|  |  |
| --- | --- |
| **Group 8** | **Death Star Image Exfiltration** |
| **Major:** | **Team members:** |
| EE | Nicholas Michael |
| CEG | Mason McDaniel |
| IT/Cyber | Chase Ennis |
| IT/Cyber  CS | Cade Wrinkle  Michael Mowad |

**Test and Evaluation Master Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement | Test Method | Evaluation Method | Threshold | Objective |
| 10 | Inspection | Verify the presence of a Raspberry Pi within the solution. | The Raspberry Pi is present. | The Raspberry Pi is present. |
| 20 | Inspection | Verify the presence of the Linux Server within the delivered solution. | The Linux server is present. | The Linux server is present. |
| 30 | Inspection | Verify the presence of the communication device attached to the Raspberry Pi within the delivered solution. | The communication device is present | The communication device is present. |
| 40 | Inspection | Verify the presence of a communication device attached to the Rebel Server within the delivered solution. | The communication device is present. | The communication device is present. |
| 50.1 | Inspection | Verify that the device can detect the USB connections from mass storage devices. | The USB connection is recognized. | Verify that software can recognize and load a 1024x1024 png from the USB device. |
| 50.2 | Test | Test that the image recognition software can recognize and load a 1024x1024 png from the USB device. | At least one image is recognized. | All images are recognized by the software. |
| 60 | Functional Test | Test that the software can differentiate between an image of the death star and non-death star images. | Software is able to classify a single image as a photo of the death star or not. | Given 100 images, 10 of which being death star images, the software can accurately differentiate and identify the 10 death star images. |
| 70.1 | Functional Test | Unidirectional communication between a transmitter and a receiver. | Receiver can recognize a connection from a transmitting device over air. | Receiver can parse data packets from a transmitting device over air. |
| 70.2 | Functional Test | The nRF24L01 is connected to an Arduino, which is then connected to a computer via serial port. A script opens a serial link with the device. The computer sends a block of data to the Arduino dongle as a query; the nRF52840 sends its own data block in its response. | Data blocks are transmitted between devices. | Establish bidirectional transmission of data between transceiver and computer via a USB interface. |
| 70.3 | Functional Test | An initiating transceiver sends a query to the reacting transceiver. When the reacting transceiver receives the query, it sends a response. The initiating transceiver awaits the response. | A response to the query is received by the initiating transceiver. | Establish bidirectional transmission of data between transceivers. |
| 70.4 | Functional Test | An initiating transceiver hashes a data block. The data block is then transmitted over ESB to the reacting transceiver. Upon reception of the data block, the reacting transceiver hashes the data block. A hash exchange then occurs. If the hashes disagree, retransmission occurs until the hashes agree. | The data block and hashes are in accord between transceivers. | Establish bidirectional transmission of data between transceivers with hash exchange. |
| 70.5 | Functional Test | An initiating transceiver possesses a series of data blocks that make up the data of an image. The initiating transceiver begins a process of transmitting a data block, the reacting transceiver then receives that data block, and then a hash exchange occurs for that block; retransmission for a data block occurs until the hashes agree between transceivers. The process repeats until all data blocks have been received and their hashes are correct. | All data blocks and hashes are in accord between transceivers. | Establishes bidirectional transmission of sequentially packaged data with hash exchanges between transceivers. |
| 70.6 | Functional Test | An image existing on a computer is converted to a bitstream and sent to the initiating transceiver via USB. The initiating transceiver then hashes that data and transmits it to the reacting transceiver. The reacting transceiver hashes the data it received. A hash exchange occurs. Retransmission occurs until both the initiating and reacting transceivers agree that the hashes are in accord. Once this is true, the received data is reformatted as an image. | The image on the reacting transceiver end is identical to the image transmitted by the initiating transceiver and their hashes agree. | Establish communication of image data wirelessly with verification. |
| 70.7 | Integration Test | The initiating and reacting receivers are positioned on either side of the air gap. 10 images are communicated from the Raspberry Pi to the initiating transceiver as bitstreams via USB. The initiating transceiver hashes each image bit stream and then transmits the bit stream to the reacting transceiver. A hash exchange occurs. Retransmission occurs if the hashes disagree between transceivers. Once the hashes agree, the next image bit stream is transmitted. This continues until all image bits streams have been transmitted and verified via hashes. After an image bitstream has been verified, it is sent to the rebel server via USB and reformatted as an image. | All 10 received image files are identical to the transmitted ones and their hashes agree. | Establish viability of proposed communication system in its indented operating environment. The images can be communicated and verified over the channel and reconstructed on the reacting transceiver side. |
| 80.1 | Functional Test | Encrypt 1024x1024 PNG of death star on the raspberry pi utilizing a key length >= 64 bits. | Image is properly encrypted. | Image is properly encrypted. |
| 80.2 | Functional Test | Decrypt 1024x1024 encrypted PNG of death star on the Linux server utilizing a key length >= 64 bits. | Image is properly decrypted. | Image is properly decrypted. |
| 80.3 | Functional Test | Encrypt MD5 checksum on the Linux server utilizing a key length >= 64 bits. | Checksum is properly encrypted. | Checksum is properly encrypted. |
| 80.4 | Functional Test | Decrypt an encrypted MD5 checksum on the Raspberry Pi utilizing a key length >= 64 bits. | Checksum is properly decrypted. | Checksum is properly decrypted. |
| 90.1 | Functional Test | Transmit encrypted 1024x1024 PNG of the death star from the Raspberry Pi to the Linux Server. | Encrypted image is accurately received. | Encrypted image is accurately received. |
| 90.2 | Speed Test | Transmit 10 encrypted 1024x1024 PNG images of the death star from the Raspberry Pi to the Linux Server and measure time. | All 10 images are accurately transmitted in < 600 seconds. | All 10 images are accurately transmitted in < 300 seconds |
| 100.1 | Functional Test | Transmit an encrypted MD5 checksum of a received image from the Linux server to the /Raspberry Pi. | Raspberry Pi accurately receives the encrypted checksum. | Raspberry Pi accurately receives the encrypted checksum. |
| 100.2 | Functional Test | Compare the decrypted received checksum against the checksum of the original image transmitted. | Checksum matches expected value. | Checksum matches expected value. |
| 110 | Functional Test | Send an incorrect checksum from the Linux server to the Raspberry Pi and verify that the Pi resends the image. | The Raspberry Pi re-transmits the appropriate image. | The Raspberry Pi re-transmits the appropriate image. |
| 120 | Functional Test | Give the Linux server a 1024x1024 PNG image of the death star with a highlighted weakness and confirm that the software accurately crops the image down to the red circle. | Image is properly cropped to expected output. | Image is properly cropped to expected output. |
| 130.1 | Functional Test | The 10 cropped images are uploaded from the Linux server to the webserver of the mobile app. | The Linux server connects to the web server of the mobile app and uploads the images. | The 10 cropped images are successfully uploaded from the Linux server to the web server for the mobile app. |
| 130.2 | Functional Test | The mobile app accesses the web server and displays the grid of cropped images. | The web server displays the received images in a grid table. | The mobile app successfully accesses the web server and displays the grid of cropped images. |

