Winter 1-2021

HARDWARE HACKING, PURPLEAIR – RAST STUDIMI

Pajtim Cakaj
University for Business and Technology - UBT

Programi për Shkenca Kompjuterike dhe Inxhinierise

HARDWARE HACKING, PURPLEAIR – RAST STUDIMI

Shkalla Bachelor

Pajtim Cakaj

Janar / 2021
Prishtinë

Programi Shkenca Kompjuterike dhe Inxhinierise

Punim Diplome
Viti akademik 2014 – 2015

Pajtim Cakaj

HARDWARE HACKING, PURPLEAIR – RAST STUDIMI

Mentori: PhD. Blerton Abazi

Janar / 2021

Ky punim është përpiluar dhe dorëzuar në përmbushjen e kërkesave të pjesshme
për Shkallën Bachelor

ABSTRAKT

Gajtë këtij punimi unë do të trajtoj temën Hardware Hacking, ma konkretisht hakimi i paisjes PurpleAir. Do të analizojë sistematikisht mekanizmat e përditësimit të firmware-it të pajisjeve të njohura të monitorimit të cilësisë së ajrit PurpleAir dhe paraqes sulme (fizike) harduerike dhe sulme nga distanca kundër këtyre pajisjeve, cilat janë dobësit që një haker i shfrytëzon për të hakuar këto paisje, cilat janë pasojat e hakimit të këtyre paisjeje si dhe do të tregohet projektimi dhe implementimi për një sistem të sigurt të përditësimit të firmware-it të **k**ëtyre paisjeve e që si rast studimi është marr paisja PurpleAir.

Fjalët Kyqe: Hardware, Hacking, PurpleAir, firmware-it.

MIRËNJOHJE/FALENDERIME

Së pari, falenderimi i takon Zotit të Madhëruar ndaj të cilit përulem me krenari.
Një falenderim dhe mirënjohje shkon për familjen time, për kurajën dhe mbështetjen që më kanë ofruar.
Një falenderim tjetër shkon për krejt stafin e UBT-së, e në veçanti për PhD. Blerton Abazi për mbështetjen që më ofroi përgjatë gjithë punës sime në finalizimin e këtij punimi të diplomës.
Së fundmi, falenderoj shoqërinë time për këshillat dhe përkrahjen e vazhdueshme, jam me fat që janë pjesë e jetës sime.

PËRMBAJTJA

LISTA E FIGURAVE

LISTA E TABELAVE

LISTA E SHKURTESAVE

MCU …….....Microcontroller Unit

UART.....Universal asynchronous receiver-transmitter

Micro-USB.....Micro Universal Serial Bus

MB.....Megabyte

OTA.....Over-The-Air

SPI .....Serial Peripheral Interface

I / O .....Input /Output

CMOS .....Complementary Metal Oxide Semiconductor

JTAG.....Joint Test Action Group

SPM .....Store Program Memory

HVPP.....High Voltage Parallel Programming

IoT.....Internet of Things

IC.....Integrated circuit

CPU…….....……...Central processing unit

EEPROM.....Electrically Erasable Programmable Read-Only Memory

ASCII.....American Standard Code for Information Interchange

IT.....Information Technology

AES.....Advanced Encryption Standard

LED.....Light-emitting diode

I2C.....Inter-Integrated Circuit

CAN.....Controller Area Network

RS232.....Computer serial interface, IEEE

HP200A..... Hewlett-Packard Audio Oscillator

Wi-Fi……...Wireless Fidelity

1 HYRJE NË HARDWARE HACKING

Fjala "Hacking" i përket hardverit kryesisht viktimizohet në disa mënyra. Në përkufizimin e pranuar zakonisht, “Hardware Hacking” nënkupton modifikimin e një pjese të pajisjeve elektronike ekzistuese për ta përdorur atë në një mënyrë që nuk është menduar domosdoshmërisht. Hardware Hacking sot ka dalë në tregun e zakonshëm si kurrë më parë. Pothuajse çdo pjesë e pajisjeve mund të shërbejë si një kandidat për tu hakuar.

Kreativiteti dhe vendosmëria mund t'ju çojnë shumë larg sesa shumica e zhvilluesve të produkteve mund të imagjinojnë. Hardware Hacking është zakonisht një përpjekje individuale, si krijimi i një arti[1].Sidoqoftë, ashtu si artistët, hakerat ndonjëherë bashkëpunojnë dhe formojnë bashkësi njerëzish që punojnë drejt një qëllimi të ngjashëm.

Filozofia e shumicës së hakerave të pajisjeve është e drejtpërdrejtë:

- Bëni diçka me një pjesë të pajisjes që nuk është bërë kurrë më parë.
- Krijoni diçka të jashtëzakonshme.
- Mos dëmtoni askënd në proces.

Hakerimi i harduerëve pa dyshim daton pothuajse 200 vjet më parë.Charles Babbage krijoi motorin e tij të ndryshimit në fillimin e viteve 1800 - një formë mekanike e hakimit të pajisjeve. William Crookes zbuloi elektronin në mes të viteve 1800 - ndoshta forma e parë e hakimit të pajisjeve elektronike.

Përgjatë zhvillimit të telegrafisë pa tela, tubave të vakumit, radios, televizionit dhe tranzistorëve, ka pasur hakerë harduerësh për të përmendur disa si Benjamin Franklin, Thomas Edison dhe Alexander Graham Bell.

Me zhvillimin dhe lëshimin e mikroprocesorit të parë (Intel 4004) në nëntor 1971, publiku i gjerë më në fund mori një shije të informatikës. Potenciali për piraterinë e pajisjeve është rritur jashtëzakonisht shumë në dekadën e kaluar pasi kompjuterat dhe teknologjia janë bërë më të ndërthurura me rrjedhën e zakonshme dhe jetën e përditshme. Në 1938 dy të rinjë Bill Hewlett dhe Dejv Packard bëhen bashkë dhe shpikin, në garazhin e tyre pajisjen për testimin e audios, oshilatorin audio me kapacitet rezistence HP200A. Hewlett dhe Packard vazhduan më tej për të themeluar kompaninë që ne njohim sot.

Për më tepër, natyra e motivuar në mënyrë të pavarur e hakerave do të thotë që mijëra ide janë

testuar dhe ndërtuar nga hakerat në mungesë të kapitalit sipërmarrës ose kufizimeve të rrezikut të investitorëve. Hakerat luajnë një rol të rëndësishëm në rritjen e teknologjisë, Në fakt, disa prej projekteve që hakerat punuan në garazhet e tyre vazhduan të formonin rrënjët e teknologjisë së sotme [1].

