NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.

SCENARIO:

Consider a program that evaluates user-entered mathematical expressions. The code uses the `eval(expression)` function, which parses the `expression` argument and evaluates it as a Python expression. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

```
import sys, math

print "*** Welcome to the Python Calculator ***"

while True:
   expr = raw_input("Enter a mathematical expression or 'quit': ")
   if str(expr) == "quit":
       sys.exit("You've ended the Python Calculator")
   else:
       print("The answer is " + str(eval(expr)))
```

Questions:

1. What will the program do when executed?

2. If user enters “`math.trunc(345.67)`”, how will the program behave?

   a. It will crash because the string passed is not a standard arithmetic expression.
   b. It will print “`The answer is 345`”.
   c. It will print “`The answer is math.trunc(345.67)`”
   d. none of the above

[Other statistical questions will be imported here while creating survey.]

NOTE: ANSWER IS TO BE SHOWN TO THE PUZZLE TAKER AT THE END OF THE SESSION.

ANSWER:

1. The program will read a line from the standard input and evaluate the input as a Python expression.

2. b

The `eval(expr)` function accepts a string and evaluates it as a Python expression. Thus any valid Python expression can be passed to the program and it will be evaluated. The string value “`math.trunc(345.67)`” is a valid Python expression that results in the integer value `345`.

TAGS:
python, code-evaluation, code-injection

CATEGORIES:
Blindspot - YES
Type - Injection
Number of distinct functions - 4
Number of total functions - 5
Blindspot function - `eval()`
Function call omitted - NO
Blindspot type - Validation missing
Number of parameters in the blindspot function - 1 parameter
Cyclomatic complexity - 5

NAME:
eval(expression)

DESCRIPTION:
In the `eval(expression)` function the `expression` argument is parsed and evaluated as a Python expression.

BLINDSPOT:
The vulnerability lies in the `eval(expression)` function. The `eval` function accepts a string value as an argument, and evaluates it as Python expression no matter what the string value is (as long as it is a valid Python expression). As a result, a user can craft inputs to execute malicious code.

CORRECT USE EXAMPLE:
#N/A

MORE INFORMATION:
#N/A

REFERENCES:

1. http://nedbatchelder.com/blog/201206/eval_really_is_dangerous.html