A black and white list with text

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Constraint | Test Method | Evaluation Method | Threshold | Objective |
| 10 | *Visual Inspection* | *Visually inspect the Linux server and the raspberry pi to check for physical connection* | *No physical connection to the Linux Server* | *No physical connection to the Linux Server* |
| 20 | *Visual Inspection* | *Visually inspect the boundary windows to ensure nothing is connected to it.* | *No physical connection* | *No physical connection* |
| 30 | *Functional Test* | *Check the RF module to ensure it’s being used, and also check each part of the project to ensure that Bluetooth, cellular, nor any other protocols are being used.* | *Wi-Fi, Bluetooth, cellular, nor any other associated protocols is used* | *Wi-Fi, Bluetooth, cellular, nor any other associated protocols is used* |
| 40 | *Budget Inspection* | *Check our orders, and spreadsheet to ensure that we don’t exceed $300 USD.* | *Must not exceed budget of $300 USD* | *Verify that the total project shall not exceed $300 USD.* |
| 50 | *Rubric Inspection* | *Check our components and track where we got them from.* | *All materials must come from course instructors* | *All materials must come from course instructors* |
| 60 | *Visual Inspection* | *Measure the server to ensure that it’s at least 5 meters or longer from the window.* | *Is greater than or equal to 5 meters from glass* | *Is greater than or equal to 5 meters from glass* |
| *70* | *Functional Test* | *Use Wireshark or any other application to ensure that the project is not interfering with anything that was not intended to.* | *No disruption to other activities* | *No disruption to other activities, and no one knows about the data transmission.* |
| *80* | *Functional Test* | *Visually check that communication is being sent and received from the raspberry pi and the server.* | *Is bi-directional* | *Is bi-directional* |
| *90* | *Functional Test* | *Ensure that AES Encryption is being enforced during the transmission.* | *Data is encrypted* | *Data is encrypted* |
| 100 | *Functional Test* | *Time the total length of time that it takes the images to transfer to the server and verify it.* | *The 10 1024x1024 images of the death star within 600 seconds* | *The 10 1024x1024 images of the death star within 300 seconds* |