2 RËNDËSIA E HARDWARE HACKING

Aftësitë e hakimit të harduerit janë një grup aftësish në zhvillim që mund të hapin një mundësi të mirë për karrierën e dikujt.

Më poshtë janë tiparet e rëndësishme të Hardware Hacking [2]:

- Përvoja praktike: kjo është karakteristika më e rëndësishme që i shton bukurinë hakimit të pajisjeve. Ky është ndryshimi kryesor midis hakimit të Hardware-it dhe softuerit.
- Mundësitë e pafundme: ne jemi të rrethuar nga pajisje të pafundme harduerike
  që mundë ti hakojmë, gjithësesi vetëm nëse kemi leje ta bëjmë këtë.
  Ekziston një numër i gjërave në të cilat ne mund të hakojmë si p.sh. në llamba LED, CPU të një kompjuteri, etj.
- Më e prekshme: ju mund ta ndjeni ndryshimin fizikisht të paisjes sapo të hakoni atë. Duhet të jemi mjaft të kujdesshëm në bërjen e gjërave të tilla pasi mund të vrasin jetën tënde.
- Forenzika digjitale: Do të ketë nevojë për harduer haking kur një pajisje shkatërrohet pjesërisht, ku forenzika do të duhet të ndërhyjë direkt me disa përbërës për të rimarrë provat.
- Testimi: Sapo të hakohet paisja, ne mund t'i provojmë manualisht nëse ato janë duke punuar sipas kërkesave të kërkuara.
- Aftësitë e Programimit: Hakerët e Hardware-it kanë aftësi programimi të ngjashme me Hakerët e softuerit. Njohuritë e tyre për Programimin e programeve janë më shumë të një niveli të ulët. Ky kuptim themelor i softuerit përdoret mjaft mirë gjatë piraterisë së Harduerit.

3 METODAT E HARDWARE HACKING

Fjala "pirateri" pasi i përket harduerit shpesh keqpërdoret. Në përkufizimin e pranuar zakonisht, "pirateria e pajisjeve" nënkupton modifikimin e një pjese të pajisjeve elektronike ekzistuese për ta përdorur atë në një mënyrë që nuk ishte menduar domosdoshmërisht. Edhe ky përkufizim është i paqartë, pasi mund t'i referohet çdo metode të modifikimit të pajisjeve, qofshin ato të mbylljes, elektronikës ose sjelljes. Modifikimi i një mbylljeje të një pajisjeje është zakonisht i drejtpërdrejtë; hapni një vrimë, prisni një vend të caktuar, etj. Por hakimi i elektronikës dhe / ose sjelljes është një temë e ndërlikuar në vetvete. Kur kërkoni të modifikoni një pajisje, nganjëherë është e vështirë të dihet se ku të fillohet dhe çfarë këndi sulmi duhet të merret kur hakoni diçka për një qëllim për të cilin nuk ishte krijuar.

Nëse po kërkoni të hakoni një pjesë të pajisjes, mënyra se si i qaseni hakimit varet nga ajo që po përpiqeni të bëni. Po përpiqeni ta bëni pa tel? A po përpiqeni të ndryshoni atë që shfaq?

Ajo që vijon janë disa metoda të zakonshme të piraterisë së pajisjeve.

Llojet e interfejsave te zakonshëvm: UART, SPI, I2C, JTAG, CAN, RS232, etj.

3.1 Metoda 1: Patching në I / O

Metoda e parë (dhe pa dyshim më e lehtë) e piraterisë së një pajisje është duke u përshtatur në mekanizmin e saj të kontrollit. Shumica e produkteve të konsumit kanë të paktën një buton ose LED tregues, dhe lidhjet për atë përbërës janë zakonisht të lehta për tu gjetur dhe për tu bashkuar.Për shembull, nëse dëshironi ta bëni një pajisje pa tel, mund ta lidhni pajisjen tuaj pa tel direkt me butonin për të drejtuar sinjalin e butoni lartë ose i ulët në varësi të asaj që pajisja pa tel e pranon. Ky lloj sulmi mund të realizohet pa gërmuar në programimin aktual të pajisjes. [3]

3.2 Portet Seriale të Ekspozuara (UART)

Ndërfaqja UART është një pajisje hardware (qark fizik në kontrollues ose një IC i pavarur) që përdoret për komunikim serik asinkron. Mundëson përkthimin e të dhënave ndërmjet ndërfaqeve serike dhe paralele duke përdorur një regjistër ndërrimi.

Më së shpeshti përdoret në pajisjet e ngulitura. Komunikimi zhvillohet drejtpërdrejt midis dy UART-ve. Ndërfaqja UART në anën transmetuese merr të dhënat paralele dhe i fsheh ato në formë serike. Më pas i transmeton të dhënat serikisht në ndërfaqen UART nga ana marrëse. Ndërfaqja UART marrëse merr të dhënat serike dhe kthehet përsëri në paralele për t'i dhënë ato pajisjes

kontrolluese të palës marrëse[3]. Komunikon duke përdorur dy linja sinjali RX (marrës) dhe TX (transmetues). Të dhënat nga pini TX në ndërfaqen transmetuese UART merren në pinin RX në ndërfaqen marrëse UART. Për të sulmuar pajisjen përmes portit UART, së pari duhet të identifikojmë pinat UART RX, TX në pajisje.

3.3 JTAG

JTAG është një ndërfaqe fizike harduerike që bën të mundur, ndër të tjera, nxjerrjen e imazhit të firmuerit nga pajisjet elektronike.Kur prodhohet një pajisje elektronike, ajo duhet të programohet me firmware në një moment. E njëjta portë përmes së cilës programohet një pajisje mund të përdoret gjithashtu për të çmontuar dhe hackuar firmware. Shumë mikrokontrollues kanë një veçori të shkarkimit të kujtesës që mund të shkaktohet përmes portës së tij të programimit që lejon një përdorues të lexojë kujtesën e plotë (në hex.) të çipit. Shumë pajisje përfshijnë një veçori që "bllokon" pajisjen në mënyrë që ajo të mos mund të lexohet ose riprogramohet sapo të ndizet, por shumë prodhues të pajisjeve nuk e zbatojnë këtë veçori, duke i lënë produktet e tyre të ndjeshme ndaj piraterisë së firmware [3].

