CS140 Winter 2006 Final Exam Solutions

(1)

In class we talked about the link count in the inode of the Unix file system being incorrect after a crash. The reference count can either be either too high or too low.

(a) Which of these two conditions (high or low) is considered more serious? Justify your answer.

The reference count being too low is the more serious problem because in this case blocks that are still in use may be added to the free list. This may cause security holes like the password file being stored in "free" blocks and then relocated to a user writable file.

(b) Describe the worst-case problem of the less serious of the two conditions.

The worst case problem of the reference count being too low are memory leaks. Since the inode and its blocks will not be reclaimed, the disk could fill up a lot sooner than it should, so a lot of space is wasted.

(2)

A recent CNN news headline reported a potential Internet denial of service attack that used "the Internet Traffic Cop" to launch the attack. The article itself refers to the Traffic Cop as the top-level Internet Domain Name Servers (DNS). The dictionary defines Traffic Cop as "a policeman who controls the flow of automobile traffic." Do you think Traffic Cop is an appropriate metaphor of the DNS? Justify your answer.

A traffic cop "controls the flow of traffic". DNS servers simply provide a mapping from a name to an IP address. I would say this is a not an appropriate metaphor, although students could successfully argue it both ways and get credit. They are dissimilar since traffic cops do not provide any sort of mapping; they do not convert one form of part of an address to another, like a root DNS server does. Also, DNS servers do not control the flow (i.e., the speed or volume) of traffic they encounter like a traffic cop does. Finally, the DNS servers are not even needed at all if the IP address of the destination is known, or in the cache, so it is hard to argue that they control much of anything.

Some people argued that the analogy was OK because DNS servers could control the flow of traffic in the sense that the mappings they give tells the traffic where to go, which is an OK, but a weaker link between the two entities. For one, this is not controlling traffic the way a traffic cop controls traffic (which controls the rate of traffic, not its final destination). Secondly, traffic can bypass the DNS "control" if it doesn't need the name to IP mapping in the first place.

(3)

In order to safely run software in a virtual machine, a virtual machine monitor (VMM) must intercept modifications of privileged CPU state such as the interrupt disable flag.

(a) Describe why this interception is necessary.
(b) Describe why reads of this privilege state as well as writes need to be intercepted.

(a)
The interception is necessary because the virtual machine should never have access to the true machine hardware resources. The goal of using virtual machines is to virtualize the resources, so that each VM thinks it has complete access to its own hardware. If a VM could change privileged CPU state, it would break this virtualization. Also, the VMM could lose control of the resources. This could occur, for example, if a VM disabled the true hardware interrupts, preventing the VMM from reclaiming the processor. To prevent this, the VMM must always intercept access to privileged state and virtualize the hardware effect of any instruction which modifies privileged state.

(b)
Again, the CPU state must be virtualized for each VM, otherwise different VMs would be sharing resources, breaking the illusion that each VM has exclusive access to the hardware. Any changes to CPU state from one VM should not be read or have anything to do with the CPU state of another VM. Thus, a virtualized copy of CPU state for each VM is needed. When a particular VM wants to read the hardware state, it must be caught by the VMM to provide the corresponding virtualized state. It cannot read the true hardware state either, because it would not be consistent or correspond to the virtualized state of the VM.

(4)

The following are names given to several fields in the Internet protocol (IP) packet header. For each field describe its likely purpose:
(a) Header checksum
(b) Time to live (TTL)
(c) IP fragment offset
(d) Protocol

(a) Header checksum

This is calculated over the IP header to ensure its integrity. (i.e. the packet header hasn't been corrupted during transmission)

(b) Time to live

This is an integer which represents the number of hops a packet can take to reach its destination. It is decremented at every hop, and the packet is discarded if the TTL reaches 0. This prevents a packet from looping around in the network forever.

(c) IP fragment

Since the MTU is different for heterogeneous networks, an IP packet may need to be fragmented if its size is bigger than the MTU for a network. (link layer) This field indicates the position of the fragment's data relative to the beginning of the data in the original datagram, which allows the destination IP to reconstruct the original datagram.

(d) Protocol

Indicates which upper layer protocol receives incoming packets after IP processing is complete. e.g. UDP or TCP.

(5) – Nazia

Why can an Ethernet get away with a flat address of 48bits while the Internet with a smaller 32bit address needed to go to a hierarchical organization?

Since the Internet needs to be able to provide connectivitiy between any two end hosts, each host needs to be uniquely identifiable. Since the routing tables grow with the number of nodes, if the Internet had flat addressing, the number of entries in a routing table would be around 2^32. Meaning every router must have an entry for every other node. Since such large routing tables are impractical (imagine the lookup time even if you had the space to store it) the Internet had to have hierarchical addressing. As for the Ethernet, it only consists of nodes in a local network, and these are not of 2^32 magnitude, therefore flat addressing (along with ARP) works fine in this case.

(6)

What is meant by the trusted computing base (TCB) of a system? Give an example.

The trusted computing base (TCB) of a system is all the parts (software or hardware or both) that enforce the system's security policy. The "trust" is that this security policy is bulletproof, in that any malicious activity or unexpected failures will not compromise security. While a wide variety of definitions were accepted, students typically lost points for:
-Using a specific example as the definition itself without further clarification, ie. a trusted encryption server or an OS kernel.
-Not mentioning that the TCB is responsible for a security policy.
-Not clarifying "trust".

