1. Si illustrino brevemente le differenze fra I/O ad interrogazione ciclica (Programmed I/O), I/O guidato da interrupt (Interrupt-driven I/O) e trasferimento diretto in memoria (DMA).

   Risposta: Mentre con la modalità Programmed I/O (I/O a interrogazione ciclica) il processore manda un comando di I/O e poi attende che l'operazione sia terminata, testando lo stato del dispositivo con un loop busy-wait (polling), con la modalità Interrupt-driven I/O, una volta inviato il comando di I/O, il processo viene sospeso fintanto che non arriva un interrupt a segnalare il completamento dell'operazione. Durante la sospensione del processo, la CPU può mandare in esecuzione altri processi o thread. Di fondamentale importanza è il vettore di interrupt che consente di selezionare la routine di gestione opportuna per ogni tipo di interrupt. Ovviamente la prima modalità è efficiente soltando nel caso in cui la velocità del dispositivo di I/O sia paragonabile a quella della CPU. La modalità DMA richiede un controller DMA e funziona in questo modo: la CPU imposta i registri del controller DMA specificando il tipo di azione di I/O, l'indirizzo di memoria ed il conteggio di byte da trasferire. Poi i dati vengono trasferiti senza più richiedere l'intervento della CPU; infatti il controller del dispositivo di I/O riceve le richieste di lettura o scrittura da parte del controller DMA a cui notifica il completamento dell'operazione una volta che ha trasferito il byte da/verso l'indirizzo di memoria corretto (specificato dal controller DMA). A questo punto il controller DMA incrementa l'indirizzo di memoria comunicandolo sul bus e decrementa il conteggio dei byte da traferire, ripetendo la richiesta di lettura o scrittura al controller del dispositivo fintanto che il conteggio dei byte non raggiungerà lo zero. Soltanto a questo punto verrà inviato un interrupt alla CPU che potrà far ripartire il processo sospeso. Siccome il controller DMA deve bloccare il bus per consentire i trasferimenti dal controller del dispositivo alla memoria, se anche la CPU ha bisogno di accedere al bus dovrà aspettare, venendo così rallentata.

2. (a) Cos'è un'interruzione precisa?
   (b) Cosa comporta per una macchina avere delle interruzioni imprecise?

   Risposta:

   (a) Un'interruzione si dice precisa quando gode delle seguenti quattro proprietà:
      i. il program counter viene salvato in un posto noto,
      ii. tutte le istruzioni che precedono quella puntata dal program counter sono state completamente eseguite,
      iii. nessuna delle istruzioni che seguono quella puntata dal program counter è stata eseguita,
      iv. lo stato di esecuzione dell'istruzione puntata dal program counter è noto.

   (b) Nel caso in cui si abbiano interruzioni imprecise è difficile riprendere l'esecuzione in modo esatto in hardware: la CPU si limita a "riversare" sullo stack tutta l'informazione relativa allo stato corrente, lasciando che sia il sistema operativo a capire che cosa debba essere

fatto. In questo modo rallentano le fasi di ricezione degli interrupt e di context-switch/ripristino dell'esecuzione, provocando grosse latenze.

3. (a) Si illustri come funziona l'allocazione concatenata per quanto riguarda l'allocazione dei blocchi nei dischi.
   (b) Soffre di frammentazione esterna?
   (c) Supporta l'accesso diretto (seek)?

   Risposta:

   (a) Lo schema di allocazione concatenata prevede che ogni blocco dati abbia una parte di esso riservata alla memorizzazione dell'indirizzo del blocco successivo (l'ultimo blocco viene marcato con un indirizzo non valido, e.g., -1). In questo modo l'implementazione di un file consiste in una lista di blocchi concatenati ed è sufficiente memorizzare per ogni file nella directory entry il blocco iniziale e quello finale.

   (b) Questo schema non soffre di frammentazione esterna, dato che i blocchi assegnati ad un file possono essere sparsi ovunque nel disco (non devono necessariamente essere contigui); infatti, per aggiungere un blocco ad un file è sufficiente aggiornare il puntatore dell'ultimo blocco in modo che punti ad uno dei blocchi liberi su disco (marcando l'indirizzo di quest'ultimo come indirizzo finale, e.g., -1 ed aggiornando la directory entry corrispondente).

   (c) L'accesso diretto (seek) non è supportato; infatti per accedere al blocco i-esimo bisogna prima scorrere la catena di puntatori dei precedenti i-1 blocchi.

