word for word, so any answer that could convey similar concepts would be given credit.

We understand from the definitions that the mechanisms are only our best attempt at supporting what the policy describes. There will very seldom be a perfect match between the policy and the mechanisms that enforce it. This implies that there will be some unauthorised states that the mechanisms may not able to protect against, and conversely some authorised states that mechanisms may disallow.

A common misconception is that it is the mechanisms that define the security of a system. By differentiating between the two concepts one can understand that just because and action is not disallowed by the mechanisms that does not mean to say that it allowed.

A small number of answers had the same worrying misconception, i.e. that languages and models such as Bell LaPadula should in some way be examples of mechanisms. I took that to be evidence of quite a basic misunderstanding,

Problem 2

During the course we have made a clear division between methods for authentication and for access control. However, for some security mechanisms it may not be as clear whether they can be classified and described as authentication mechanisms or access control mechanisms.

Consider the key-cards that are distributed to staff and students at the DSV department to allow access to the building and its rooms.

Characterise the key-card system in terms commonly used to describe authentication methods, and then in terms commonly used to describe an access control mechanism. For each of these viewpoints motivate why the system is best characterised in the terms that you have used.

As an authentication mechanism one might characterise the key-card system as both authentication through something the entity (card holder) knows in terms of the PIN code, and something the entity has in terms of the plastic card. Note how for some rooms and at certain times it is enough to swipe a valid card in order to gain access. However, for some places and at some times a higher level of security is deemed necessary, and then both the card and the PIN code are required to gain access. In this case we see how the authentication mechanisms utilises the principle of Defence in Depth, with a close association to the Saltzer and Schroeder's Principle of Separation of Privilege.

Other ways to classify authentication methods are s*omething that the entity is*, or *where the entity is*. There is a photograph of the card holder on the card, so one might claim that there is an element of who the entity is, i.e. what their face looks like that links them to the card. But then we should question how this aspect is checked by the authentication system. It is only humans that can view this picture and make the connection, and in practice there is no part of the system that requires human interaction. Perhaps if we were all required to wear our cards visible while in DSV rooms, and if the policy that we all sign required us to monitor and report if we see someone with a card that is not theirs, then we could call it part of the system.

One might make the case that part of the authentication is *where the entity is* given that the rooms that are being accessed have a physical location, and one must be outside the door to authenticate. Though some might develop this as an argument I suggest that the physical location is an intrinsic property of the system, rather than a useful attribute for authentication. The card and PIN code are not more correct authentication methods because they are in the Forum building (Well made arguments to the contrary were of course given credit too).

It seems to be a fairly simple protocol, and though the subject is sometimes prompted to enter a PIN code with the way that the keypad symbols light up, if we were to view it as a challenge-response protocol it is such a simple and predictable protocol that it does not seem valuable to analyse the system in such terms.

In terms of access control we can view the system as an Access Control Matrix (ACM) where the subjects are users and the objects locks on the corridors and rooms. The obvious rights are *open lock*, but sometimes you may notice that teachers can *set room to unlocked*, and *set room to locked,* for example to allow and disallow access to lecture halls.

The course has taught us that access control is seldom implemented as a matrix. We might therefore consider whether the lock system is best characterised as an Access Control List (ACL) or Capabilities. As an ACL the rights are primarily associated with the objects, which in this case is the locks. Is it likely that each lock has a list of users, or groups of users, associates with the rights for such subjects? It may not be easy for a student to tell from experience. Being a member of staff my experience is that changes to the system are made at a central system but that it can take a very long time for those changes to propagate to the actual lock. This suggests to me that it is indeed very likely that the information is downloaded to the locks.

For the system to be capability based the rights would in some way be associated with the users, or rather in the cards that users hold. The card would then list rooms (or classes of rooms) and the rights for each of those objects. Since the only means for the card to contain such information is on the magnetic strip it does not seem very likely as this is such a simple mechanism. Moreover, one would expect that any changes to rights (such as being allowed special access to the sandbox laboratory in 408 during the SEC:I assignments) would have to involve some change to the configuration of the card. Since students did not have to put their card into a card writer to gain new access, one could surmise that it is very unlikely that the system is capability based.