Në mënyrë që të hakoni firmware përmes një porti programues, duhet:

- Identifikoni pajisjen dhe nëse ajo ka aftësinë për të hedhur kujtesën e saj
- Ndërtoni ose blini një programues që mund të marrë këtë memorie dhe ta transmetojë në një kompjuter
- Merrni hedhjen hex, nga çipi me programuesin
- Çmontoni hex. në gjuhen asamblesë

Sapo hakeri të ketë gjuhën e asamblesë, ai shikon për firmware. Nga atje, dikush mund të modifikojë skedarin e firmware, duke ndryshuar variablat dhe regjistrat për të ndryshuar sjelljen e pajisjes. Pastaj hakeri rikompilon firmware-in në hex dhe riprogramon pajisjen me firmware-in e hakuar. Kjo është një metodë e përparuar e piraterisë së pajisjeve, por mund të sigurojë rezultatet më efektive.

3.4 Nxjerrja e kujtesës / Memory Extraction

Shumë pajisje kanë qip memorues në pllakë nga i cili mund të jeni në gjendje të nxjerrni përmbajtjen. Kjo nuk do të thotë domosdoshmërisht se ato përmbajtje do të mosshifrohen - kështu që mund të ketë punë shtesë për të bërë, siç është nxjerrja e çelësit të enkriptimit.

Sidoqoftë, nganjëherë mund të përdorni çipin e ruajtjes, të ndërveproni me të për të hedhur përmbajtjen e tij dhe më pas të jeni në gjendje të shikoni softuerin që sistemi ekzekuton - madje mund të jeni në gjendje ta modifikoni atë drejtpërdrejt - duke ju lejuar të ndryshoni çelësin e hyrjes, të ekspozoni shërbime të tilla si telnet / ssh, ose çaktivizoni firewall-et siç janë iptables në sistemet e ngulitura Linux. [4]

3.5 The Logic Analyzer

Një analizues logjik është një instrument elektronik që kap dhe shfaq sinjale të shumëfishta nga një sistem dixhital ose qark dixhital. Një analizues logjik mund të shndërrojë të dhënat e kapura në diagrame kohore, dekodime protokolli, gjurmë makinerie shtetërore, gjuhë montimi ose mund të ndërlidhë montimin me softuerin e nivelit burimor.Analizuesi logjik do të regjistrojë çdo sinjal që ndodh në linjat që nuhat, dhe se të dhënat potencialisht mund të përkthehen në diçka të dobishme.Bene përkthimin e sinjalit për protokollet SPI, I2C dhe serial. Për shkak të kësaj, ai është një mjet jetësor për piraterinë së pajisjeve[3].

3.6 Zëvendësimi i një Komponenti

Kjo metodë përdoret shpesh në Bending Circuit. Përdoruesi dëshiron që pajisja të tingëllojë ndryshe, kështu që ai zëvendëson një komponent (zakonisht eksperimentalisht) për të marrë një tingull tjetër nga një pajisje. Shumë hakerime interesante janë arritur duke zëvendësuar një komponent. Për shembull, zëvendësimi i llambave të biçikletës me LED me intensitet të lartë, ose zëvendësimi i motorëve në një makinë lodër për ta bërë atë të ngasë me rrezikshmëri të shpejtë.

4 RAST STUDIMI - PURPLEAIR

Në këtë pjesë, së pari japim një prezantim të shkurtër rreth paisjes për monitorimin e cilësisë së ajrit PurpleAir dhe më pas analizën e firmware, duke zbuluar paraqitjen e saj.
Kontributet kryesore gjatë këtij studimi përmblidhen si më poshtë:
Ne analizojmë sistematikisht mekanizmat e përditësimit të firmware-it të pajisjeve të njohura të monitorimit të cilësisë së ajrit PurpleAir dhe paraqesim sulme (fizike) harduerike dhe sulme nga distanca kundër këtyre pajisjeve. Ne jemi të parët që fillojmë një shfrytëzim firmware kundër pajisjeve të monitorimit të cilësisë së ajrit.

Ne projektojmë dhe implementojmë një sistem të sigurt të përditësimit të firmware-it për një trashëgimi të njohur MCU ATmega1284P sipas kërkesave nga MITER Cyber Academy, duke adresuar sulmet që mund të ndodhin gjatë përditësimit të firmware-it. Ne ofrojmë udhëzime se si të dizajnojmë dhe zbatojmë sigurinë e përditësimit të firmware-it për një trashëgimi të tillë të MCU-ve.

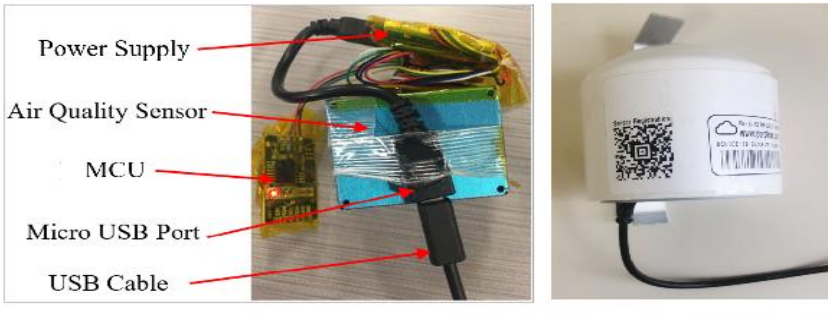
Ne paraqesim një larmi të grackave të mundshme të sistemit të përditësimit të sigurt të firmware-it. Qëllimi ynë është të ndihmojmë shitësit për të shmangur këto gracka në zbatimin e tyre.Më në fund, ne diskutojmë mbi shfrytëzimet e mundshme kundër pajisjes.

4.1 Pajisjet e Monitorimit të Cilësisë së Ajrit PurpleAir

Fig. 1 (a) tregon pajisjen tregtare Purpleair pa këllëf, dhe Fig. 1 (b) tregon përbërësit e tij të brendshëm, duke përfshirë dy sensorë të cilësisë së ajrit PMS5003, një MCU dhe një bord qark të furnizimit me energji elektrike. MCU është një kompjuter i vogël në një çip të vetëm i krijuar për sisteme të ngulitura. Krahasuar me çipa të tjerë të programueshëm, MCU është një kompjuter i vogël në një çip të vetëm i krijuar për sisteme të ngulitura. Karakterizohen me konsum të ulët të energjisë, çmim më të ulët, madhësi të vogël, por llogaritje të kufizuar dhe burime të kujtesës. MCU mund të zgjerohet me module shtesë funksionale siç janë flashi i jashtëm dhe sensorë të ndryshëm. Këto karakteristika çojnë në një adoptim të gjerë të MCU-ve për aplikime të lehta IoT siç janë rrjetet e sensorit të cilësisë së ajrit.Një çip MCU shpesh përbëhet nga një njësi qendrore e përpunimit (CPU), ora e sistemit, memoria dhe pajisjet periferike [5]. Firmware, i quajtur ndryshe imazhi i aplikacionit, mund të digjet në memorien (e brendshme ose të jashtme) të MCU dhe të

ekzekutohet. Për shkak të burimeve të kufizuara dhe arkitekturës relativisht të thjeshtë, MCU-të shpesh i dedikohen një ose më shumë detyrave të thjeshta në vend që të përpunojnë shumë detyra komplekse njëkohësisht.

