AP CSP

Final Project Write-up

Steven Madigan

The Annoying Script/AppleScript

My project idea a surprisingly simple. I came into it not even looking at a “problem”, and to be honest, it doesn’t solve or even identify any problems. The real purpose of the script app was to learn the basics of passing and writing AppleScript, which an OS X-proprietary scripting language that can be used to perform system tasks.

There are some difference between straight AppleScript and Automator script, though. I do not know, nor am I familiar with, Automator or any of its uses. Upon a search of GitHub while I was researching this project, I landed on several Automator processes which would also reach the same end effect.

I will now explain the process as it is required by the AP Guidelines for submission according to the sections also provided.

Section II

2a) The innovation in the project that I selected is definitely hard. It doesn’t use anything like a compact SoC or an IoT device like the Raspberry Pi. It is a very useful springboard, though, because learning automation code on a desktop system is a great starting place for all sorts of knowledge. By learning what tasks must be written in order for an end goal to happen, you also learn a great deal about the way the computer and the OS works underneath.

My script is a very mundane one, but one which proves that even the simplest knowledge of a scripting language can have huge consequences. In 9 lines of code, you can bridge the gap between your local machine and any device on the iMessage network. The ability to slam someone’s device with texts is wholly childish, but it’s a potential gateway to finding exploits in the way that the OS treats apps in the sandbox.

2b) The compilation of the project itself was very simple. With no true knowledge of AppleScript or the transfer of information under the hood of Apple’s OS X, I turned to the wide world of the internet to help. With some reference from Stack Overflow, I learned about the feasibility. I was able to deduce that there were open doors between a script and one of Apple’s own stock apps. My big question was whether or not such a script could actually insert code into the sandbox on an iOS device enough to cause damage. I also found various bits of the code from GitHub, and my final project is effectually an open-sourced compiled version of several other projects. I was able to learn to pare down my code so that heavy-weight lines were removed, and only the bare minimum was left.

2c) I think that the main goal of my project was an amplification of risks involved with the concept. As far as I was able to develop the project given the skills that I have, I don’t see there being any truly insidious or negative impacts of the code. It is very annoying when deployed, but there is no remaining damage. I wasn’t able to implement any type of advanced payload. The only thing that I was able to do was send abbreviated links which led to infected sites. The positive side is obvious, though, and that would be learning how to write Automation code for Apple’s OS X.

2d) The downside of this project is that the actual flow is data is completely backend. The execution of script generates a