(7)

One of the hassles of the web today is all the websites that require you to use a password to authenticate yourself. If you use the same password for every website then one bad website can access your other websites. Suppose you had the same password for every website but instead of directly using the password you compute a secure hash of the password concatenated with the website name to form the real password. In other words a website's password would be set to SecureHash(password concat websiteName); Would this use of a secure hash prevent a bad website from gaining access to other websites? Justify your answer.

This secure hash method will generally prevent bad websites from gaining access to other websites. There are two main points to consider. One is that each password is "salted" with the website name, meaning one website's password cannot be used to access another website. The other is that, if we trust the hash is secure (irreversible), a bad website will not be able to construct the password of another website even if our hashing algorithm is known, save for a dictionary attack or other elaborate hash-breaking technique. Many answers were accepted, provided these points were mentioned in some form and no incorrect information was given.

(8) (

Could a workload on a file system with write-ahead logging that used physical blocks ever use less log bandwidth that a similar file system that used logical logging? Justify your answer.

Log bandwidth refers to the amount of data transferred to and from the log portion of the file system, assuming some workload of file operations in a given time period. In general, logical logging will use less bandwidth than physical logging when the metadata operations can be described more concisely than the physical representation of the metadata (for example, "create file A filled with zeroes" is smaller than a 512-byte inode sector for file A). However, one can contrive a situation in which the logical description of an operation might be longer than the space it would take in a physical log.
For example, if our logical log has an entry "create file A filled with zeroes", but our disk has tiny sectors and can fit all the relevant metadata into a sector or two, the logical logging of this operation may take up more space, hence use more log bandwidth. Many answers were accepted provided they stated assumptions and gave no incorrect information.

(9)

For each of the following security attacks say if public key encryption can help prevent the attack. Be sure to justify your answer.
(a) Abuse of valid privileges
(b) Spoiler/Denial of Service attack
(c) Listener or eavesdropper attack.
(d) Trojan Horse
(e) Buffer overflow attack

(a) Abuse of valid privileges
No. Valid priveleges by definition are things the user is allowed to do, so no type of encryption will prevent them.

(b) Spoiler/Denial of Service attack
No. Overwhelming system resources has nothing to do with what the data consists of, but rather the volume of data.

(c) Listener or eavesdropper attack.
Yes. This is the reason we have public key encryption. Someone sends data encrypted with a readily available public key to you. It can not be read along the way because only you have the private key, which allows you to decrypt the message.

(d) Trojan Horse
No. A trojan horse is malicious software hidden inside software that is already trusted. There is no way to prevent this with public key encryption.

(e) Buffer overflow attack
No. This attack takes advantage of a flaw in software, and encryption can't

prevent this.

Note: some people said authentication could solve, potentially, a DOS attack (by filtering unwanted agents) and a trojan horse (by only accepting truly trusted sources of software). Credit was given IF the person specified that encryption with a private key was required to provide authentication. Encrypting with a public key, as it says in the question, cannot provide authentication.

(10)

Assume your Pintos file system needed to update a multiple sector long data structure on disk.
(a) Describe how you might arrange to detect if your system failed in the middle of writing the data structure.
(b) Describe how you might build your system to recover from such a failure

(a) A simple way to detect that the system failed in the middle of updating a multiple sector data structure would be to embedded a robust checksum of the data structure that is updated on every write. Should we detect that the recorded checksum doesn't match the current checksum of the data then the most likely cause is a crash during an update.
(b) In order to recover from such a failure, you need some kind of replication. Simply have two copies of the data structure that you alternate between would work. After a crash you simply choose the newest copy with a correct checksum. It is also possible to use a log and write-ahead logging. By writing to the log first before writing to the data structure you can always roll forward after a crash.

(11)

Describe which file system buffer cache write back policy you would suggest for a file system that employed write-ahead logging for both user data and metadata. Justify your policy.

If the file system uses write-ahead then we need not be in any rush to write back blocks from the buffer cache. The log can be used to recover any changes made in the buffer cache but not written to disk. Some kind of aggressive write back policy such as write-back when the block is evicted or when the log wraps can be used and will likely give better performance than a write through policy.

(12)

Assume your system uses SCAN or elevator algorithm as a disk-scheduling algorithm. How would you modify this algorithm if rather than having a single disk your storage device was a RAID-4.

A SCAN disk scheduling algorithm assumes you have a single disk arm for which you are trying to optimize the motion. Clearly this is not the case any RAID devices that stripes the file system across multiple disk. If you had knowledge of the mapping of blocks onto the disks, you could do a queue per disk and schedule within those queues.

(13)

For each of the following file descriptor data structures, order them from the most to the least amount of space consumed when mapping a very large file.

(I) Index files

(II) Linked files

(II) Extent base files.

Repeat the ordering but this time for very small files (less than a disk sector in size).

In this question, points were awarded basically for your justification, and your assumptions. However it was important to note that extent based files consume the least amount of space when mapping any file. (small or large). Other methods can do as well under certain circumstances, but not better. However, if you argued fragmentation, some points were awarded for that. One possible answer is below:

For large files:

Index files

Linked Files

Extent base files

Linked files contain overhead for each block, whereas Index files have overhead for pointers to indirect blocks which contain pointers to blocks, and extent based files are just the base and bound.

For small files:

Index files

Linked Files

Extent base files

Index files contain overhead for the index, even if the file is small, linked files contain a pointer to the block, and then a pointer to the next block (null), this could be similar to the space required by the extent base files, under these constraints.