Sensori i cilësisë së ajrit mate masën e ambientit (p.sh. përqendrimet e numrit të grimcave të PM2.5, lagështia) dhe është i lidhur me MCU përmes UART. MCU është një çip ESP8266 [6] lexon matjet nga sensori i cilësisë së ajrit përmes UART. ESP8266 përmban funksione të ndryshme të tilla si lidhja WiFi dhe portet periferike.Pllaka e qarkut të furnizimit me energji elektrike përdoret për të furnizuar të gjithë përbërësit. Kolektivisht, këto përbërës mund të monitorojnë metrika të shumta mjedisore dhe të raportojnë matjet në serverin cloud. PurpleAir ofron një mekanizëm OTA, që pajisjet mund të përdorin për të shkarkuar pa tel një firmware të sapo lëshuar nga serverat e largët dhe për të riprogramuar vetë. Duke pasur parasysh shkallën e madhe të mundshme të një sistemi IoT, mekanizmi OTA është i përshtatshëm për përditësimin e firmware-it. Përditësimin përmes OTA-s së sigurt është kritik për përmirësimin e jetës së këtyre pajisjeve inteligjente [13]. Jo vetëm që OTA rrit funksionalitetin dhe shkallëzimin e pajisjeve, por dobësitë e sigurisë mund të rregullohen edhe pasi pajisjet të jenë nisur. Kostot për mirëmbajtjen e aplikacioneve të IoT zvogëlohen kryesisht duke u përditësuar automatikisht në distancë. Sidoqoftë, dobësitë e rrjetit mund të ekzistojnë në procesin e shkarkimit të të dhënave nëse protokolli i rrjetit, p.sh., WiFi, nuk është i siguruar. Për shembull, një detyrë OTA e ndërtuar në krye të Protokollit të Transferimit të Hypertext (HTTP) dërgon firmware në pajisjet IoT pa një proces të vërtetimit të firmware ose mekanizëm të kriptimit të të dhënave. Malware mund të futet në pajisjen IoT përmes procesit OTA. Prandaj, për të siguruar shkallëzimin dhe sigurinë e pajisjeve IoT, duhet të implementohet një mekanizëm i sigurt OTA.



(a) PurpleAir Sensor Without Case (b) PurpleAir Sensor

Fig. 1 – PurpleAir

4.2 Aksesi në firmware

Tani paraqesim sulmin e pajisjes për të hyrë në flashin e pajisjes, e cila përdor ESP8266 MCU. Pajisja Purpleair vjen me një kabllo Micro-USB "vetëm për karikim" pa tela të dhënash. Sidoqoftë, një kabëll gjenerik Micro-USB mundëson funksionalitetin e programimit përmes kabllit. ESP8266 ofron një port universal - marrës-transmetues asinkron (UART) për transmetimin e të dhënave, p.sh. programimi. Për të mundësuar komunikimin UART, ne e lidhim pajisjen me një kompjuter korrigjues përmes një kabllo Micro-USB me një normë baud prej 115200. Ne përdorim një mjet të bazuar në Python esptool në kompjuterin tonë të testimit për të marrë të dhënat e dërguara nga pajisja.Esptool [7] është krijuar për të komunikuar me bootloader ROM të Espressif Systems [8]. Esptool ofron funksionalitete të ndryshme, ne rendisim disa nga këto funksionalitete që janë të lidhura ngushtë me kërkimin tonë.

- “esptool.py -port PORT flash id”. Siç tregohet në Fig. 2, kjo komandë mund të lexojë informacionin bazë të një firmware, të tilla si adresa MAC e pajisjes, madhësinë e flash memories dhe informacionin e prodhuesit. Argumenti PORT i referohet numrit të portës UART.
- “esptool.py -port PORT -b 115200 read flash 00x200000 flash contents.bin”. Siç tregohet në Fig. 3, kjo komand lexon përmbajtjen e flash memories nga çipi.Në rastin tonë, esptool lexon 0x200000 (≈2M) bajt duke filluar nga adresën 0 të memorjes flash dhe e ruan atë në atë diskun lokal me një emër fajlli "përmbajtje flash.bin".
- “esptool.py erase flash”. Kjo komandë fshin të gjitha bajtat e flash dhe do të zëvendësohen nga bajte të pakuptimta "0xFF". Në mënyrë të ngjashme, “erase region” fshin një seksion specifik të të dhënave të flash memorjes, me parametrat e specifikuar duke përfshirë adresën fillestare dhe madhësinë e flash memories siç tregohet në komandat e mëparshme.
- “esptool.py -p PORT write flash 0x1000 my app.bin”. Kjo komandë shkruan një fajll binar (my app.bin in our one boot.bin) në qip. Parametrat janë të ngjashëm me komandat e mëparshme.

```
PS C:\Users\chao> esptool.py --port COM4 flash_id
esptool.py v2.6
Serial port COM4
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
MAC: 5c:cf:7f:5c:9f:c4
Uploading stub...
Running stub...
Stub running...
Manufacturer: c8
Device: 4015
Detected flash size: 2MB
Hard resetting via RTS pin...
PS C:\Users\chao>
```

Fig. 2 - Leximi i informacionit bazë të ESP8266

```
PS C:\Users\chao> esptool.py --port COM4 -b 115200 read_flash
 0 0x200000 flash_content.bin
esptool.py v2.6
Serial port COM4
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
MAC: 5c:cf:7f:5c:9f:c4
Uploading stub...
Running stub...
Stub running...
2097152 (100 %)
2097152 (100 %)
Read 2097152 bytes at 0x0 in 188.8 seconds (88.8 kbit/s)...
Hard resetting via RTS pin...
```

Fig. 3 - Leximi i përmbajtjes së flashit nga ESP8266.

4.3 PurpleAir Flash Map

Bazuar në analizën e mësipërme, ne tani paraqesim hartën e flashit nga pajisja PurpleAir. Madhësia e flashit është 2 MB, ndërsa 1 MB e fundit janë të dhëna të pakuptimta të mbushjes.