A screenshot of a computer error

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Standard | Test Method | Evaluation Method | Threshold | Objective |
| 10 | Analysis | Data is encrypted | Encryption performed accurately follows the outlined AES standards | All encrypted communication between systems adheres to AES standards of encryption |

A white text on a black background

Description automatically generated

***Summary Tables of Test Results***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Requirement* | *Test Date* | *Test and Evaluation Location* | *Result* | *Notes* | *Date Passed (Accepted)* |
| *1.1* |  |  |  |  |  |
| *1.2* |  |  |  |  |  |
| *2.1* |  |  |  |  |  |
| *⁞* |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Constraint* | *Test Date* | *Test and Evaluation Location* | *Result* | *Notes* | *Date Passed (Accepted)* |
| *1.1* |  |  |  |  |  |
| *1.2* |  |  |  |  |  |
| *2.1* |  |  |  |  |  |
| *⁞* |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Standard* | *Test Date* | *Test and Evaluation Location* | *Result* | *Notes* | *Date Passed (Accepted)* |
| *1.1* |  |  |  |  |  |
| *1.2* |  |  |  |  |  |
| *2.1* |  |  |  |  |  |
| *⁞* |  |  |  |  |  |
|  |  |  |  |  |  |

***Test and Evaluation Methods***

*Testing of requirements is to validate the system’s achievement of a capability or level of performance. Testing of constraints is to verify that the system’s properties, capabilities or levels of performance conform to limits set by of the constraints. Testing to standards is to verify that the system’s capabilities, function or levels of performance comply with industry standards.*

*The following examples illustrate simple approaches for writing descriptions of test and evaluation methods to verify achievement of a requirement, to verify the system conforms to constraints, and to verify the system complies with standards.*

***Requirements (verification of achievement)***

*Requirement 1.1 will be tested at Facility X which hosts equipment Z and test chamber Y. The test method is as follows: Subsystem J will be monitored using a P test meter at standard temperature and standard pressure for H hours of continuous operation. Specialized test equipment and test fixtures are not required. The results will be evaluated by direct observation of the tests. The threshold is U units and the objective is V units.*

*Requirement 2.1 will be tested at Facility X which hosts equipment Z and test chamber Y. The test method is as follows: Subsystem J will be monitored using a P test meter at standard temperature and standard pressure for H hours of continuous operation. A specialized test connector Fitting K is required and the results will be evaluated by indirect observation. The Value at test port A will be processed using a discrete Fourier transform and frequency domain analysis. The threshold is U units and the objective is V units.*

*….*

***Constraints (verification of conformity)***

*Constraint 1.1 will be tested at Facility Y which hosts Windows- and Linux-based laptops, desktops and workstations. The test method is as follows: Graphical User Interface (GUI) software for subsystem H will be compiled on OS D without error or warnings. Each of the GUI features will be demonstrated following Use Cases A, B, C as described below. The threshold is zero run-time errors. The objective is zero run-time errors.*

*Constraint 1.2 will be tested at Facility Y which hosts Windows- and Linux-based laptops, desktops and workstations. The test method is as follows: Following successful demonstration of Requirement 2.1.1, Algorithm W will be tested using the Graphical User Interface (GUI) software for subsystem H. Each of the GUI fields values will be compared against the expected values following Use Cases A, B, C as described below. The threshold is accuracy of +/-X units. The objective is +/-X/2 units.*

*…*

***Standards (verification of compliance)***

*Standard 1.1 will be verified in accordance to IEEE Standard 102.11C dated Sep. 2016. The tests will occur at Facility X which hosts equipment Z and test chamber Y. Specialized test equipment and test fixtures are not required. The results will be evaluated by direct observation of the tests. The threshold is U units and the objective is V units.*

*Standard 1.2 will be tested for compliance with IEEE 802.11 a/b/g/n/ac at the Interoperability Laboratory at the University of New Hampshire. The cost of this test service will be $X, and will be completed along with a compliance report within Y weeks.*