Operating Systems Design Exam 3 Review: Spring 2012

Paul Krzyzanowski
pxk@cs.rutgers.edu

Question 1

An Ethernet device driver implements the:

➡ (a) Data Link layer.

(b) Network layer.

(c) Transport layer.

(d) Session layer.

Network device drivers implement the data link layer.

Network protocol drivers implement network and transport layers. IP is an example of the network layer and is responsible for packet routing & delivery to machines. TCP and UDP are examples of the transport layer and are responsible for validating data, creating a virtual circuit (for TCP), and delivering packets to sockets. The session layer is managed at the user level.

Question 2

A virtual circuit will not guarantee:

➡ (a) Constant packet latency.

(b) In-order packet delivery.

(c) Reliable packet delivery.

(d) Packet-based data transmission.

A virtual circuit is software that tries to simulate the appearance of a circuit switched network. It does this by retransmitting damaged or lost packets and attaching sequence numbers to packets so that it can deliver data to applications in the order that it was sent. A virtual circuit has no control over the end-to-end latency in the network. Real switched circuits generally provide maximum bounds on packet latency.

Question 3

Packet encapsulation is useful for:

➡ (a) Protocol layering.

(b) Separating the header from the data.

(c) Encrypting packet data.

(d) Fault-tolerant packet delivery.

Packet encapsulation is the wrapping of one packet within another one (within another one…). For example, a TCP packet is encapsulated within an IP packet, which is encapsulated within an Ethernet packet.

Question 4

How are additional protocol headers added to packet data as it goes down the network stack?

[✓ (a) The socket buffer is allocated to be big enough to hold all anticipated headers.]

(b) Each layer of the protocol stack creates its own header and adds it via a linked list.

(c) At each layer, data is copied to a new buffer big enough to hold the existing data and new headers.

(d) A separate socket buffer is created to manage headers.

The idea of the socket buffer is to avoid re-allocating and copying data. When a socket buffer is allocated at the top level (sending data), it attempts to create a buffer that is big enough to hold data + the maximum size of anticipated layers in the stack (e.g., ethernet + IP + TCP).

Question 5

How many times is the data in a message copied from a user process creating it to it being sent onto the network?

(a) Zero times.

(b) One time.

[✓ (c) Two times.]

(d) One time for each layer of the network stack.

The data is copied once from user space to a socket buffer in the kernel space. Then it is copied a second time from the socket buffer to the device.

Question 6

Linux’s New API (NAPI) added the following feature to the network stack:

(a) Support for jumbo packets to support gigabit Ethernet transceivers.

(b) Support for IPv6 in addition to IPv4.

(c) A clean separation between the transport and network layer.

➡ (d) A combination of interrupts and polling to avoid a high rate of interrupts.

NAPI tries to avoid a high stream of interrupts from the network card during periods of high traffic. When the kernel receives an interrupt from a network device, it turns off network device interrupts and switches to polling (also called *soft irq*). As long as packets are available at the device each time the kernel polls, it keeps interrupts disabled. If no packet is available, polling is turned off and interrupts are re-enabled.

Question 7

A stub function in a remote procedure call:

(a) Is the implementation of the remote function on the server.

➡ (b) Provides the interface of the server function on the client.

(c) Is a placeholder function that needs to be filled in by the programmer.

(d) Is a local function that is called if the server cannot be contacted.

The stub function on the client is the function that the program calls whenever it wants to access the remote function. To the program, it looks like that remote function but is, of course, local. Its purpose is to take the incoming parameters, marshal them, send them to the server, wait for a response, unmarshal the response, and return back to the caller.

Question 8

A programmer using remote procedure calls (RPC) uses an interface definition language (IDL) to:

(a) Implement the remote procedures in an architecture-independent manner.

(b) Provide the kernel with the interfaces to a set of remote procedures.

(c) Marshal parameters for the remote procedure.

➡ (d) Identify the names, inputs, and outputs of remote procedures.

An interface description language (IDL) defines the remote functions and their parameters and return values so that an RPC precompiler can generate client stub functions (one for each remote function) and a server stub (that contains the main program that listens for network messages).

Question 9

Which caching approach yields session semantics?

(a) Read ahead.

(b) Write behind.

(c) Write through.

[✓ (d) Write on close.]