ESP8266 - Read Only-Memory (ROM) përmban disa kode të bibliotekës dhe një ngarkues të nisjes të fazës së parë. Pajisja [9], [10] PurpleAir mbështet OTA dhe flashi i saj është i ndarë në katër seksione, duke përfshirë dy imazhe të programit, një rajon të memorjes vetëm të lexueshme të programueshme të fshirjes elektrike (EEPROM) dhe një seksion të paracaktuar të të dhënave. Siç tregohet në Fig. 4, secili imazh i programit përmban dy seksione, të shënuara si boot.bin dhe

user.bin. boot.bin përdoret për të ruajtur bootloader-in. user.bin përdoret për të ruajtur kodin e aplikacionit. Të dy imazhet e programit mund të jenë identike. EEPROM është një lloj i kujtesës jo të paqëndrueshme që siguron ruajtjen e vazhdueshme të të dhënave në reboots. Në rastin e PurpleAir, WiFi SSID dhe fjalëkalimi janë ruajtur në këtë rajon. Më në fund, seksioni i parazgjedhur i të dhënave përmban imazhet e fajllit esp init data default.bin dhe blank.bin, të cilat ruajnë parametrat e parazgjedhur të sistemit, siç janë konfigurimet Wi-Fi, përveç SSID dhe fjalëkalimi. Harta flash e pajisjes PurpleAir është e personalizuar dhe e ndryshme nga zbatimi zyrtar OTA i ESP8266 [11].

Për shembull, zbatimi zyrtar i OTA-s ka vetëm një boot.bin, i cili ruan adresën e imazhit aktiv të programit dhe ekzekuton atë program gjatë rinisjes.

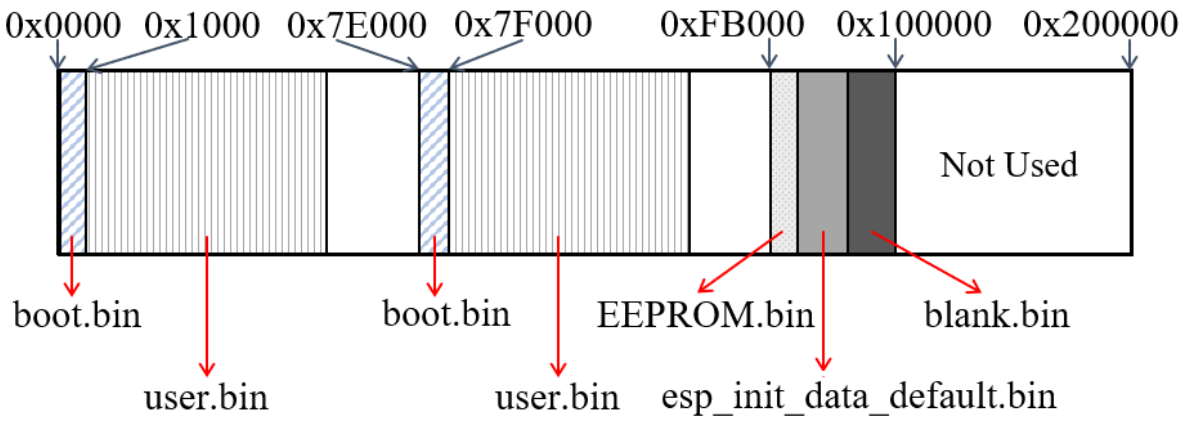


Fig. 4 - Harta e Flash-it të Firmware PurpleAir

4.4 Shfrytëzimet dhe sulmet

Qëllimi përfundimtar i sulmit tonë është të ndryshojmë firmware-in e pajisjes. Për këtë qëllim, do të paraqiten dy lloje të sulmeve, përkatësisht sulmi fizik dhe sulmi në distancë. Në sulmin fizik, ne supozojmë se kundërshtari mund të hyjë fizikisht në pajisje. Ky supozim është i arsyeshëm pasi që pajisja e cilësisë së ajrit mund të vendoset jashtë dhe e lirë për t'u përdorur. Për më tepër, pajisja mund të mbetet e pa monitoruar për një periudhë të zgjatur kohe. Sidoqoftë, në sulmin në distancë, ne supozojmë se hyrja fizike është e padisponueshme për kundërshtarin.

Sulmi fizik: Një kundërshtar mund të vendosë sulmin fizik duke lidhur MCU me një kompjuter përmes një kabllo MicroUSB. Ne rendisim disa shfrytëzime të mundshme që mund të vijnë nga

ky sulm për të demonstruar pasojat e tij të rënda: **(i)** Flashing një firmware me qellim të keq.Një sulmues mund të programojë një firmware të dëmshëm dhe ta vendos atë në qip. Firmware i modifikuar mund të injektojë të dhëna të fabrikuara në rrjetin e sensorit të cilësisë së ajrit, gjë që mund të keqinformojë publikun. **(ii)** Vjedhja e kredencialeve të Wi-Fi. Për shkak të mungesës së vërtetimit dhe enkriptimit të firmware, përmbajtja e flashit është e lirë për të hyrë. Prandaj mund të nxjerrim kredencialet Wi-Fi nga flashi, gjë që shkakton dëme përtej rrjetit të sensorit të cilësisë së ajrit.

Sulmi në distancë: Paisja PurpleAir ofron një mekanizëm OTA, i cili lejon pajisjen të bëj update firmware-in e saj pas vendosjes. Përmes analizës së trafikut, ne e dimë që kur një pajisje PurpleAir rindizet dhe lidhet me serverin cloud, pajisja dërgon një kërkesë te serveri cloud dhe kërkon versionin aktual të firmware-it. Nëse ekziston një firmware i ri, do të kryhet OTA për të sinkronizuar firmware-in me serverin cloud. Ky mekanizëm siguron që pajisja të jetë e përditësuar(update). Pasi kemi analizuar me kujdes mekanizmin PurpleAir OTA, ne identifikojmë një dobësi të rëndë: pajisja nuk e vërteton serverin dhe serveri nuk e vërteton pajisjen. Kjo është, nuk ka vërtetim reciprok midis pajisjes dhe serverit. Për më tepër, komunikimi është në tekst të thjeshtë. Firmware nuk mbrohet me asnjë mekanizëm dhe në tekst të thjeshtë.Mungesa e vërtetimit reciprok ngre çështje të sigurisë. Siç tregohet në Fig. 5, një sulmues ose mund të pretendojë të jetë serveri cloud ose një pajisje mashtruese për të shkaktuar ligësi: **(i)** Nëse sulmuesi është në gjendje të imitojë serverin, sulmuesi mund të trillojë në mënyrë të parëndësishme një firmware të dëmshëm dhe ta dërgojë atë në pajisje, e cila përditëson firmware-in lokal me firmware in e dëmshëm pa asnjë proces verifikimi.

