Overall Test Plan

Our project will be developed piece by piece, so we will be testing each addition to our Honeypot on its own if applicable, and then test it in context of the entire program. For example, our logger function will log all commands run on our server and save them to a log file. This can be tested on its own by running some simple commands in a terminal, and then tested in context by deploying it to our server. We will also want to carry out tests for each module of our Honeypot locally and remotely. Ultimately, everything needs to run remotely, but without knowing the program works locally we have little to go off of in terms of troubleshooting.

Test Case Descriptions

Server Connectivity Test 1 (SCT1)

- This test will determine if we can connect to the server when on the same local network
- This test will be conducted by using an SSH client to try and login to the server on the same local network
- The inputs for this test will be the IP address of the server, and the credentials to log in to the server
- The output should be a prompt with information on the server and the date/time of the last log in
- Normal
- Blackbox
- Functional
- Unit

Server Connectivity Test 2 (SCT2)

- This test will determine if we can connect to the server remotely from a foreign network
- This test will be conducted by using an SSH client to try and login to the server from a foreign network
- The inputs for this test will be the IP address (or domain name) and forwarded port of the server, and the credentials to log in to the server
- The output should be a prompt with information on the server and the date/time of the last log in
- Normal
- Blackbox
- Functional
- Unit

Server Connectivity Test 3 (SCT3)

- This test will determine if users other than ourselves can find the server and attempt to log into it
- This test will be conducted by opening a port (likely port 22) on the network for the server and observing if there are bad login attempts in /var/log/auth.log
- The inputs for this test will be the the command ‘cat /var/log/auth.log’
- The output should be a log file showing all login attempts to the server
- Normal
- Blackbox
- Functional
- Unit

Alert Test 1 (AT1)

- This test will determine if we can alert the admin to show that activity is taking place on the honeypot
- This test will be conducted by having someone remotely login to the server and waiting for an alert
- The inputs for this test will be login credentials
- The output should be a notification alerting the admin(s) that there is activity occurring on the server
- Normal
- Whitebox
- Functional
- Unit

Alert Test 2 (AT2)

- This test will determine if we can alert the admin to show that an attacker has gained access to the honeypot
- This test will be conducted by cross referencing the auth.log and the alert system to see if someone other than the project members has logged on to the server
- There will be no inputs for this test
- The output should be a notification alerting the admin(s) that there is activity happening on the server as well as a successful login notice in the auth.log
- Normal
- Whitebox
- Functional
- Unit

Logger Test 1 (LT1)
- This test will determine if we can record all activity on the server to a log file
- This test will be conducted by deploying the logger program to the server, running various commands on the server, and checking to see if the newly created log file matches what was run in the session
- The inputs for this test will be various linux commands
- The output should be a log file full of commands and the results of running those commands
- Normal
- Whitebox
- Functional
- Unit

Logger Test 2 (LT2)
- This test will determine if we can trigger the logger to record all activity on the server to a log file
- This test will be conducted by deploying the logger program to the server, running various commands on the server, and checking to see if the newly created log file matches
- The inputs for this test will be various linux commands
- The output should be a log file full of commands and the results of running those commands
- Normal
- Whitebox
- Functional
- Integration

Packet Dump Test 1 (PDT1)
- This test will determine if we can retrieve packets from server sessions
- This test will be conducted by running the tcpdump command on the server
- The inputs for this test will the ‘sudo tcpdump -c 10’ in the terminal
- The output should be a display showing roughly 10 packets captured and related information
- Normal
- Blackbox
- Functional
- Unit

Packet Dump Test 2 (PDT2)

- This test will see if we can write the results of the packet dump into a log file
- This test will be conducted by having someone remotely login to the server, running the packet dump to funnel to a log, and then checking the capture log
- The inputs will be the packet dump command along with some extra parameters
- The output should be a log file containing packet info from the current session
- Normal
- Whitebox
- Functional
- Unit

Packet Dump Test 3 (PDT3)

- This test will see if we can make the alert system trigger the packet dump
- This test will be conducted by having someone remotely login to the server and waiting for an alert, and then checking the capture log
- The inputs for this test will be login credentials and some linux commands
- The output should be a notification alerting the admin(s) that there is activity happening on the server and a new capture log
- Normal
- Whitebox
- Functional
- Integration

Packet Dump Test 4 (PDT4)

- This test will see if we can stop the packet dump after a server session has ended
- This test will be conducted by having someone remotely login to the server and waiting for an alert, and then checking the capture log after they have logged out
- The inputs for this test will be login credentials and some linux commands
- The output should be a notification alerting the admin(s) that there is activity happening on the server
- Normal
- Whitebox
- Functional
- Integration

Spoof Command Test 1 (SpCT1)

- This test will determine if commonly run linux commands can be masqueraded
- This test will be conducted by using an SSH client to run linux commands, and recording the activity on the server
- The inputs for this test will be common linux commands
- The output will be typical results for common linux commands, as well as a log file that recorded any sensitive information
- Normal
- Whitebox
- Functional
- Unit

Malicious Access Test 1 (MAT1)

- This test will see if we can gain access to the attacker's system with information retrieved from the server session
- This test will be conducted by recording information from the server session, finding sensitive information recorded, and using SSH to login to the attacker's system
- The inputs for this test will be the information/credentials gathered from the attacker's server session
- The output should be a prompt with information on the attacker's system
- Normal
- Whitebox
- Functional
- Unit

Test Case Matrix

Test ID	Normal/ Abnormal	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
SCT1	Normal	Blackbox	Functional	Unit
SCT2	Normal	Blackbox	Functional	Unit
SCT3	Normal	Blackbox	Functional	Unit
AT1	Normal	Whitebox	Functional	Unit
AT2	Normal	Whitebox	Functional	Unit
LT1	Normal	Whitebox	Functional	Unit
LT2	Normal	Whitebox	Functional	Integration
PDT1	Normal	Blackbox	Functional	Unit
PDT2	Normal	Whitebox	Functional	Unit
PDT3	Normal	Whitebox	Functional	Integration
PDT4	Normal	Whitebox	Functional	Integration
SpCT1	Normal	Whitebox	Functional	Unit
MAT1	Normal	Whitebox	Functional	Unit