One might make the case to the access control system being role-based. IT seems that there are likely to be a number of basic roles: student, teacher, technical staff, cleaning staff, etc. Our arguments motivating this might be that some students become teaching assistants, and it would make sense for a role-based mechanism to aid in quickly switching those rights. Without closer inspection of the workings of the system it is difficult so say more.

We have also used terms such as DAC, MAC, and ORCON to describe strategies to manage the changing of rights. In broad terms, with DAC the object owner is free to define the rights, in MAC it is the system owner, and with ORCON the originator of the object. The originator of the lock system is the designer and manufacturer of the lock. It does not make sense to assume that they would control the rights. To differentiate between DAC and MAC we would have to know who owns the individual locks, and how that differs from the owners of the whole system. I would suggest that the difference is of very little significance here, i.e. it is most likely that DSV can be viewed as both the owner of the locks and of the system as a whole. In this example we cannot therefore distinguish between DAC and MAC. I would not normally expect SEC:I students to manage this depth of insight during the exam. Sensible, motivated discussions that suggested either DAC or MAC were given credit.

Some examinees confused which entity was to be authenticated in their discussions. Since we are talking about access control to rooms, it makes sense to say that the entity is the person. In that context it is not the card that is authenticated, but the card holder.

Problem 3

> One method that can protect a system from the effects of malware infection is to ensure that any changes to software on secondary memory are detected and reported so that the system administrator can effectively respond. Such a method would work even if the infections is caused by previously unknown malware.
>
> a) Suggest in outline a reliable mechanism that can detect changes to software.
>
> b) Discuss possible weaknesses that this method has as useful malware protection.

a) This is clearly a problem of the integrity of programs. During the course we have touched on some means to provide integrity services. This problem does not seek to directly uphold integrity (e.g. by ensuring that only trusted processes have access to the resources), but to detect changes. Therefore I would claim that the given answer to this problem is:

The mechanism might keep cryptographic checksums or hash values of programs and regularly and automatically check them. When a discrepancy between the hash value of software on disk and the hash value that is recorded in a database is detected then the system administrator can be alerted. Such a hash value might be inserted into the database during the first installation of the software, or even before fetching the software. Many online sources of software display hash values of that software in order for customers to ensure that the software that is downloaded is the same as the software at the source.

Some answers that were along these lines suggested that the software need not be checked regularly but instead only before it is executed. The strategies may be a question of personal taste. My own motivation for regular checks is that it is less annoying to be warned during a security check than it is just as one wants to run a program, only to be told that an infection is suspected and the program must not be run.

Answers that suggested keeping track of file sizes were considered very weak. We know from the course that recording and checking the size of a file is hardly a reliable mechanism since among other things the malware might be able to change the file but without changing the file size. Such an answer shows that the examinee has entirely missed the connection to integrity services, and the role that cryptography can play, which was after all a major theme of the course.

Unfortunately a number of examinees attempted a trivial answer "Use antivirus software"! I cannot accept this as a valid answer since it is not a sensible answer to the problem "Suggest in outline a reliable mechanism that can detect changes to software".

b) We can imagine several reasons why this strategy might prove to be less effective:

i) If this strategy is well understood by malware creators then it may be difficult to protect the database of hash values or the process that compares them. We need to be able to trust the hash values, so we would also need to have trustworthy integrity checks on that database and process so that malicious software cannot fool the strategy by altering the hash values or subverting the check mechanism.

ii) The current 'culture' of software distribution allows for frequent updates of software, sometimes to include new features, but very often in order to fix problems that have been discovered after the software has left the developers. Many times the problems will be security problems, which means that not updating means that security will suffer. The hashcode mechanisms will presumably have to be designed so as to allow for software updates, and thereby hash value updates, without raising bogus alarms. Such a fix would help the psychological acceptability of such a mechanism, but also open up another possible means for malware to bypass the checks.

iii) The distinction between programs and data is of importance, since we might normally assume that data changes are frequent, whereas program code tends to be immutable. However, the clear dividing line between program and data has become all the more muddied over the years. For example, a modern formatted text document might include instructions to the program that manipulates it, which ostensibly makes it both data and at the same time an interpreted program.

iv) Regular automatic scans and checks require computing resources. Were the checks to make a noticeable difference to the user's of a system the user may be less inclined to allow such checks to run.

v) If the mechanism only checks the integrity of legitimate programs it can only be used to detect the existence of computer viruses. Other kinds of malware such as worms would not be detected since they (by definition) do not attach themselves to programs.