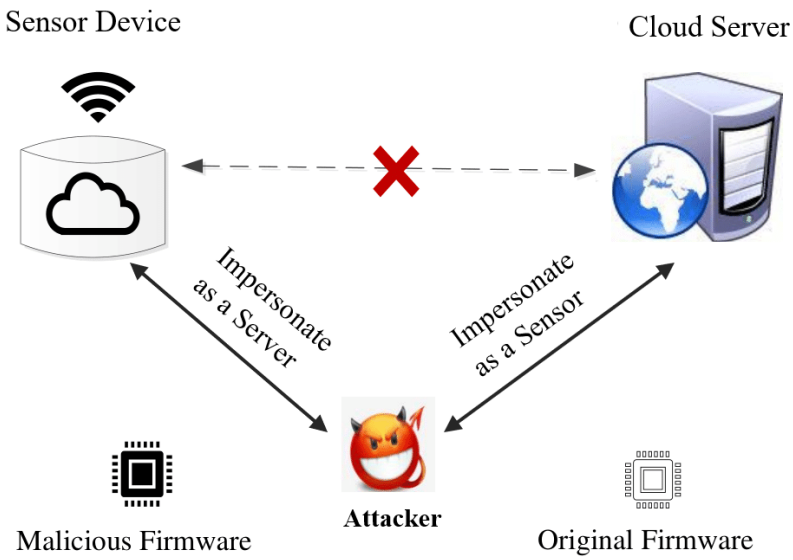


Fig. 5 - Pasqyra e sulmit në distancë

Firmware me qëllim të keq do të ekzekutohet pas rindezjes. Këtu është një mënyrë për të bërë rolin e serverit. Sulmuesi mund të bllokojë routerin WiFi me të cilin është lidhur pajisja viktimë. Pas një periudhe dështimesh të lidhjes, pajisja PurpleAir rivendoset automatikisht në modalitetin AP, i cili lejon sulmuesin të konfigurojë pajisjen viktimë, e cila do të lidhet me një router mashtrues WiFi. Routeri mashtrues WiFi mund të luajë rolin e serverit.**(ii)** Kur sulmuesi imiton një pajisje, sulmuesi mund të marrë firmware-in në server duke dërguar një pyetje në serverin cloud, siç tregohet në Fig. 6, ne i kemi konfirmuar këto dy sulme në eksperimentet tona.

Fig. 6 - Paketa e reagimit të përditësimit të PurpleAir me firmware të ri të bashkangjitur

4.5 Sigurimi i përditësimit të firmware-it

Për t'iu kundërvënë sulmeve kundër përditësimit të firmware-it, janë bërë shumë përpjekje [12], [13], [14].Një çështje kryesore e pajisjes PurpleAir është se ajo përdor çipin ESP8266, i cili nuk ka ndonjë mekanizëm të integruar hardverik për sigurinë. Për shembull, nuk ka asnjë mënyrë për të çaktivizuar portin e tij UART. Një port i hapur UART i nënshtrohet sulmit fizik. Një pajisje e sigurt IoT ka nevojë për veçori të sigurisë së pajisjes. Për të mposhtur sulmet në distancë kundër përditësimit të firmware-it, duhet të përdoret vërtetimi i ndërsjellë dhe kriptimi. Në këtë drejtim, MITER Cyber Academy [15], propozon kërkesa për hartimin e një sistemi të sigurt për përditësimin e firmware-it pa lidhje interneti në mënyrë që firmware i sigurt të mund të shkarkohet nga interneti dhe të përditësohet në nivel lokal.

Ne dizenjojmë dhe implementojmë një sistem të tillë të sigurt të përditësimit të firmware-it sipas kërkesave të propozuara dhe diskutojmë për grackat e mundshme.

4.5.1 Përmbledhje e sistemit të sigurt të përditësimit të firmware-it

Për të garantuar një përditësim të sigurt të firmware-it, ne do të përfshijmë kriptimin dhe vërtetimin gjatë procesit të përditësimit. Në mënyrë të veçantë, shitësi do të kriptojë firmware-in përpara se të lëshohet, ndërsa çelësi i dekriptimit duhet të jetë i sigurt brenda secilit produkt. Vetëm produkti mund të deshifrojë firmware-in. Produkti duhet të verifikojë integritetin e firmware-it gjithashtu. Për këtë qëllim, vlera hash e firmware-it të synuar është e koduar dhe e bashkangjitur së bashku me firmware-in e koduar. Më në fund, për të parandaluar që sulmuesi të përdorë sulme fizike dhe të marrë të dhëna të ndjeshme, të tilla si çelësi i deshifrimit ose firmware-in, ndërfaqet e korrigjimit, të tilla si portet JTAG, SPI dhe UART, duhet të çaktivizohen ose të paktën të kenë qasje të kufizuar (p.sh. të mbrojtura nga një fjalëkalim i fortë). Bitet e bllokimit duhet të aktivizohen gjithashtu nëse çipi i produktit e mbështet atë. Në këtë mënyrë, analiza e firmware do të dështojë. Sistemi i sigurt i përditësimit të firmware-it siç tregohet në Fig. 7 përbëhet nga katër komponentë, duke përfshirë një bootloader të sigurt, një mjet për mbrojtjen e firmware, një mjet për përditësimin e firmware-it, dhe një mjet i kthimit të prodhuesit. Një përdorim tipik i sistemit të përditësimit të firmware-it të sigurt funksionon si më poshtë.

