Justin Valcarcel
Application Security Assignment 2

easytocode.py - I had some trouble understanding syntaxially the namespacing segment of the code (lines 18 and 19) but eventually figured out it essentially doesn't let you access builtins by name and only gives you access to True/False/None in the builtins namespace. Normally, this would not be a problem because you could access the modules in a more roundabout way but I am not sure how to with the period (.) blacklisted. I was also looking at https://isisblogs.poly.edu/2012/10/26/escaping-python-sandboxes/ for more tactics but having the period removed that prevents me from encoding/decoding blacklisted strings. I also thought about emptying FORBIDDEN_STRINGS, setting forbidden_string to False and assigning something different to filedata but was unsuccessful in getting past that blacklist filter.

One thing I noticed is that there aren't any memory limitations set by this sandbox. I was able to hold up the interpreter by using "a = ' ' * 99999999999. I did notice a tiny bit of slowdown with other applications on my computer but it was not significant and may not have been have just been a coincidence.

Upon further inspection of going after the memory limitation I actually set "a = ' ' * 999999999999999" and was able to get a memory error, I have posted it in the comments of the 'simple.input' file.

potentiallyhackablesandbox.py - The first thing I noticed was the short blacklist and that I can use the period and underscore (_) symbol again. However, the first thing I actually tried was do what crashed the program last time with "a = ' ' * 999999999999999" and it did.

I tried to access the modules in a more roundabout way and can get past the "BAD_STRINGS" list by concatenating the strings but with the __builtins__ namespace disabled I was still unable to invoke anything useful. I've spent a ton of time on this homework and while I've learned a lot about Python, I'm still having trouble navigating the modules via dir() and accessing them directly and not with __builtins__.

asandbox.py - This is the example we went over in class. Honestly, I have very little idea how to attack this outside of inputting a huge calculation like a = 999999999 ** 99999999. I input something similar to that and it's been a good 5 minutes now and it hasn't finished.

https://github.com/ceinfo/ApplicationSecurity - This was in Java which I'm not too familiar with, only in the context of programming Android apps. I looked at it though and really had a lot of trouble understanding what does what. It does look like there are limitations on allocating memory. I was not really too sure on what was going on as far as validation, it looks like you can use a variety of commands like SUB/ADD and anything else is an exception.

https://github.com/mramdass/Turing_Complete_Sandbox - This requires at least Python 3.3 and Linux to work. I do not have Linux with Python 3 on it so I was unable to run it.

https://github.com/fjm266/appSec1 - Similar problems to what I've been running into and that is referencing modules without the builtin namespace.

https://github.com/crimsonBeard/App-Sec - This one kept crashing whenever I put in any input at all.

https://github.com/kellender/Secure_Turing_Complete_Sandbox- - This was in C and I honestly didn't have much clue on what was going on.