Session semantics mean that changes are not visible to others until the file is closed. Write-through violates this since it makes changes immediately visible. Write-behind violates this since it just adds a delay before changes are visible. Read-ahead is not applicable since it does not result in file modification.

Question 10

What operation is not possible on a stateless remote file system?

(a) Read ahead.

➡ (b) File locking.

(c) Getting file attributes.

(d) Appending to a file.

File locking requires keeping state: the server needs to know whether a file is locked or not. Read ahead is just a request for data. Getting file attributes is just a request for data. Appending to a file is just a request to append data to a given file. Note that NFS, originally designed to be stateless, did not support file appends. This was not due to the stateless design but rather to the fact that they did not implement a remote procedure to append data to a file.

Question 11

Under AFS, a callback from a callback promise:

[✓ (a)] Ensures that the client invalidates an obsolete version of a cached file.

(b) Is used by a client to inform the server that it is modifying a file.

(c) Informs a client that a lock has been released on a remote file.

(d) Tells the client that it is now allowed to cache data for a specific file.

The callback promise is made when a client downloads a file. This results in the server keeping state that a particular client may have a copy of the file cached in its file system. When the server gets an upload of a new version of the file, it contacts each client that may have a cached copy and sends them an invalidation message for the file – the callback.

Question 12

The principle of least privilege can best be enforced with:

(a) Stack canaries.

(b) Access control lists for all files.

(c) Cryptographically secure authentication

➡ (d) A sandbox.

The principle of least privilege is the policy of restricting any unnecessary services from a process. For example, if a process does not need to use the network, do not give it permission to do so. If it does not need to read any files from the /etc directory, do not allow it access.

Access control lists restrict only access to files and character/block devices. Moreover, they specify access policies per user and group. When a process runs, it runs with the user's ID and hence inherits all of the user's access privileges. With sandboxing, one can enumerate file, network, and device restrictions on a per-process basis.

Question 13

Privilege separation is useful in enforcing:

[✓ (a) The principle of least privilege.]

(b) Mandatory access control.

(c) Discretionary access control.

(d) Multi-level secure access.

Privilege separation breaks a process into two or more processes, some of which will be granted reduced privileges. That way, even if a process is compromised, the ability to do damage is limited.

Question 14

File permissions on Windows or Linux are an example of:

[✓ (a) An access control list.]

(b) A capability list.

(c) Mandatory access control.

(d) An access matrix.

Linux, Windows, and most operating systems employ file-based permissions, which are examples of an access control list.

Question 15

Return Oriented Programming:

[✓ (a) Allows buffer overflow attacks to run code even if stack memory is not executable.]

(b) Is a technique for injecting executable code into a buffer (also known as stack smashing).

(c) Is a programming technique that ensures that buffer overflow attacks cannot occur.

(d) Is where the compiler generates code at the return of a function that checks for buffer overflow.

Return Oriented Programming was created to deal with non-executable stacks. By changing the return address, you can redirect a program to go to a snippet of some other function. When it returns from that, you're again in control of the return address and can make it go somewhere else.

Question 16

Which technique cannot be used guard against buffer overflow attacks?

(a) Address space layout randomization.

(b) Stack canaries.

➡ (c) Signed software.

(d) Non-executable stack memory.

(a), (b), and (d) make buffer overflow attacks more difficult. (a) means that you cannot know addresses of functions in dynamically-loaded libraries ahead of time. (b) means that a function will check for a buffer overflow prior to returning. (d) means that you cannot inject executable code into a buffer.

Signed software validates the integrity of the software when it's loaded but does nothing to guard against bugs in the software when it's running.

Question 17

A *chroot jail* disallows a process:

(a) Access to the network.

[✓ (b) Access to parts of the file system.]

(c) The ability to execute specific system calls.

(d) The ability to write files.

The *chroot* system call changes the root of the file system for a process to a given directory.

Question 18

A secure message from Alice to Bob is encrypted with:

(a) Alice's private key.

(b) Alice's public key.

➡ (c) Bob's public key.

(d) Bob's private key.

(a) If Alice encrypts a message with her private key, anybody can decrypt it with her public key.

(b) If Alice encrypts a message with her public key, it can only be decrypted with her private key, which only she has.

(c) If Alice encrypts a message with Bob’s public key, it can be decrypted with Bob’s private key, which only Bob has.

(d) Only Bob has Bob’s private key. Alice cannot do this.