Shitësi përdor mjetin e mbrojtjes së firmware-it për të kriptuar firmwerin me një çelës AES dhe gjithashtu përdor bordin e programuesit (AVR Dragon [16]) në Fig. 8 për të hedh bootloader-in e sigurt në seksionin flash të bootloader-it të çip-it përmes ndërfaqes ISP të bordit të programuesit, i cili është i lidhur me një sërë kunjësh të programueshëm I / O në ATmega1284P. Çelësi AES i shitësit është i koduar në bootloader. Gjatë një procesi të përditësimit të firmware-it nga një shitës ose një përdorues, mjeti i përditësimit të firmware-it dërgon firmware të koduar në ngarkuesin e sigurt përmes portës UART. Bootloader deshifron dhe verifikon firmware-in dhe kopjon firmware-un e dekriptuar në seksionin e memorjes flash të aplikacionit. Pajisja më pas rindizet për të ekzekutuar firmware-in e ri. Shitësi mund të përdorë mjetin e kthimit të leximit për të nxjerrë firmware (të dëmtuar) nga çipi për korrigjimin e gabimeve përmes portës UART. Gjatë leximit, bootloader-i enkripton firmware-in dhe shitësi duhet të deshifrojë firmware-in e enkriptuar. Në skenarin e botës reale, mjeti i përditësimit të firmware-it mund të zëvendësohet me mekanizmin e brendshëm OTA. Në këtë rast, bootloader-i merr një firmware-in nga serveri cloud, pastaj e hedh atë në qip. Funksionet e tjera të bootloader dhe tre mjeteve të tjera do të funksionojnë pa ndonjë ndryshim. Sidoqoftë, në disa raste, mjeti i përditësimit të firmware-it mund të jetë i preferueshëm

- për shembull, kur është i ri firmware është shumë i madh për ta shkarkuar pa mbishkruar firmware-in e mëparshëm.

