**Holodeck Next Generation**

Fuzzer Use Cases:

(realized from the Fuzzer One PRD)

* Application
  + User sets up a client/server test with fuzzing on both ends
  + User moves to a machine with an agent on it and gains master control
  + User observes fuzzing activity on on all agent machines from the master
  + User starts new agents on the network; agents pick up licensing from the master
* Agents
  + User starts agent on a machine and uses the console machine to set up tests on the agent
* Fuzzing types
  + User randomly fuzzes contents of a file
  + User fuzzes the network stream by replacing every third and sixth byte
  + User uses a template to identify the HTML file header and fuzz data only in it
  + User uses a template to identify a custom protocol header and fuzzes only the data in the target address field
  + User resets the random fuzz
  + User resets the parametric fuzz of replacing every third and sixth byte
* File Corruption
  + User fuzzes the data on a configuration file
  + User uses an attack pattern to identify and fuzz scripts in a HTML page
  + User sets up the fuzzer to fuzz every third and sixth byte read in, from any file
* Network Corruption
  + User fuzzes the data coming in from a server connection
  + User uses the attack pattern library to identify and fuzz SQL strings
* Registry Corruption
  + User fuzzes the registry reads on the entire registry randomly
  + User uses an attack pattern to fuzz registry reads from the Windows Configuration keys
  + User fuzzes only the HKLM\Software branch of the registry
* RPC Tests
  + User uses templates to identify the "User Id" field of SQL queries and fuzzes all incoming and outgoing queries
  + User fuzzes all outgoing DCOM data by fuzzing 10 bytes out of every 200 bytes
  + User randomly fuzzes all incoming WebService Data
  + User uses an attack pattern to replace incoming DCOM data with code to perform a buffer overrun (will be an effective test only for DEP enabled apps)
* Reporting
  + User views total failed tests on the network
  + User views detailed test run results per agent
  + User generates HTML report
  + User generates report readable by XML aware readers
* Attack Pattern Library
  + User selects a network attack pattern and runs the AUT
  + User adds an attack pattern
  + User removes an attack patter
  + User updates an attack pattern
* Diagnostics
  + User generates the summary after a set of test runs
  + User replays a crash discovered via a File Corruption test
  + User replays a SQL string corruption test
* Mini-dump/Debugger
  + User runs a test under and IDE with the break into debugger option available; when the AUT crashes, the IDE's debugger breaks in at the crash location
  + User runs the test on a standalone installation; on crash, a mini-dump is generated
* Project Package
  + User creates a non-fuzzer enabled diagnostic package
  + User creates a fuzzer enabled diagnostic package
* Developer Kit
  + User generates a sample project
  + User generates a project with a sample network fuzzing implementation
  + User uses a generated project to fuzz DCOM data and Registry keys
  + User uses a generated project to fuzz data on an AUT only on trigger
  + User uses a generated project to run an advanced template and parametric fuzzing based test on multiple AUT components, driven from the overall test automation for the AUT
* User Interface and Integrations
  + User uses the Fuzzer menu to set up network fuzzing on the app
  + User uses the debugger to attach to the application being current fuzzed
  + User drops an executable into the fuzzer dock in the IDE to add a new AUT
* TeamMentor Integration