DICE and MUD Protocols for Securing IoT Devices

Muhammed Ayar

DICE and MUD Protocols for Securing IoT Devices

White Paper

Author: Muhammed Zahid Ayar, Muhammed.Ayar@student.sans.edu
Advisor: Randy Marchany
Accepted: (Date your final draft is accepted by your advisor)

Abstract

An exponential growth of Internet of Things (IoT) devices on communication networks is creating an increasing security challenge that is threatening the entire Internet community. Attackers operating networks of IoT devices can target any site on the Internet and bring it down using denial of service attacks. As exemplified in various DDoS attacks that took down portions of the Internet in the past few years (such as the attacks on Dyn and KrebsOnSecurity (Hallman, Bryan, Palavicini Jr, Divita, Romero-Mariona, 2017)), IoT users need to take drastic steps in securing them. This research will discuss the steps in attempting to secure IoT devices using DICE and MUD.

1. Introduction

Identification of networked assets is the first step in securing any network. With IoT devices, the asset inventory problem is also growing exponentially, and there is an urgent need for efficient and positive identification of these devices on the network for properly configuring, managing, and updating them. With the rapid growth of the number of IoT devices and significant growth expected in the future, attackers view these devices as relatively defenseless and compromise them to use in their attacks (Zelonis, 2018). Once attackers control networks of IoT devices, they are able to attack any target on the Internet to bring it out of service, and therefore put the overall healthy functioning of the Internet at risk. By using the increasing number of vulnerable IoT devices available online, various DDoS attacks occurred that took down significant portions of the Internet such as the attacks on Dyn and KrebsOnSecurity. With the numbers of IoT devices predicted to rapidly rise even further, it is essential to take drastic steps in securing these devices (Hallman, Bryan, Palavicini Jr, Divita, Romero-Mariona, 2017).

1.1 Overview of DICE and MUD

There are a number of efforts underway by different organizations to enumerate and manage IoT devices securely. Microsoft's Device Identifier Composition Engine (DICE) and Cisco's Manufacturer Usage Description (MUD) are such efforts. Microsoft's Device Identifier Composition Engine – DICE (used to be known as RIOT), is a hardware and software- based family of techniques that attests the health of the hardware and software on IoT devices (Microsoft, 2015). IoT devices that employ cryptography utilize a private key called a Unique Device Secret (UDS) to secure their operation. Malicious actors may leak the key by compromising the code on the chip. Obtaining the key can lead the attacker to impersonate the device and replace the firmware (Elliptic Semiconductor Inc., 2015). Therefore, it is essential to prevent disclosure of the UDS. DICE implements three measures to help secure the UDS.

First, the power-on latch locks read access to the UDS before early boot-code transfers control to subsequent execution layers. Second, even if power-on latch prevents reading the UDS, copying the UDS as-is to memory by the early boot code could lead to it's disclosure from RAM. A cryptographic one-way function makes a hash of the UDS to store in RAM so that in the event of RAM disclosure, the original UDS is safe. Now using the transformed UDS key, the original UDS will not be revealed if subsequent malicious actions compromise the RAM or later code. Third, to prevent compromise of the device by attempts to modify the early boot-code, the cryptographic one-way function uses a measurement of the boot code as input together with the UDS. The function outputs a key called the Compound Device Identifier taking both the UDS and early boot code measurement as input (optionally taking the hardware state and configuration as input as well). This process ensures that modification of early boot code generates a new key so that the original UDS is secure (Microsoft, 2015).

Cisco MUD is an extension to DHCP (as well as LLDP and 802.1x) that includes a URI to the IoT device's manufacturer. The device uses this URI to download configuration information that defines what the device is supposed to do. This configuration helps prevent the IoT device from participating in DDoS in the event of device compromise. The policy set for the device on its network access point would only allow traffic from it to go where, and in the manner the manufacturer specifies, and not to the malicious actor's DDoS target.

1.2 Choosing protocols

With different approaches available to help secure IoT devices, some may be infeasible due to cost but may be advantageous in other aspects. Each organization that utilizes IoT devices faces the problem of intelligently choosing which approach is most suitable for their networks.

As this research shows, implementing security techniques on different devices can be difficult; therefore, it is essential for there to be practical guides to facilitate adopting techniques in securing IoT devices. This research will clarify the various advantages and

disadvantages of DICE and MUD as demonstrated through experimentation to help organizations choose the best approach in securing their IoT devices.

2. Method

This research attempts to test Microsoft DICE and Cisco MUD on IoT devices using the Rapid7 IoT testing methodology (Heiland, Sevier, Littlebury, 2017) as a means of helping organizations choose the most effective IoT security protocol for their needs.

2.1. Testing Steps

The first step is mapping out the IoT ecosystem of the setup. As part of the end-to-end ecosystem methodology, this involves mapping the device, related sensors and actuators, related mobile apps and control software, Cloud API and associated web services, and network communication protocols used such as Ethernet, 802.11 Wi-Fi, and Bluetooth.

The second step, as outlined in the Rapid7 IoT device testing methodology, involves setting up DICE and MUD protocols to run on the following IoT devices:

- Cloud JAM
- CEC1702 IoT Development Kit

Both these devices support Microsoft DICE by including specific hardware cryptography features. Cisco MUD, in theory, should run on all IoT devices including these two.

Following Rapid7's methodology, 'Functional Testing' of devices takes place to ensure that they are working as intended. 'Open Intelligence Gathering' involves

performing recon on device components (to use for later security testing of protocols), such as subsystems, communication types, and radio frequencies.

After these steps, 'Capture & Analysis of Data Communication' to and from devices is performed using proxies between end-points (such as between the device and the Cloud or between the mobile application and the Cloud) while looking for vulnerabilities in communications. Vulnerabilities could arise from the external entities to which the devices may attempt to communicate. For example, devices making undocumented call-outs to strange addresses or domains; identifying exposed ports and checking for improper misconfigurations; testing wireless attacks such as subverting pairing and replay attacks; and DoS attack attempts.

