Connor McCoy
CIS4360
11/05/2014

Project Proposal

What?

I'm creating a game (working title "BotNet") that simulates commanding a botnet to hack into a server. The objective is to hack into a server by inputting the correct username/password, which is most naively done via brute force, expedited by gathering as many slaves as possible. Each player in a session competes against every other player to be the first to do this.

To simulate this experience, each player will have up to 10 open connections (called "slots"). They are guaranteed to retain at least 1 slot, but each of the other 9 can be captured by other players. To start, I plan to cap the number of slots a hostile player can capture from another individual player at 1; however, a player can capture up to this cap from every other player in the session.

Furthermore, these slots will be simulated by each player's device sending a message to the server each cycle (of perhaps 1 second). The message contains their own device's contribution towards "hacking" the server, their commands to their slaves, their offensive actions toward other players, their defensive actions, and finally the contribution of their slaved slots towards other player's benefit. The server will then process this data and send updates to each player accordingly.

If possible, the devices will send appropriate messages to each other directly to better simulate a botnet; however, this is a stretch goal as the user wouldn't be able to tell the difference.

Offensive measures a player can take include "hacking" other players to slave them (they send a Trojan worm). Additionally, players can attempt to access a simulated directory on a slave which informs them which passwords that player has already tried (and thus accelerate one's own efforts through elimination of possibilities).

Defensive measures include HIS security (simulated as a memorization/pattern analysis minigame using characters; core elements such as identifying self, choosing window sizes, and a malware "hit rate"/symbols will be retained).

Utility measures include performing traceroutes, to attempt to identify who's currently attacking you with their slaves or who's already stolen that juicy slot you want.

Additional stretch goals include incorporating more intelligent dictionary and rainbow table attacks into the game.

How?

At first I will attempt to develop the game using Xamarin, which is a platform that allows me to deploy on iOS, Android, Windows and Mac with a single, shared C# codebase. Failing that, I'll probably create the game in Java (the UI would be a series of swing/FX panels, and it would be distributed in a JAR file). Otherwise, I'll implement the game as a native Android app.

In all cases, I'll probably create the server in Java and run it on my desktop.

The slots mentioned previously will become sockets (this is basically the same in Java and C#).

I have, if possible, less than no artistic talent so I will deliberately go for an old, blocky, "green" art style (think the computer terminal UIs from Aliens). Hopefully, that should suit the project and be easy to implement.

So far, I've identified 4 minigames:

1. "Slaving"—the player sends a Trojan worm to an opponent, perhaps disguised as a server response.
   Every 30 seconds the server should send feedback to players concerning the progress of their hack. Trojan worms will be buried in this feedback message. The player won't have time to review it all manually (it would take ~1 minute); they can either accept the message, quarantine parts of it, or reject it. Rejecting it forfeits any progress made that cycle in their hack of the server. (This is a wonky gameplay mechanic, working on a way to make it more accurate).
2. "Rooting"—the player escalates their access level high enough to view and download the set of passwords their opponent has already attempted.
   Essentially, a themed text-based adventure game.
3. "HIS"—the player attempts to identify friendly (self) symbols and flag hostile symbols.
   A character memorization/analysis minigame.
4. "Traceroutes"—the player attempts to discover who is controlling a group of slaves.
   Still considering how to implement this at present; big decision is to do it visually (graph) or entirely text based.

Why?

I'd like the opportunity to compile what I've learned in this class in game which, crucially, shows how offensive and defensive security measures interact at a high level and intuitively depicts the computer arms race.

Resources

http://xamarin.com/
(Xamarin main website)

VisualStudio
(with Xamarin)

NetBeans
(for at least the Java server)

AndroidStudio, GennyMotion
(should I develop for Android natively)