第 11 回 IT ソフトウェア翻訳士認定試験

＜1 次試験＞　11 月 21（日）10：00～15：00

問題 1・2 の両方について解答のこと。選択ではありません。

＜問題 1＞　全文を訳して提出してください。

Buffer overflows: Reasons to apply mundane-sounding software fixes

Many times when updates or security patches appear for the OS or applications, a common reason for the update is that an attacker can make the program execute arbitrary code, usually by tricking a user into opening a "maliciously crafted" file, be it for a document, a movie, or an elaborate Web page containing scripts and images. This wording has come to sound rather mundane and uninformative to many users, who usually just install the available patches without having an idea of what is going on.

The programming language used for writing most OS X programs is Objective-C, but others such as C and C++ may also be used and as with any programming language there are quirks in these which allow for various inadvertent faults to appear if the programmer is not careful.

One of these faults is called a "Buffer Overflow," which allows a program to write outside of a designated allocation of memory and result in odd behavior, including crashes and in some circumstances the execution of a hacker's code.

Buffer overflows are very easy to do in some programming languages. For instance, take a look at the following code in C (it's very simple to understand):

```
#include <stdio.h>

int main (int argc, const char * argv[]) {
        char word[10];
        scanf("%s",word);
        printf("The word is: %s", word);
        return 0;
}
```

This is a very simple "C" program that will run in the Terminal and perform four tasks. It will create a character storage variable named "word", ask you to enter a value for this variable, print out the value of this variable to the Terminal window, and then return a "0" value to the Terminal. It's very crudely similar to the concept of the clipboard, in that there is a place to store the data, a way to enter it, and an output for the data.

```
char word[10];
```

This line declares a character variable that we've named "word," which is 10 bytes long so it can hold 10 characters (one character is one byte). Each of the 10 bytes is an individual RAM unit (or "memory address") that will be used to hold the values stored in the "word" variable. Allocating it in this way allows the memory addresses for this variable to be easily tracked, and as such forms a 10-byte "buffer" for the storage variable.

＜問題 2＞　下線部分のみを訳して提出してください。

How to Set Up a Virtualization Server

By using a single physical server to run many virtual servers, you can decrease operational costs and get far more bang for your buck. The best part? You can do it cheaply and easily.

It's impossible to buy a server today that isn't multicore, but many small-business requirements don't call for that much horsepower. The end result is a relatively expensive physical server that does little but consumes power and generates heat. That's why using a multicore server — one with 4, 6, or 12 processing cores on a single CPU — to host several virtual servers makes sense, no matter what size your computer is.

Choosing a Host

The key to successfully virtualizing servers in a smaller environment starts with the physical host server. Though it will be responsible for hosting possibly dozes of virtual servers, it will require far fewer CPU resources than you may think.

Depending on the virtualization software (aka the hypervisor) that you use, you will likely be able to run a surprising number of virtual servers on a four- or six-core CPU. The reason for this is that generally most servers run near idle a large portion of the time. When they are tasked with work, their resources tend to be spread out among the CPU, RAM, disk, and network input/output, with only a subset of the virtual servers requiring significant CPU resources. By taking advantage of this law of averages, you can consolidate multiple physical servers onto a single host server.

That isn't a hard-and-fast rule, however. Some servers, such as database servers, run heavier loads more consistently, and may not be suitable candidates for virtualization in a smaller infrastructure. It all depends on the hardware resources available to the host server, on the virtualization software features, and on the requirements of the virtual server. Fortunately, setting up and testing these

requirements beforehand isn't difficult.

In choosing the hardware, a good rule of thumb is that having more cores in the host server trumps having higher clock speeds; if your choice is between, say, a 4-core CPU running at 2.93GHz and a 6- or 12-core CPU running at 2.4GHz, you'll be better off with the latter. The capability to spread the virtual-server load across more CPU cores typically translates into faster, more consistent performance across all the virtual machines.

Virtualization hosts can always use more RAM, so be sure to get as much as you can, and to select the fastest type available. It is relatively straightforward to oversubscribe CPU resources – to allocate more virtual CPUs to the virtual servers than physically exist within the host server – but it's far more difficult to oversubscribe RAM. The more RAM you have, the more virtual machines you can run.