'Vulnerability Testing of Mobile and Control Applications' follows after capturing and analysis of data communications, by looking for weak authentication, improper storage of data, and conformity to standard encryption methods.

3. Findings and Discussions

3.1. CloudJAM

RushUp IoT company provides CloudJAM, a product that supports Microsoft DICE. CloudJAM is a small device that incorporates temperature and motion sensors and sends this data as well as receives commands from and to a Cloud application through Wi-Fi. The testing below will be based on the Rapid7 IoT testing methodology.

3.1.1. IoT ecosystem of the CloudJAM

CloudJAM has a board called the NUCLEO-F401RE Nucleo that has the following Nucleo Expansion Boards built-in:

X-NUCLEO-IDW01M1 – Wi-Fi capabilities

X-NUCLEO-IKS01A2 – environmental and motion sensor

X-NUCLEO-NFC01A1 – NFC sensor

The device has an embedded debugger to allow development and debugging. This feature allows developing firmware and then flashing onto the device.

The CloudJAM device has the following features:

- Micro-USB supplies 5V power
- Integrates ST-Link V2 in-circuit debugger and programmer
- STM32F401RET6 microcontroller that has an ARM Cortex M4 processor
- Humidity and temperature sensor - HTS221
- Nano pressure sensor - LPS22HB MEMS
- 3D accelerometer and 3D gyroscope. - LSM6DSL iNEMO inertial module
- High-performance eCompass - LSM303AGR
- 2.4 GHz IEEE 802.11 b/g/n Wi-Fi - SPWF01SA-11 module.

- 64-Kbit Dynamic NFC / RFID - M24SR64-YMN6.
- 4 push buttons
- 9 LEDs
- 50x50x6mm size with mounting holes
- Stores SSID and password in NFC component for Wi-Fi connection
- Can utilize ST25 NFC Tap mobile Android app from Google Play Store to set Wi-Fi credentials

3.1.2. Setting up the CloudJAM

The CloudJAM IoT device is set up to run Microsoft DICE over Azure IoT hub using the steps in Appendix 1. Cisco MUD testing requires a MUD Manager to accept and process the MUD URL from the IoT device. Figure 1 displays the MUD procedure:

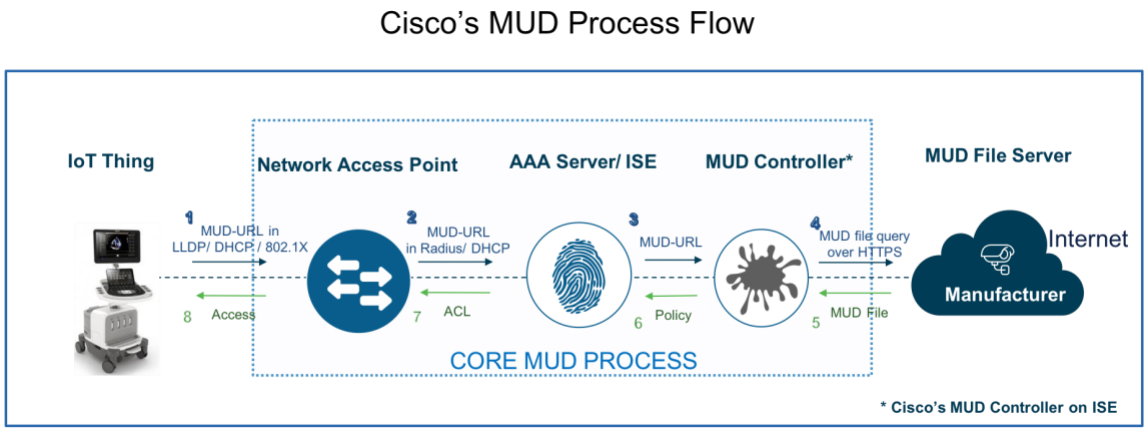

Figure 1: MUD Process Flow (Why MUD?, n.d.)

As the diagram above shows, the IoT device utilizes Cisco MUD:

- Emits either an 802.1x, LLDP, or DHCP request to its network access point such as router or switch. This request houses a MUD URL.

- The network access point should be able to know how to handle the request that contains the MUD URL. Currently, the only Cisco network access points that support MUD URLs are the Catalyst switches whose price points start at around $1K and can cost upward of $15K or more.
- After receiving the request, the Catalyst switch extracts the MUD URL, encapsulates it in a RADIUS packet, and sends it to an Authentication, Authorization, and Accounting (AAA) server. Cisco's AAA server is the Identity Services Engine (ISE). The base license for Cisco ISE costs around $500.
- The AAA server passes the MUD URL to the MUD Controller which is also contained within Cisco ISE.
- The MUD Controller uses HTTPS to contact the server hosting the MUD file that the MUD URL indicates.
- After verifying that the MUD file is from the device's manufacturer, the MUD Controller retrieves a copy of the MUD file. The MUD file is a YANG data model using JSON representation and specifies the manufacturer's intention for the IoT device's and communication patterns.
- The MUD Controller translates the manufacturer's specifications and permissions for the device into context-specific policies given to the AAA Server/Cisco ISE.
- The ISE server enforces the policies in the form of Access Control Lists that it sends to the Network Access Point.

Testing the MUD server using Cisco's proprietary hardware and software can cost at least $1500 for a Catalyst switch and the Cisco ISE software. Appendix 2 lists steps for an alternative setup to perform Cisco MUD tests using the open source osMUD (MUD manager) running on an OpenWRT virtual router image.