Question 19

A hybrid cryptosystem uses:

(a) Two levels of encryption to secure data: symmetric encryption followed by public key encryption.

(b) Two symmetric keys, A and B, and encrypts data three times: $E_A(D_B(E_A(M)))$.

➡ (c) Public key cryptography to send a key and symmetric cryptography to encrypt data.

(d) A combination of public and private keys to encrypt data securely.

Public key cryptography is used to send a randomly-generated session key. That session key is a symmetric key that will be used for that communication session. Public key cryptography does not require a shared secret but is considerably slower for both encrypting data and generating keys.

Question 20

The Challenge Handshake Authentication Protocol (CHAP) is secure NOT because:

(a) Both sides have a shared secret that is never sent across the network.

(b) The server issues a random challenge for each authentication session.

[✓ (c) Only hashed passwords are stored on the server.]

(d) All responses from the client are hashed values.

CHAP requires both parties to have a shared secret, S. The server generates a challenge string, C and sends it to the client. The client can compute a hash of the two values, H(S,C), and send it back. The server can compute the same value to verify the result. The shared secret (password) cannot be stored in hashed form on the server since the server needs to extract the value to compute H(S,C).

Question 21

An intruder cannot deduce the next S/key password having seen the previous one because:

(a) Passwords are hashed.

(b) The next password will be unrelated to the previous one.

[✓ (c) One-way functions are used to generate passwords.]

(d) An encrypted communication session will be needed to transmit the password.

S/key works by generating a set of passwords, each of which is generated by applying a one-way function to the previous password. The user presents them in reverse order. Hence, given password *P*, the next password will be $f^{-1}(P)$. An intruder cannot compute this since *f* is a one-way function.

Question 22

For Alice to talk to Bob, the Kerberos ticket that Alice obtains contains:

(a) Alice's secret key.

(b) Bob's secret key.

[✓ (c) The session key.]

(d) A Kerberos authorization message signed by the Kerberos server.

Kerberos generates a random session key that Alice and Bob will use to communicate with each other. It sends two messages to Alice: one contains the session key encrypted with Alice’s secret key. The other, called the *ticket* or *sealed envelope*, contains the session key encrypted with Bob’s secret key.

Question 23

Your digital certificate will not contain:

(a) The certificate issuer's identification.

(b) Your name.

[✓ (c) Your private key.]

(d) Your public key.

A digital certificate is a way of distributing public keys. It contains your identification (so someone knows whose key it is), the certification authority’s identification (so someone knows who issued is), and a signature (a hash of all the data encrypted with the certification authority’s private key). Your private key should never be distributed.

Question 24

TCP is a Transport layer protocol whereas UDP is a network layer protocol.

TRUE [✓ FALSE]

No. Both TCP and UDP are transport layer protocols. IP is a network layer protocol.

Question 25

The Address Resolution Protocol (ARP) converts Ethernet addresses into IP addresses.

TRUE [✓ FALSE]

No. ARP converts IP addresses to Ethernet addresses. There's another protocol called Reverse ARP (RARP) that converts Ethernet addresses to IP addresses but this is an obsolete protocol. DHCP, the dynamic host configuration protocol, does this in a more useful manner.

Question 26

Sockets are implemented as a pseudo file system type under VFS (Virtual File System).

TRUE [✓ FALSE]

No. Even though sockets and file descriptors sit in the process' file descriptor table, operations on file systems go to VFS but operations on sockets go to the network stack. A socket is not implemented like a file system. It just happens to support read, write, and close operations.

Question 27

NFS provides session semantics.

TRUE [✓ FALSE]

No. That would mean file modifications would not be visible on the server until the file was closed. NFS propagates changes while the file is still open.

Question 28

A digital signature is an encrypted hash.

[✓ TRUE] FALSE

Yes. A digital signature is a hash of a message (the thing you're signing). The hash is then encrypted with your private key. Anyone who has your public key can validate that the message has not been modified because its hash can match the hash in the signature. Nobody else can re-create the signature since they do not have your private key.

Question 29

Diffie-Hellman is a public key encryption algorithm.

TRUE [✓ FALSE]

No. Diffie-Hellman is neither a public-key nor symmetric algorithm. It is a way for two parties to come up with the same number by using private and public pairs of numbers that, unfortunately, are called keys.

The End