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.

Please note that in general a 50% complete answer will be required to obtain a pass mark for this problem:

- Hard Certificate
- Saltzer and Schroeder's Principle of Psychological Acceptability
- Phishing
- Buffer Overflow

Hard Certificate

Definition

A hard certificate is a certificate that is kept on a hardware device that is specially designed so as to protect the certificate from being copied or changed, but still allows that certificate to be utilised for cryptographic operations. A certificate is a data structure that primarily contains one of the two keys in an asymmetric key pair. In the context of *hard certificate* the contained key is presumed to be a private key, which therefore puts high demands on the confidentiality and integrity of that certificate.

Relationship to Soft Certificate

A soft certificate is likewise one that contains the sensitive private key, but it is kept on the secondary memory of the system that uses that key. Soft certificates are generally kept encrypted on secondary memory until such time as they are needed, affording them some security. However, if the secondary memory can be accessed and the cryptographic key discovered the soft certificate is compromised.

Example

During a lecture I showed a device that is used to authenticate bank transactions. This device is a smart card reader that is connected to a computer with a usb connection. A particular kind of credit card (one with BankID on it) is inserted into the reader, and a PIN number is typed into the device whenever an authentication is required. The chip on the smart card contains a hard certificate. The certificate itself is never available to the computer or the card reader. The required cryptographic functions are executed by the chip on the card itself, which is the only component that has access to the certificate.

Saltzer and Schroeder's Principle of Psychological Acceptability

Definition

Though a word-for-word quotation of Bishop's definition is not expected, answers should give the gist of:

The principle of psychological acceptability states that security mechanisms should not make the resource more difficult to access that if the security mechanisms were not present [Bishop05 p206].

Relationship to Saltzer and Schroeder's Principle of Economy of Mechanism

To summarise the principle of psychological acceptability, the mechanisms should make security easy for the user. The principle of economy of mechanisms says (in essence) that the mechanisms should be simply constructed. Simply constructed and simple to use are not necessarily the same kind of simplicity. Indeed they may be in conflict given that mechanisms that make security easy for the user will more than likely be very complicated in their construction. A command line controlled operating system is relatively simple in its construction, but most would find its security mechanisms difficult to interact with, so we have extremely complicated window, pointer, icon etc systems to allow one to more simply interact with them.

Example

A firewall that is difficult to configure because the interface is unintuitive is likely to be insecure. Likewise, if it requires so many resources that it makes the computer slow down considerably, then a user might decide that the security win is not worth the loss of fast communications.

Phishing

Definition

Phishing is a form of social engineering that is perpetrated through electronic media. Its name alludes to the principle of fishing in that one casts out attractive bait ostensibly in a non-directed manner in the hopes that a victim will act upon it. Phishing normally involves some kind of spoofing of a legitimate party in the hopes of gaining private information from uses.

Relationship to MITM attack

The Man/Monkey in the Middle attack involves redirecting traffic that should be passing between two parties through a third malicious party. The traffic can thereby be eavesdropped and manipulated. This is in contrast to phishing where (in one common form of the attack) a user is fooled into visiting a website that is in the control of the attacker which implements a version of a website that the user can mistake as a legitimate website. In contrast to MITM, phishing does not involve the legitimate website in the communication.

Example

A common example of phishing is that an email is sent out (most probably as spam) to a large number of recipients where the contents are a spoof of a warning from an email account administrator, saying that for some reason the user's email account will be closed down if action is not taken. The email may contain a URL that takes the user to a site that is also a spoof of a legitimate site, but that asks the user to enter information that the perpetrator can use to their own advantage. That might be as simple as the users password to their email account. Given that attackers can continue to perpetrate any number of attacks that could cost the user dearly.

Comment: Most students did not distinguish between phishing and spoofing or social engineering in general, so to be strict most did not really know what phishing was, even though I am sure that we have all experienced it. Such exact definitions were however not required in order to gain credit.