3.1.3. Functional testing the CloudJAM

To test the functionality of CloudJAM, the CloudJAM device is started up by powering it with a USB cable, and it automatically starts sending messages to the Azure IoT hub. We also test sending messages from the Cloud (Azure IoT hub) to the device. When starting the CloudJAM, it first starts the procedure of connecting to an SSID that is using a pre-set configuration on the device. It gives an option to modify the SSID configuration by holding down the 'USER' button as the device is starting.

The device contains the following nine status LEDs:

- TLK – bicolor status LED
- 5V – showing power supply presence
- USR – user free LED
- WLD – Wi-Fi blink (Blue LED)
- WUP – Wi-Fi power up (Red LED)
- WLK – Wi-Fi link
- PIN – iNEMO interrupt 2
- INT - iNEMO interrupt 1
- WTX – Wi-Fi TXD1

During initialization, only the 5V and the WTX LEDs are luminous. While the AP is setting up and waiting to connect to AP, the WLK LED is on. The WLD and WUP LEDs blink on/off three times: twice after initialization message: '[Init]. Initilized WiFi data structures WiFi: WAIT FOR Module Activation Procedure...' and shows up on UART output, and then once after '[Init]. Try to Connect to SSID: <SSID redacted>...' shows up on UART output. After synchronizing with the NTP server, the USR LED lights up and messages are sent to the Azure IoT Hub.

The Azure IoT Hub Device Explorer tool (Azure IoT SDK – Csharp, 2019) allows the user to manage the devices connected to the Azure IoT hub as well as receive

and send messages to the devices. The 'Management' tab in Device Explorer shows devices that are setup (provisioned) in the user's Azure IoT hub as well as their status. Even though the CloudJAM (with name 'rushup' in Azure IoT Hub provisioning, in this case) is connecting and communicating with the Azure IoT Hub, for some reason, the Management tab in the Device Explorer still shows the ConnectionState as 'Disconnected':

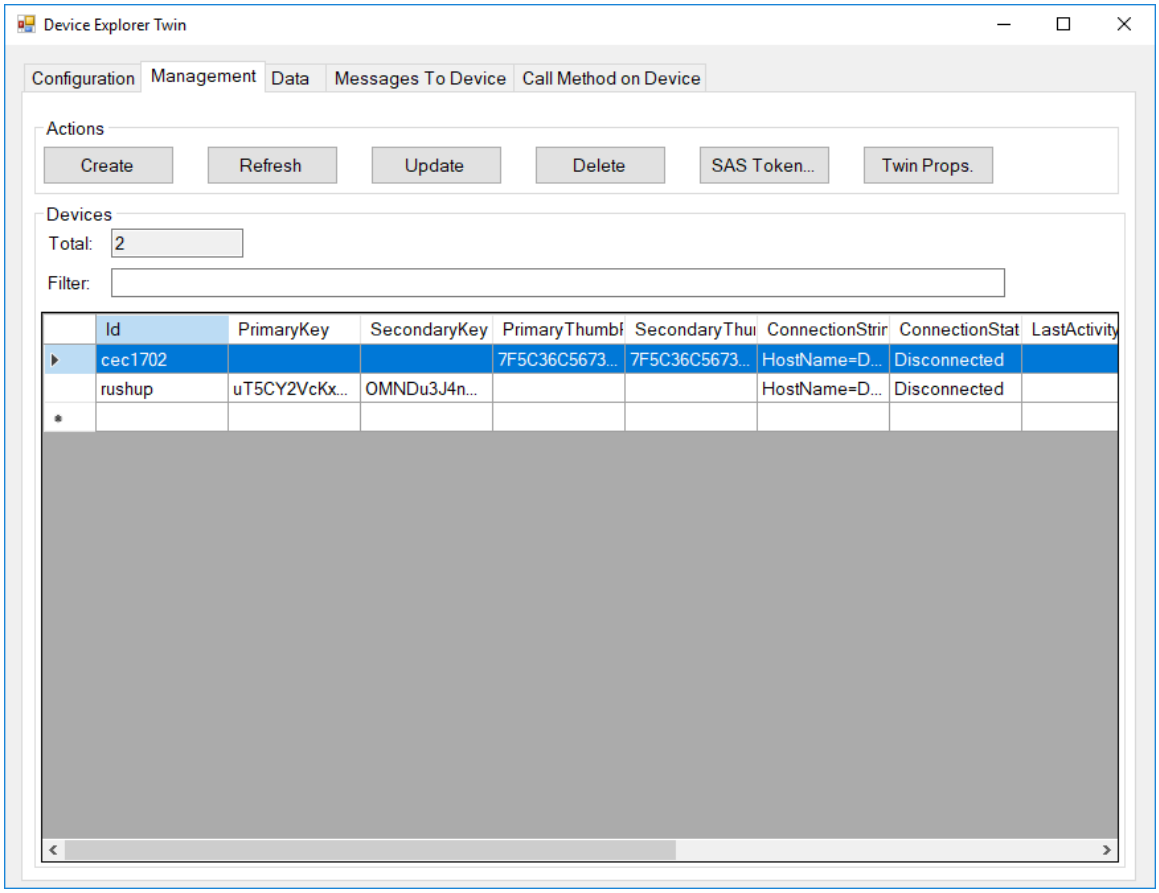

Figure 2: Management Tab in Device Explorer

When pressing the 'Refresh' button on Device Explorer, sometimes the PrimaryKey and SecondaryKey values disappear for the rushup (CloudJAM) device, and sometimes they reappear. The cause of this result is still unknown. On the 'Data' tab of Device Explorer, it shows the messages that Azure IoT hub is receiving from the device:

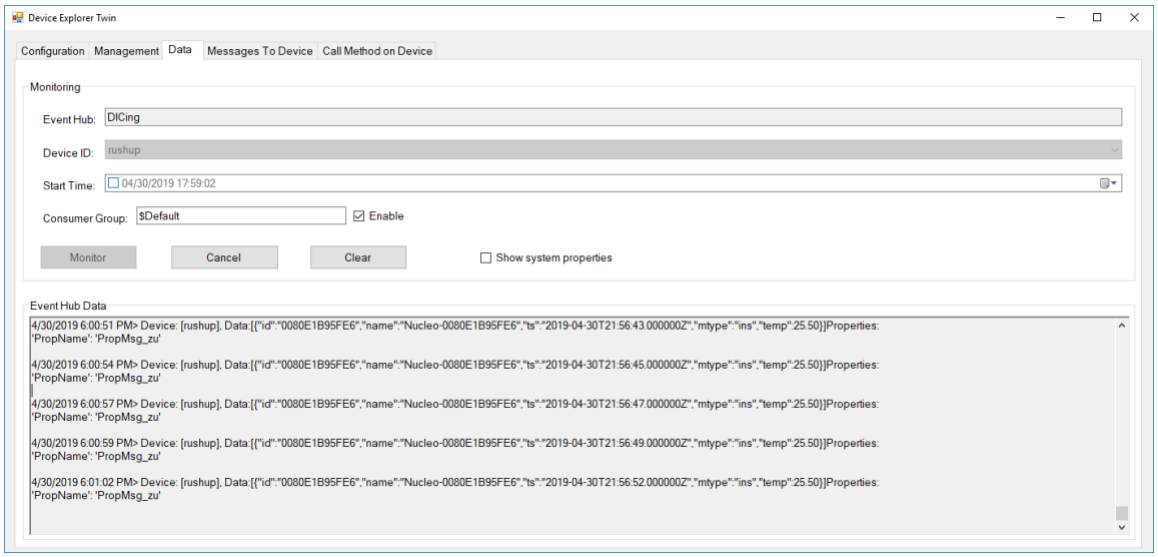

Figure 3: Above are part of the messages seen in the UART analysis of the device communication discussed in Section 3.1.5 on Capture and Analysis of CloudJAM Data Communications.

Some messages from CloudJAM the device to the Cloud include:

- Current Temperature: 25.5 degree Celsius
- The device is not sending motion (acceleration) data, pressure, and humidity to the Cloud possibly due to a failure to initialize sensor board, as shown in the output in Section 3.1.5.
- From the 4 push-buttons on the device:
    - Pressing the ‘USR’ user button doesn’t do anything when the device is running, however, when device initially starts, pressing and holding the user button allows the user to set up the SSID credentials for connecting to the Wireless Access Point (AP).
    - Pressing the ‘RST’ reset button resets the device to initialize and restart its connection process to an AP, and also starts sending messages containing information regarding the temperature and other data to the Azure IoT hub.

- Pressing the 'WRST' button lights up the WLD and WUP LEDs and then stops Wi-Fi transmissions by sometimes displaying 'WiFi error' on UART output. Transmissions sometimes start up again after some time.
- Pushing the 'WUSR' button during initialization results in 'WiFi AP ready' message showing in UART output.

Sending messages from the Cloud (either from Azure Portal or Device Explorer) to the device include:

- reset – unlike the physical 'RST' button on the device, sending 'reset' message from Cloud only resets the device when connected to the Cloud
- quit – stops the application and therefore, the sending of further messages from device to Cloud. The device does not accept further messages from the Cloud after this without a hard reset with the push button.
- led on - turns on USER LED
- led off – turns off USER LED

3.1.4. Open Intelligence Gathering from the CloudJAM

CloudJAM is made up of an STMicroelectronics Open Development Environment board called the NUCLEO-F401RE Nucleo. This board has an STM32 microcontroller, STM32F401RET6 ARM Cortex M4 with FPU & DSP, 96 kB SRAM, 512 kB FLASH, 84 MHz CPU, and ART accelerator, running in an LQFP64 package. It has an onboard ST-LINK/V2-1 programmer/debugger using USB re-enumeration with 3 interfaces: Virtual COM port, debug port, and mass storage. It also has an ST-LINK USB Vbus or external power supply option, 1 user LED shared with Arduino for a total of 9 LEDs, 1 MCU reset, 1 Wi-Fi module reset, 1 MCU user, and 1 Wi-Fi user push-buttons. It also

has a 32.768 kHz LSE crystal oscillator; supports Arduino Uno V3 and ST Morpho board expansion connectors; supports IAR, Keil, ARM Mbed, and GCC-based IDEs; has a 5V/0.5A USB micro power input; Additionally, CloudJAM has three STM32 Nucleo Expansion Boards whose mapping out is in the paragraphs below.

The first Nucleo Expansion Board is the X-NUCLEO-IDW01M1 – which provides Wi-Fi capabilities using the SPWF01SA module. The SPWF01SA module can transmit/receive IEEE 802.11 b/g/n using low power at 2.4 GHz. It has an STM antenna and runs on an STM32 ARM Cortex-M3 processor with 64 KB RAM and 512KB of FLASH memory. It has 1 MB extended flash for FOTA. The device has simple AT command UART interface to update Firmware; TCP/IP, TLS/SSL integrated protocol stacks; supports 8 simultaneous TCP/UDP clients and one socket server. It also supports one socket client with TLS/SSL up to TLS 1.2 including encryption algorithms: AES 128/256, MD5/SHA-1/SHA-256 hashes, and RSA/ECC PKI. CloudJAM also has application layer functionality that supports web servers serving dynamic web pages. It supports REST APIs to receive and post web content. CloudJAM supports WEP/WPA/WPA2 security; station, IBSS, and miniAP (supporting 5 stations) system modes; transmits 18.3 dBm @ 1Mbps DSSS and 13.7 dBm @ 54 Mbps OFDM; receives -96.0 dBm @ 1 Mbps DSSS and -74.5 dBm @ 54 Mbps OFDM. It has 16 configurable GPIOs. CloudJAM also supports advanced low-power modes with a standby Real Time Clock (RTC) operating at 43 µA. Sleep mode DTIM=1 running at 15 mA; receiving traffic typically at 105 mA; and transmitting traffic typically at 243 mA @ 10 dBm (low-power mode). CloudJAM includes a single voltage supply at 3.3 V; has a temperature range of -40 °C to 85 °C; is FCC/CE/IC/SRRC certified and RoHS compliant; has the ST morpho and the Arduino UNO R3 connector layouts; is compatible with the STM32Cube software development tools.

