**MIS 6330: IT Security**

**Prof. A. Lahiri**

**Individual Homework 5**

1. When we do *testing for functionality*, we are only focused about getting the correct output with given input using the functionality.

For example,

If the program calculates square of input number, then whether the output   
is exactly the square of the input number is tested when *testing for functionality*.

When we do *testing for vulnerability*, we check input and output to be correct   
in terms of size, type and form that the functionality is expected to.  
For example,

Suppose, the type of input for program is numeric input only   
(floating point or integer) and form of input as decimal notation inputs only.

*Testing for vulnerability* would check if someone is entering any other type   
of input and would check if there is appropriate handling present for   
this type of bad input.

While, numeric input (correct type of input) could be entered in   
multiple ways like Decimal notation (123.456) or Scientific notation (2\*101)   
or even Exponential notation (6.02e23), but current program accepts   
decimal inputs.

These types of extensive checks are done while *testing for vulnerability*   
which are not covered while *testing for functionality*.

1. Consider the C procedure below.

void hello(char \*tag)

{

char inp[16];

printf(“Enter value for your %s: ”, tag);

gets(inp);

printf(“Hello your %s is %s\n”, tag, inp);

}

**hello(“message”) produces an output of this form:**

Enter a value for your message: *test* (🡸 input from user)

*Hello your message is test* (🡸 output from hello)

* 1. The main vulnerability of above piece of code is that using “**gets()**” function, the input from user is being taken in. However, the function will not   
     check the size of buffer left to store the input and continue taking in input till new line or end of file is detected.

Due to this, it could cause a buffer overflow if input is large enough.

* 1. The above code could be modified to use “**fgets()**” instead of “**gets()**” as follows:

fgets(inp,16,stdin);

1. The vulnerability present in Symantec Anti-Virus Engine (AVE) would have   
   allowed attackers access to the memory locations of the   
   kernel (core part) of the operating system.

Using this access, the attacker could easily do “**Denial of Service (DOS)**” attack on any machine with this vulnerability by simply sending an email   
with infected file as an attachment.

Lessons that can be learnt from this incident are as follows:

* To avoid such attacks, the software should be rigorously tested for   
  overflow (in terms of storing capacity) for every component where an input is taken.
* Employees should be made aware to open emails from only   
  trusted source to avoid any such attacks via attachments.
* Employees should also be made aware about the procedure of reporting   
  any such malfunction in their work station directly and as early as possible   
  to IT Department as well the company that created the software.

This way, the company that created could release the patch for any such bug before it spreads into more systems.

* IT Department should audit work machines to make sure all latest patches are installed in a timely manner to avoid any such attacks.