Buffer Overflow

Definition

Buffer overflow is a vulnerability that can affect programs that are written in languages that do not have automatic memory management but where memory management is at the control (and responsibility) of the programmer. The problem arises where a program, in writing to a memory buffer writes beyond the limits of that buffer, thereby overwriting adjacent memory. One of the most dangerous effects of such a vulnerability is that an attacker may supply specially crafted data that the program copies into memory that thereby cause the program to execute instructions that are included in that crafted data. The effect can therefore be that an attacker can execute any code that they like with the same privileges as the attacked process. Program checks of input data to ensure that it will fix into buffers before copying to them is an effective means of avoiding many buffer overflow problems.

Relationship to Code Injection

Another general class of vulnerability is code injection. Here too an attacker can craft data that may effect the execution of a program. The difference is that in code injection the problem comes from allowing input to the program to be interpreted as program instructions. Code injection therefore only affects interpreted languages such as SQL or python.

Example

A program requests a year to be entered by the user, and assumes that the user will only enter four digits and thereto allocates four bites of memory. The program does not however check that only four characters are entered but continues to copy user input to memory one character at a time, moving forward in memory one byte at a time until the end of user input is met. The memory adjacent to the four byte buffer will be overwritten if more that four characters are input. What effect this has can very from bringing about a memory exception to manipulating program execution.

Comment: Several answers gave the impression that buffer overflows most often or always cause an execution error that aborts the process. This is far from the truth. I usually characterise this as being the best possible i.e. least dangerous outcome. Far more dangerous is that buffer overflows might allow an attacker to execute any code that they want to with the same privileges as the attacked process. Somehow a large number of you got the idea (perhaps from a misunderstanding of something said at lecture?) that buffer overflows main problem is that they can crash or halt processes. **This is a very serious misconception**. Directed buffer overflow attacks are far more serious than that!

Problem 5

Though IT technology in general has introduced many threats to personal privacy, tools can and have been developed that enhance our ability to control what information about us is spread to others.

a) Describe in general terms what *remailers* do to enhance privacy, and explain why such services can be assumed to be both good and bad for society.

b) Cryptography is an important element in remailers where a high level of privacy protection is necessary. Explain why and how.

a)

Remailers strip all information that can identify the sender from the headers of emails in order to make the messages anonymous. Note however that if the users themselves were to include information that can lead back to them in the body of their email then no remailer will help in stripping that info.

The simplest types of remailers keep a mapping between the sender's email address and a pseudonym, thereby allowing replies sent to that pseudonym to be returned to the original sender. Such pseudonymous remailers are no longer popular since the demise of the penet.fi after the site was forced by Finnish courts to reveal details of their address mapping tables.

More advanced remailers all use networks of servers that forward the email through each other . This is done to ensure that no one point in the network will be the goal for intrusion attempts that would thereby make the system insecurity. In order to break the anonymity one would have to crack a large number of the servers.

By allowing anonymous emails the system affords users some degree of information self-determination, at least in terms of what information about them is automatically included in email headers.

Bishop has a section in the course book *Anonymity for Better of Worse* [Bishop05 p230] which I refer the reader to for a discussion with several ideas that examinees could use to relate to the good and bad of remailers.

Several answers confused privacy and anonymity with confidentiality. Such answers were considered to show a lack of understanding of the privacy discussion that is part of the course.

b)

As stated above, the idea with using a network of remailers is that no one remailer should have enough information so that a compromise of any one system (or for that matter all remailers except one) could reveal enough information to identify a sender. If someone were to sniff traffic between remailers there should not be any possibility to analyse data and thereby identify where data originates from. Encryption of the messages between all parties in an anonymising network ensures that such analysis will be intractable.

A common scheme is that the user who is to be kept anonymous will encrypt the recipient's address and the body of a message with the public key of the last remailer in a chain of remailers. This ciphertext is then in its turn encrypted (together with the address to the last remailer) with the public key of the next to last remailer, and likewise this is encrypted in sequences leading all the way back to the first remailer used. This allows each remailer to 'unpeal' the outer layer of encryption and forward the contents to the next remailer, or ultimately the recipient. In this way any one remailer can only know where the message they received came from and where they send it.

Note how anonymity is preserved even if the message from the last remailer is sent to the recipient in clear. That tells us that confidentiality of the message itself is not the goal (as some students mistakenly presumed).

For part a the best answers are concrete explanation of what actions remailers take, thereby showing that you understand what a remailer is.

For part b, one must show an understanding of what a remailer is to have a sensible discussion, and the dangers of network sniffing.

Reference

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