The second Nucleo Expansion board - X-NUCLEO-IKS01A2 - senses motion and environmental conditions such as pressure and temperature. It contains the LSM6DSL 3D accelerometer (±2/±4/±8/±16 g) along with a 3D Gyroscope (±125/±245/±500/±1000/±2000 dps). It has the LSM303AGR Microelectromechanical

systems (MEMS) 3D magnetometer (±50 gauss) and MEMS 3D accelerometer (±2/±4/±8/±16 g). It also has an HTS221 capacitive digital humidity and temperature sensor; the LPS22HB MEMS pressure sensor 260-1260 hPa digital barometer; an I2C PIN to interface with the STM32 microcontroller; a DIL23 socket to accommodate additional MEMS adapters and/or other sensors; it is also compatible with STM32Cube firmware; the LSM6DSL has I2C sensor hub features; has an Arduino UNO R3 connector and is RoHS compliant.

The third Nucleo Expansion board is the X-NUCLEO-NFC01A1 which is a dynamic NFC tag evaluation sensor. It contains the M24SR64-Y dynamic NFC/RFID tag Integrated Controller with 64 kbit EEPROM memory. It also has an NFC antenna of size 31 mm x 30 mm operating at 13.56 MHz; comes with an Arduino UNO R3 connector, 3 general purpose color LEDs; and is powered by Arduino UNO R3 connectors.

3.1.5. Capture and Analysis of CloudJAM Data Communications

Connecting the CloudJAM device to USB power gives the following communications output through UART serial connection:

```
[Init]. Starting configuration procedure for SSID and PWD....
[Init]. Keep pressed user button to set Wi-Fi Access Point parameters (SSID and PWD) from NFC (if mounted) or via serial terminal. Otherwise parameters saved to FLASH will be used.
[Init].
[Init]. Read from FLASH:
        SSID =<SSIDname redacted>
        Key  =<WiFi pwd redacted>
        Authentication  =WPA2

[Init]. AP settings set.
```

```
      [Init]. Initilized WiFi data structures WiFi: WAIT FOR Module
Activation Procedure...


      [Init]. Wi-Fi Module Activated

      [Init]. WiFi MAC Address is: 00:80:E1:B9:5F:E6

      [Init]. Try to Connect to SSID: <SSIDname redacted>...

      [Init]. WiFi connected to AccessPoint


      [Version]. FP-CLD-AZURE1 version : 02.00.00.00

      [Version]. Microsoft Azure IoT SDK version 1.0.11

      [Version]. Openstm32 Compiler Platform

      [Version]. Build Date: Mar 11 2019 Time: 14:45:30

      [Version]. STM32 Nucleo Build.

      [Version]. Built for X_NUCLEO_IKS01A1 sensor board.

      [Version]. Starting IoTHub HTTP Client application.


      [IotHub]. Launching NTP procedure.

      [IotHub]. Connected with NTP Server: time-d.nist.gov

      [IotHub]. Set UTC Time: Thu Apr 18 04:03:44 2019


      [IotHub]. Sync with NTP server completed.


       Socket already close[IotHub][E]. Failed init sensors (Accelero).

      [IotHub]. Starting the IoTHub client sample HTTP...

      [IotHub][E]. Failed to init sensor board, using dummy data.

      [IotHub][E]. Failure to set option "timeout"

      [IotHub]. IoTHubClient_LL_SetMessageCallback...successful.
```

```
      [IotHub]. Sending message: {"id":"0080E1B95FE6","name":"Nucleo-
0080E1B95FE6","ts":"2019-04-
18T04:03:45.000000Z","mtype":"ins","temp":25.50}

      [IotHub]. IoTHubClient_LL_SendEventAsync accepted message [0] for
transmission to IoT Hub.

      [IotHub]. Confirmation[0] received for message tracking id = 0
with result = IOTHUB_CLIENT_CONFIRMATION_OK

      [IotHub]. IoTHubClient_LL_DoWork...sent data.

[IotHub]. Sending message: {"id":"0080E1B95FE6","name":"Nucleo-
0080E1B95FE6","ts":"2019-04-
18T04:03:51.000000Z","mtype":"ins","temp":25.50}
```

To capture the packets that CloudJAM is sending to the Access Point, a Panda Wireless PAU06 300Mbps Wireless-N USB Adapter is being used with a high-gain antenna that is operating in monitor mode. A separate adapter is used so that the laptop adapter doesn't need to be changed to monitor mode and so that it is not busy capturing packets in addition to being used for managing the Azure IoT hub and other wireless traffic. The packet capture is run using a Linux VM to set up the PAU06 in monitor mode and Wireshark is used to capture packets. The Kali Rolling version during experimenting is 4.15.0-kali3-amd64 #1 SMP Debian 4.15.17-1kali1 (2018-04-25) x86_64 GNU/Linux. Appendix 3 lists the steps used to put the PAU06 device in monitor mode.

By analyzing the packet capture of the CloudJAM device upon connection, a probe request is observed from the CloudJAM device with the onboard SSID that the user sets. After that, a probe response comes from the router. Using the Wireshark filter "wlan.addr == 00:80:e1:b9:5f:e6" limits the packet display in Wireshark to CloudJAM communication by using the MAC address of the CloudJAM device.

