Data Structure

Name: clientTable
Type: Array of pointers to type ClientResourceRec
Location of definition: dix/resource.c

resource.c:
static ClientResourceRec clientTable[MAXCLIENTS];

Note* statically defined arrays have local scope, so at file scope this array can only be read/written to by code in resource.c. Values will remain in memory until program termination

# of references: 57
# Reads: 56
# Writes: 1

List of writes:
dix/resource.c: clientTable[i].resources[j] = NULL;

Exploitability?
Statically defining the array actually limits the scope of this data quite well. Untrusted code would need to be located in the same C file “resource.c” to have scope. Selections (from the other write up) have a clientPtr and are globally accessible, so if we wanted to iterate through clients, we could in theory iterate over all selection objects to learn about clients instead, making this option possibly moot. Not sure if all clients necessarily have a selection associated with them.