**Building SecurityGame** (pick your favorite):

* Add source files located in src directory to your favorite IDE. Add the CSV jar to your project library. Compile and run GameMaster.
* Compile code in src directory via command line and add CSV jar to your classpath. Run GameMaster.
* From the root directory execute “ant” in the command line. Execute jar file located in new classes directory.

**Making a Defender agent:**

1. Extend the Defender class to create your own defender agent class.
2. Override the makeMoves()method of the Defender class by your own defender logic.
3. You have three methods i.e. strengthen() , firewall(), and honeypot() to apply your defender actions on the nodes.

Note that the system will not allow you to do the following:

* Strengthen a node past 20.
* Strengthen a public node (security and point value of zero).
* Make a honeypot with that looks like a public node.
* Make a honeypot with more than 20 in security value or point value.
* Make a honeypot with no edges.
* Isolate a node by removing all edges to it or in some manner disjoint the graph.

**Making an Attacker agent**:  
  
While making an attacker agent class all you need to know about these three classes :

1. **Attacker class**
2. **Your own attacker agent class**:  
   IMPORTANT NOTE: Your attacker object will be recreated for every action. Because of this, model your Attacker to only make a decision on the current information. Do not try to use variables that will carry on in to the next makeSingleAction(). Make use of the three protected variables inherited from Attacker. These variables include:
   1. protected ArrayList<Node> capturedNodes - a list of the already captured nodes
   2. protected ArrayList<Node> availableNodes - a list of the available nodes for attacking and probing.
   3. protected int budget - the current budget of the Attacker. Be careful that your next move will not cost more than your budget.
3. **Node class**:   
   This class is used for creating nodes for the network. Important variables information for Attacker agents:
   1. nodeId - integer representing the node
   2. sv - security value of a node (-1 means unknown)
   3. pv - point value of a node (-1 means unknown)
   4. isHoneyPot - boolean for if the node is a honeypot (-1 means unknown, 0 means false, 1 means true)
   5. captured - if the node is a public entry node or has been successfully captured via attack
   6. bestRoll - the highest roll on this node (if -1, the node has never been attacked), if the security value is -1, bestRoll may be used to reason what the sv might be (actual security value >= bestRoll if sv == -1)
   7. neighborAmount - number of connections this node has. Will be different than neighbor.size() if the node has not been captured. (-1 means unknown)
   8. neighbor - list of neighbors to this node. Will be empty if the node has not been captured

1. Extend the Attacker class to make your own attacker agent.
2. Override the makeSingleAction()method of the Attacker class by your own attacker logic.
3. You have 6 different AttackerActionType for the attacker i.e.
   1. ATTACK
   2. SUPERATTACK
   3. PROBE\_SECURITY
   4. PROBE\_POINTS
   5. PROBE\_CONNECTIONS
   6. PROBE\_HONEYPOT
4. You need to decide which action you should take on which node.

**Running agents from the Game master:**

1. Specify the number of games your want.
2. Add your Defender agent to the section in main that says to “add defender here”.
3. Add your Attacker agent to the section in main that says to “add attacker here”.
4. Add your Defender and Attacker to the methods getDefenderByName() and getAttackerByName().