Source	**Destination**	**Byte Length**	**Message**
Stmicroe_b9:5f:e6	Broadcast	112	Probe Request, SN=217, FN=0, Flags=....., SSID=<SSID redacted>
36:1f:e4:e4:44:8b	Stmicroe_b9:5f:e6	240	Probe Response, SN=4039, FN=0, Flags=....., BI=100, SSID=<SSID redacted>
Stmicroe_b9:5f:e6	36:1f:e4:e4:44:8b	48	Authentication, SN=12, FN=0, Flags=.....
	Stmicroe_b9:5f:e6 (00:80:e1:b9:5f:e6)	28	Acknowledgement, Flags=.....
36:1f:e4:e4:44:8b	Stmicroe_b9:5f:e6	59	Authentication, SN=791, FN=0, Flags=.....
Stmicroe_b9:5f:e6	36:1f:e4:e4:44:8b	110	Association Request, SN=13, FN=0, Flags=....., SSID=<SSID redacted>
	Stmicroe_b9:5f:e6 (00:80:e1:b9:5f:e6) (RA)	28	Acknowledgement, Flags=.....
36:1f:e4:e4:44:8b	Stmicroe_b9:5f:e6	98	Association Response, SN=792, FN=0, Flags=.....

Table 1: CloudJAM communication to Access Point

The above packet capture summarizes the initial communication between the CloudJAM and the router access point. After an initial probe request and response, the device sends an authentication packet with the authentication algorithm set to Open System. The AP replies with an authentication response giving more information such as

Vendor information, in this case, Broadcom. The CloudJAM device sends an association request packet and receives a response from the AP followed by the four-way WPA handshake that occurs between the client device and AP and then the client device sends a DHCP packet to the AP with IP value set to 0.0.0.0.

The remaining communication in the packet capture between the router and the CloudJAM device looks unintelligible upon observation and most likely utilizes an encryption algorithm. However, this data most likely contains the information that is output when viewing the UART output above from the USB port that connects the CloudJAM device to the host machine. Additionally, performing any of the device functionality such as sending commands from Cloud to the device to turn LED on/off, resetting, and quitting or pressing push buttons, did not generate legible packet traffic.

3.1.6. Vulnerability Testing of Control Applications

During this phase of testing, vulnerabilities in authenticating/connecting to Azure IoT hub (DICE) and MUD are being sought out.

- When testing input for the SSID and password while connecting to the access point, if a password of 189 characters is input (without even pressing 'Enter'), then the application locks up and can no longer process any other input.
- If a slightly smaller password is input, one of 184 characters, then pressing 'Enter' moves to the next question of security type and when '2' is input by the user, the application locks up and does not proceed. Packet capture shows that at this point the only traffic is comprised of probe requests and responses along with null function responses from AP to the client.
- Testing ST25 NFC TAP Mobile application to transfer SSID configuration didn't work.
- In the limited source code review, no vulnerabilities were discovered.
- A possible avenue of testing could involve attempting to update the firmware from a 'malicious' source by using simple AT commands given through the UART interface or by using REST APIs mentioned in the open source intel gathering for the device in Section 3.1.4.

- Another possible route is to set up a 'malicious' AP with the same SSID as the one configured in the device but with higher power so that the device connects to it. This AP can be used in attempting to decrypt the traffic to Azure IoT hub using a MITM attack. Using a bridging the air gap attack, lateral movement from a compromised host in an organization to a wireless AP may allow for the modification of the firmware and then the unencrypted traffic to be sent to a remote attacker.
- After much back and forth with the manufacturer, the CloudJAM device is still unable to emit the DHCP option header to test Cisco MUD; there is no vulnerability testing of MUD for this reason.

3.2. CEC1702 IoT Development Kit

The CEC1702 device from Microchip is an evaluation board used to test various IoT applications using a Plug-in-Module (PIM) that also supports programming in keys for DICE authentication.

3.2.1. IoT ecosystem of the CEC1702 Kit

CEC1702 IoT Development Kit (Microchip Part # DM990013-BNDL) includes the following:

- CEC1x02 DevBoard including a Plug-in Module (PIM) with a CEC1702 chip for DICE
- Mikroe WiFi7 Click module with ATWINC1500 WiFi chip
- Mikroe Thermo5 Click module with EMC1414 Temperature sensor

3.2.2. Setting up the CEC1702 Dev Kit

The CEC1702 Dev Kit is set up to run Microsoft DICE over Azure IoT hub. osMUD is a MUD manager platform that takes a MUD URL and downloads the respective MUD file from the specified location to pass back to the Access Point connecting the IoT device. Please see Appendix 2 for setting up osMUD on OpenWRT for MUD testing. For setting up the CEC1702 Dev Kit to connect to the Azure IoT Hub, follow the steps in this official Microchip manufacturer guide's Getting Started tab: https://catalog.azureiotsolutions.com/details?title=CEC1x02DevBoard&source=null

3.2.3. Functional testing the CEC1702 Dev Kit

The Microchip ICD4 (Microchip Part # DV164045) debugs the CEC1702 as well as flashes the firmware onto it using the MPLAB IDE. From the 'Debug' menu, selecting 'Debug Project' in MPLAB starts the debug process by flashing the firmware onto the CEC1x02 Dev Board. The ICD4 is connected via JTAG 8-pin cable to the Dev Board using a debugger adapter (Microchip Part # AC102015). First, the ICD 4 connects to the laptop to power it on via USB, and its LEDs become purple when booting. After it is ready, the Dev Board plugs in via USB to power it up using micro USB. During firmware flash, the LED of the ICD4 connecting to the Dev Board becomes yellow. During the debug process, the output of the ICD4 is:

```
**************************************************



Connecting to MPLAB ICD 4...


Currently loaded versions:
Application version.............01.05.18
Boot version...................01.00.00
FPGA version...................01.00.00
```

```
Script version.................00.02.77
Script build number............5215401e64
Target voltage detected
Target device CEC1702 found.
Device Revision Id  = 0x84

Resetting...
Reset complete

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0xb0000, end address = 0xfafff
program memory: start address = 0x100000, end address = 0x101fff

Programming/Verify complete

Running
```