4. Si spieghino le differenze fra architetture UMA e NUMA.

   Risposta: Un sistema multiprocessore UMA (Uniform Memory Access) è un sistema strettamente accoppiato che gode della proprietà che il tempo di accesso a qualunque locazione di memoria è sempre lo stesso (indipendentemente dal fatto che si faccia riferimento a memoria locale o di un altro nodo del sistema). Invece un sistema NUMA (Non Uniform Memory Access), pur rimanendo un sistema strettamente accoppiato, è caratterizzato dal fatto che l'accesso alla memoria locale è più veloce (da 2 a 15 volte) dell'accesso alla memoria remota. Ciò consente di ottenere sistemi scalabili con un gran numero di CPU.

5. Qual è la differenza fra servizi di rete e servizi distribuiti?

   Risposta: I servizi di rete offrono ai processi le funzionalità necessarie per stabilire e gestire le comunicazioni tra i nodi di un sistema distribuito (es.: l'interfaccia fornita dalle socket). In sostanza gli utenti devono essere consapevoli della struttura del sistema e devono indirizzare esplicitamente le singole macchine. I servizi distribuiti invece sono modelli comuni (paradigmi di comunicazione) trasparenti che offrono ai processi una visione uniforme, unitaria del sistema distribuito stesso (es: file system remoto). I servizi distribuiti vanno quindi a formare il cosiddetto *middleware*, ovvero, uno strato software fra il sistema operativo e le applicazioni che uniforma la visione dell'intero sistema.

6. Si spieghi brevemente cosa si intende per attacco basato su *buffer overflow*.

   Risposta: Siccome la maggior parte dei sistemi operativi e dei programmi di sistema sono scritti in C (per ragioni di efficienza) e quest'ultimo non mette a disposizione dei meccanismi per impedire che si facciano dei riferimenti oltre i limiti dei vettori, è molto facile (ad esempio, passando in input al programma una stringa di caratteri sufficientemente lunga) andare a sovrascrivere in modo improprio delle zone di memoria con effetti potenzialmente disastrosi. La tecnica del buffer overflow sfrutta proprio questa vulnerabilità; infatti, se il vettore è locale ad una funzione C, allora la memoria ad esso riservata verrà allocata sullo stack e sarà possibile, con un indice che ecceda il limite massimo consentito per il vettore, andare a modificare direttamente lactivation record della funzione. In particolare, modificando lindirizzo di ritorno, è possibile di fatto *redirezionare* l'esecuzione del programma, provocando lesecuzione di codice potenzialmente pericoloso. La tecnica del buffer overflow viene in particolare usata per eseguire uno ShellCode, ovvero, un programma in linguaggio assembly che esegue una shell (e.g., /bin/sh in Unix o command.com in Windows) da cui l'attaccante può iniziare comodamente altre azioni pericolose nei confronti del sistema. In questo modo quando la funzione prova a restituire il controllo al chiamante, leggendo sullo stack lindirizzo di ritorno, in realtà provoca lesecuzione dello ShellCode.

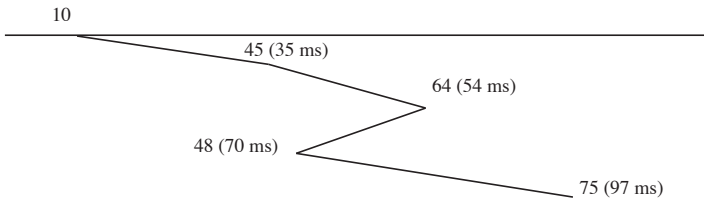
   Oltre ad *iniettare* lindirizzo di inizio dello ShellCode sullo stack, bisogna riuscire anche ad inserire il codice stesso dello ShellCode nello spazio di memoria indirizzabile dal processo vulnerabile. A tale scopo, una tecnica relativamente semplice consiste nel definire una variabile dambiente che lo contenga come valore e richiamare il programma vulnerabile passandogli tale ambiente.

7. Si consideri un disco gestito con politica C-LOOK. Inizialmente la testina è posizionata sul cilindro 10, ascendente; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 64, 45, 48, 75, rispettivamente agli istanti 0 ms, 30 ms 40 ms, 70 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite nell'ordine seguente: 45, 64, 48, 75.



2. Il tempo di attesa medio per le quattro richieste in oggetto è dato da $\frac{(35-30)+(54-0)+(70-40)+(97-70)}{4} = \frac{5+54+30+27}{4} = \frac{116}{4} = 29ms$.

Il punteggio attribuito ai quesiti è il seguente: 4, 3, 3, 3, 2, 2, 3, 3, 4, 3, 2 (totale: 32).