Project 1: Control Hijacking

Julian Cooper JELC@STANFORD.EDU
AA228/CS238, Stanford University

1. Buffer Overflow

Vulnerability: Want to exploit `strcpy` being called without bounds checking.

Exploit: We can overwrite the return address of `foo` with the address of our shellcode, then execute the shellcode to get a root shell.

Working notes:

- Using gdb we can find address of the buffer variable, 0x7fffffffdc20.
- We then can find the address of the return address, 0x7fffffffdd28.
- The difference of these two addresses is 0x108 (264), which is the number of bytes we need to overwrite to get to the return address. Note, we need to add 8 bytes to this to account for the saved return address, so our exploit must be 272 bytes long.
- Copy shellcode without terminating null pointer into our buffer exploit.

2. Graphs

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3. Code

```
import sys

import networkx


def write_gph(dag, idx2names, filename):
    with open(filename, 'w') as f:
        for edge in dag.edges():
            f.write("{}, {}\n".format(idx2names[edge[0]], idx2names[edge[1]]))
```

```
def compute(infile, outfile):
    # WRITE YOUR CODE HERE
    # FEEL FREE TO CHANGE ANYTHING ANYWHERE IN THE CODE
    # THIS INCLUDES CHANGING THE FUNCTION NAMES, MAKING THE CODE MODULAR,
    BASICALLY ANYTHING
    pass


def main():
    if len(sys.argv) != 3:
        raise Exception("usage: python project1.py <infile>.csv <outfile>.gph
    ")

    inputfilename = sys.argv[1]
    outputfilename = sys.argv[2]
    compute(inputfilename, outputfilename)


if __name__ == '__main__':
    main()
```