Analysis of the firmware code (app_control_led function in prov_dev_client_ll_sample.c), shows the following messages that the device can accept:

- led on
- led off
- led red
- led green
- led blue
- led yellow

- led magenta
- acq_time <seconds>

The LED messages are designed to turn on/off or change the color of the LED light. The acq_time <seconds> function is used to change the temperature acquisition interval of the device from its environment to a specified number of seconds.

3.2.4. Open Intelligence Gathering from the CEC1702 Dev Kit

The CEC1702 chip is an ARM Cortex M4 based microcontroller. It operates with 3.3V and 1.8V and has a complete ARM-standard of debugging support with a JTAG-based DAP port composed of SWJ-DP and AHB-AP Debugger access functions. The CEC1702 contains 64K of boot ROM as well 2 blocks of SRAM totaling 480KB (with each block being used either for program or data). The CEC1702 has up to 65 General Purpose I/O pins (GPIOs) powered by 1.8V each. It also has 2 standard 16C550 UARTs.

The CEC1x02 DevBoard has 480KB of RAM. The software used to program the firmware of the CEC1702 DevBoard is written in the C programming language. The device uses the MQTT protocol to communicate with the Azure IoT Cloud and has WiFi connectivity to the Internet using the Mikroe WiFi7 Click module. Its I/O hardware interfaces consist of GPIO and I2C/SPI. The DevBoard communicates with the Azure Device Provisioning Service using HTTPS and attests its identity using X.509 certificates. The Mikroe Click modules connect to the CEC1x02 DevBoard through MIKROBUS2 slots.

3.2.5. Capture and Analysis of CEC1702 Dev Kit Data Communications

This step involves collecting data from various sub-systems of the CEC1702 Dev Kit as it is running different tasks. Following that data analysis helps clarify how normal

data communication works and discover possible security vulnerabilities. For this, more information from the manufacturer is required to get the CEC1702 Dev Kit properly connecting to the Azure IoT Hub. Currently, it is experiencing difficulty with an invalid pub key error. Even after going back and forth with the manufacturer for over a month, troubleshooting the connection to Azure IoT Hub to test DICE is ongoing.

3.2.6. Vulnerability Testing of Control Applications

After much back and forth with the manufacturer, the CEC1x02 Dev Board was still unable to connect to Azure IoT Hub for DICE Testing nor was it able to be modified to emit a DHCP option header to test Cisco MUD. There is no vulnerability testing that occurred at this time of this device for these reasons.

3.3. Findings Discussion

After a few months of work, it was not as easy as anticipated to secure two IoT devices using the two proposed protocols, DICE and MUD. One of the devices, CloudJAM, came with DICE working almost out of the box, as after following a relatively simple tutorial, the device successfully connected to the Azure IoT hub using DICE. However, this device didn't implement MUD yet due to not finding where in the source code of the firmware to modify the DHCP header to contain the MUD URL. The manufacturer was also unable to provide this information. It appears that a newer version of DHCP client software is required on the firmware as the current version does not support option 161 in the DHCP header, which the MUD URL requires. The same issue applies for the CEC1702 device regarding modifying the DHCP header. The manufacturer is communicating with me as DICE is still being tested to work with the device. Among the information conveyed is that Azure IoT hub is updated since they last got the device to work with it, so they recommended newer firmware source code. However, as of now, this source code produces compilation errors.

4. Recommendations and Implications for Future Research

After getting DICE and MUD to work with both devices, it is recommended that organizations use the testing in Section 3.1.6 and conduct similar tests for the CEC1702 device. It is also recommended that standards and best practices in the industry promote manufacturers to help in supporting these protocols, primarily by upgrading the DHCP clients in the IoT devices to allow emission of DHCP Header option 161 that supports the MUD URL.

5. Conclusion

Security in IoT is still very new. Even though DICE and MUD seem like effective proposals to help solve the security problem in IoT, there are very few devices that support DICE in the market. MUD intends to work for all IoT devices; however, the implementation of it is difficult. Manufacturers should help guide customers to upgrade the DHCP client in IoT devices to implement MUD. Setting up the MUD manager is also difficult and costly. It would cost at least $1500 to buy the Cisco Catalyst switch and ISE server. The open source route of setting up osMUD is free but only supports DHCP currently. In conclusion, I would recommend that researchers and manufacturers develop and produce more guides detailing practical steps in implementing DICE and MUD in various use cases to help consumers making DICE and MUD work in securing their IoT devices.

References

Acar, A., Fereidooni, H., Abera, T., Sikder, A.K., Miettinen, M., Aksu, H., Conti, M., Sadeghi, A., & Uluagac, A.S. (2018). Peek-a-Boo: I see your smart home activities, even encrypted! CoRR, abs/1808.02741.

Azure IoT SDK - CSharp. Retrieved April 30, 2019, from https://github.com/Azure/azure-iot-sdk-csharp

Bruhadeshwar, B., Bachani, M., Peterson, J., Shirazi, H., Ray, I., & Ray, I. (2018). IoTSense: Behavioral Fingerprinting of IoT Devices. CoRR, abs/1804.03852.

CEC1702 IoT Development Kit. Retrieved February 3, 2019, from https://www.microchip.com/DevelopmentTools/ProductDetails/dm990013-bndl

Cloud JAM. Retrieved February 3, 2019, from https://catalog.azureiotsolutions.com/details?title=Cloud_JAM&source=all-devices-page

Elliptic Semiconductor Inc., An Overview of Secret Key and Identity Management for System-on-Chip Architects. (Apr 07, 2015). Retrieved March 27, 2019, from https://www.design-reuse.com/articles/15407/an-overview-of-secret-key-and-identity-management-for-system-on-chip-architects.html

Hallman, Roger & Bryan, Josiah & Palavicini Jr, Geancarlo & Divita, Joseph & Romero-Mariona, Jose. (2017). IoDDoS — The Internet of Distributed Denial of Service Attacks: A Case Study of the Mirai Malware and IoT-Based Botnets. 10.5220/0006246600470058.

Heiland, Deral & Sevier, Nathan & Littlebury, Chris. IoT Security Testing Methodology. (2017, May 10). Retrieved January 30, 2019, from https://blog.rapid7.com/2017/05/10/iot-testing-methodology/. Video available at https://www.youtube.com/watch?v=Vg4NALUMsAs .

Muhammed Zahid Ayar, Muhammed.Ayar@student.sans.edu

Kim, J.-J & Hong, S.-P. (2015). A device identification method in the internet of things (IoT) environments. 10. 33614-33616.

Microsoft, DICE: Device Identifier Composition Engine. (Jan 01, 2015). Retrieved March 25, 2019, from https://www.microsoft.com/en-us/research/project/dice-device-identifier-composition-engine/

Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N.O., Guarnizo, J.D., & Elovici, Y. (2017). Detection of Unauthorized IoT Devices Using Machine Learning Techniques. CoRR, abs/1709.04647.

Miettinen, Markus et al. "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT." 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (2017): n. pag. Crossref. Web.

Why MUD?. Retrieved April 22, 2019, from https://developer.cisco.com/docs/mud/.

Wong, William G., A Roundtable Q&A on the Device Identity Composition Engine (DICE). (Mar 09, 2018). Retrieved March 25, 2019, from https://www.electronicdesign.com/embedded-revolution/roundtable-qa-device-identity-composition-engine-dice

Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis. In Proceedings of the Symposium on Applied Computing (SAC '17). ACM, New York, NY, USA, 506-509. DOI: https://doi.org/10.1145/3019612.3019878

Zelonis, Josh. Top Cybersecurity Threats In 2019. Landscape: The Zero Trust Cybersecurity Playbook. Forrester. December 10, 2018

Muhammed Zahid Ayar, Muhammed.Ayar@student.sans.edu

Appendix 1
Setting up CloudJAM for DICE

1- For setting up CloudJAM to use DICE, firmware modification is needed to support DICE. For that, modify the source code of the firmware with an IDE such as AC6 System Workbench for STM32 to do coding. Download this IDE from this link:

http://www.openstm32.org/System+Workbench+for+STM32

2- When CloudJAM is running, viewing its output requires running a serial terminal such as TeraTerm to view the UART output. Download this software from this link:
https://ttssh2.osdn.jp/

3- Then set up the Azure IoT hub to which the CloudJAM connects using DICE by following these steps:

https://catalog.azureiotsolutions.com/docs?title=Azure/azure-iot-device-ecosystem/setup_iothub

4- Provision (connect) the CloudJAM device to the Azure IoT hub using Device Explorer as mentioned in the steps in this link:
https://catalog.azureiotsolutions.com/docs?title=Azure/azure-iot-device-ecosystem/manage_iot_hub

5- Before connecting to the Azure IoT hub, build and compile the firmware code and write it to the CloudJAM device as in step 3.1 on this page (Getting started tab):

https://catalog.azureiotsolutions.com/details?title=Cloud_JAM&source=null

Appendix 2
Setting up osMUD on OpenWRT for MUD testing

1- Setup openWRT on a virtual machine. This VM acts in place of a physical router. The steps in the link below help in setting up a VM that runs OpenWRT:

https://openwrt.org/docs/guide-user/virtualization/virtualbox-vm

2- Once the OpenWRT router VM is set up, install the osMUD manager onto it. Prepare an OpenWRT docker container image in which to build osMUD using the steps in this guide:

https://github.com/osmud/osmud/blob/master/BuildAndInstall.md

3- Once the docker container image is ready from the previous step, build the osMUD image in the docker container and get the osMUD IPK file, using the steps in this link:

https://github.com/osmud/osmud#build-osmud-for-openwrt

4- After obtaining the osMUD image file, install osMUD onto the router with the following steps:

https://github.com/osmud/osmud#install-osmud-on-the-router

Appendix 3
Setting up PAU06 Wi-Fi Adapter for monitor mode in Kali Linux

1- First, see the associated physical interface for the wireless card with the command:

```
ls /sys/class/ieee80211
```

This command shows something like ‘phy0’ or phy and another number depending on how many times the adapter was unplugged/replugged into the machine.

2- Delete the automatically added child interface wlan0 to add a child interface in monitor mode:

```
iw dev wlan0 del
```

iw is the command used for wireless configuration.

3- Add the child interface in monitor mode with the following command (with the appropriate phy<number>:

```
iw phy phy0 interface add wlan0mon type monitor
```

Check the information regarding the newly added interface with the following command:

```
iw dev wlan0mon info
```

This command shows the type of interface listing as monitor.

4- Set the new child interface into the ‘UP’ state using ifconfig:

```
ifconfig wlan0mon up
```

5- Then to be able to capture not only management frames but also high-throughput data frames, widen the channel bandwidth with the following command:

```
iw dev wlan0mon set channel 1 HT40+
```

Now the wlan0mon child interface is ready to for capturing wireless packets with Wireshark or other packet capturing tool.