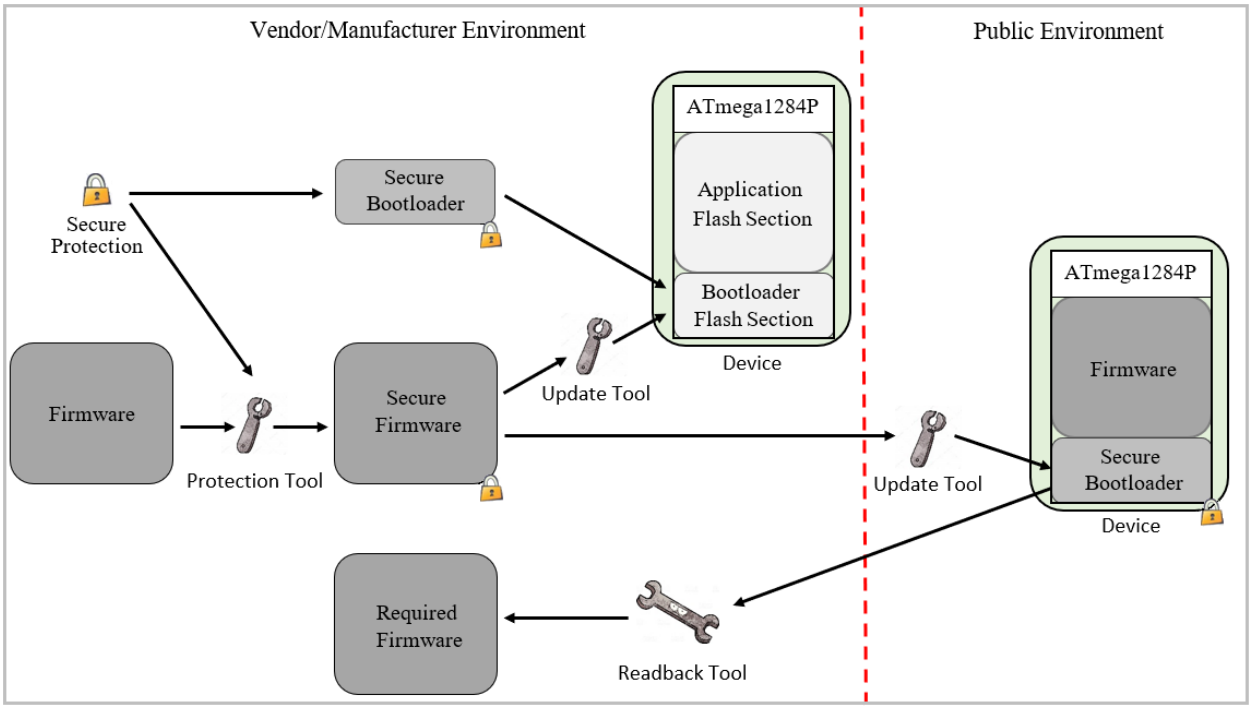


Fig. 7 - Një përmbledhje e sistemit të përditësimit të sigurt të firmuerit

4.5.2 Projektim i hollësishëm

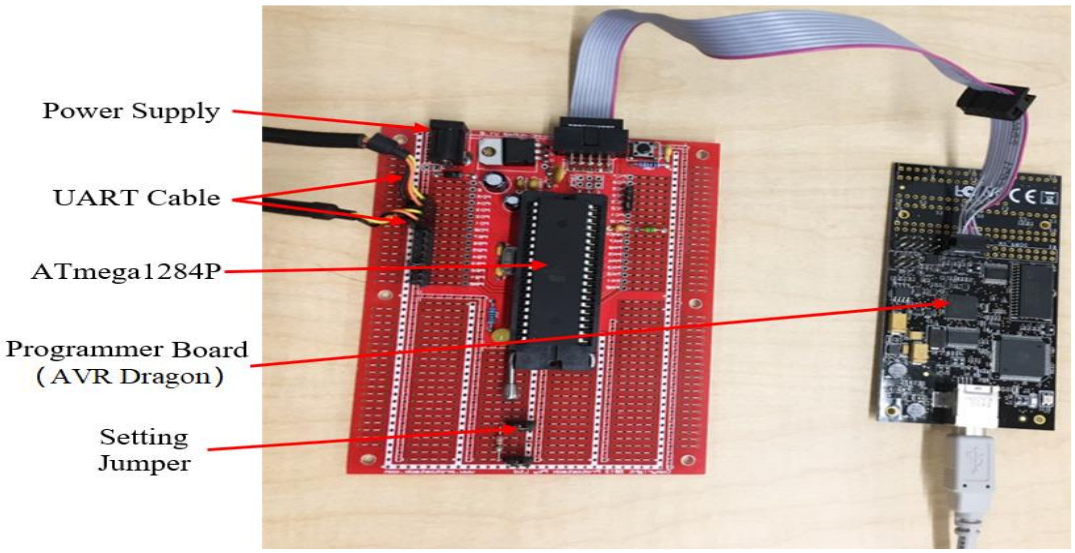


Fig. 8 - Një sistem i sigurt i përditësimit të firmware-it

Tani paraqesim një dizajn të detajuar të sistemit të përditësimit të sigurt të firmware-it.
Si demonstrim, ne zbatojmë projektin në një çip ATmega1284P të treguar në Fig. 8 ATmega1284P është një mikrokontrollues 8-bitësh CMOS me performancë të lartë, me fuqi të ulët bazuar në arkitekturën e RISC të përmirësuar AVR, duke shfaqur memorje flash 128KB ISP, 16KB SRAM, 4KB EEPROM, 32 regjistra pune me qëllim të përgjithshëm, dy ndërfaqe UART, një port serial SPI dhe një ndërfaqja e provës së grupit të veprimit të përbashkët të testit (JTAG)
JTAG ofrohet për korrigjimin dhe programimin në chip. Në lidhje me sigurinë, ATmega1284P ofron një mekanizëm mbyllje programi për sigurinë e softuerit. Në mënyrë të veçantë, MCU ka tre bajtë siguresash, d.m.th bajt të ulët siguresash, bajt të lartë siguresash dhe bajt të zgjatur siguresash dhe një bajt bllokues. Byti i ulët i siguresave përdoret për t'u marrë me operacionet që lidhen me orën. Byte i lartë i siguresave ka disa cilësime të ndryshme, të tilla si mundësimi i portës programuese JTAG / SPI, kohëmatës roje dhe cilësimet e madhësisë së bootloader-it.Byti i zgjatur i siguresave përdoret për vendosje të nivelit të tensionit. Byte-i i bllokimit përdoret për të kontrolluar lejet e leximit dhe shkrimit midis seksionit të bootloader-it dhe seksionit të flash-it të aplikacionit. Mund të përdoret me bajt të lartë të siguresave për të parandaluar aksesin e leximit dhe shkrimit të paautorizuar në flash memorje dhe EEPROM përmes JTAG dhe SPI.

Memoria flash ATmega1284P ndahet në dy seksione: seksioni i aplikacionit dhe seksioni i bootloader. Siguresa e bajtit të lartë përdoret për të alokuar madhësinë e këtyre dy seksioneve. Seksioni bootloader zakonisht përdoret për të shkruar firmware-in e aplikacionit në seksionin flash të aplikacionit. Për të modifikuar bootloader-in, na duhet një bord i jashtëm programues (siç është ai i treguar në Fig. 8) për të shkruar bootloader-in në MCU. Firmware i aplikacionit mund të përdorë kodin e vendosur në bootloader.Mbishkrimi i seksionit të bootloader kërkon përdorimin e udhëzimeve Store Program Memory (SPM). Për të parandaluar që firmware i aplikacionit të shkatërrojë bootloader-in, udhëzimet SPM nuk lejohen të ekzekutohen në hapësirën e adresave të seksionit të aplikacionit flash. Sidoqoftë, tregohet se aplikacioni ka një hile për të modifikuar bootloader-in në mikrokontrolluesin AVR duke vendosur një ndërprerje në firmware-in e aplikacionit dhe duke përdorur udhëzimin "jmp" për ta detyruar AVR-në të hidhet në seksionin e bootloader-it. Bootloader-i pastaj mund të mbishkruhet nga udhëzimet ekzistuese SPM brenda seksionit të bootloader-it.

1) **Bootloader i Sigurt**: Në zbatimin tonë aktual, bootloader i sigurt mund të funksionojë në tre mënyra: modaliteti i ngarkimit të firmware-it, modaliteti i nisjes së firmware-it dhe mënyra e kthimit të leximit. Bootloader është programuar të lexojë cilësimin e kërcyesit në Fig. 8 për t'u futur në mënyra të ndryshme. Gjatë modalitetit të ngarkimit të firmware-it, bootloader-i merr një firmware të koduar nga mjeti i përditësimit të firmware-it dhe verifikon integritetin e tij, i cili parandalon modifikimin arbitrar të firmware-it të koduar.
   Sapo të ketë sukses verifikimi, ngarkuesi i sigurt deshifron firmware-in dhe e kopjon atë në seksionin e memorjes flash të aplikacionit. Më pas, bootloader hyn në modalitetin e nisjes së firmware dhe e nis imazhin në memorjen flash të aplikacionit.
   Ngarkuesi mund të jetë gjithashtu në modalitetin e kthimit. Kur bootloader-i është në këtë modalitet, shitësi mund të komunikojë me të dhe bootloader-i do të kopjojë dhe kriptojë firmware-in nga qip-i. Kjo mund të ndodhë kur çipi ka disa gabime të papritura dhe a
   duhet të bëhet analiza e firmware-it për diagnostikim. Në këtë rast, ngarkuesi i nisjes kërkon që mjeti i kthimit të leximit të sigurojë një emër përdoruesi dhe fjalëkalim për vërtetim, i cili shmang hyrjen e paautorizuar. Këto kredenciale janë paracaktuar dhe ruajtur në EEPROM nga shitësi. Meqenëse ne mundësojmë bitët bllokues të bazuar në harduer të

ATmega1248P, një sulmues nuk mund të kompromentojë firmware ose kredencialet e ndjeshme përmes sulmeve fizike.

2) **Mjet për mbrojtjen e firmware-it:**Mjeti i mbrojtjes së firmware-it përdoret për të gjeneruar një firmware të sigurt, i cili përmban pesë faza: **(i)** Ne krijojmë një çelës kriptimi AES dhe konfigurojmë mjetin e mbrojtjes së firmware-it që të ketë qasje në çelës. Para vendosjes, çelësi AES gjithashtu do të ruhet në çipin ku do të instalohet firmware. Kjo mund të bëhet duke instaluar një version fillestar të firmware, ku çelësi është i koduar në mënyrë të vështirë. **(ii)** Mjeti për mbrojtjen e firmware-it enkripton firmware-in e synuar me tastin AES të gjeneruar më sipër. **(iii)** Mjeti për mbrojtjen e firmware it ushqen firmware-in e koduar në një funksion hash dhe llogarit vlerën e hash-it. Ne e shënojmë vlerën e hashit si ID të firmware-it, pasi që në mënyrë unike i referohet firmware-it. **(iv)** Mjeti i mbrojtjes së firmware-it ushqen ID-në e firmware-it dhe informacionin tjetër themelor të firmware-it (p.sh. madhësia, numri i versionit) në funksionin e hashit. Vlera e hashit e gjeneruar këtu përdoret si shuma e kontrollit të firmware-it për hapin e verifikimit. Ne më pas kriptojmë shumën e kontrollit të hashit me çelësin e paracaktuar. Kontrolli i koduar i kriptuar do të jetë një pjesë e firmware-it të sigurt, në mënyrë që integriteti i firmware-it të jetë i garantuar gjithashtu. **(v)** Mjeti i mbrojtjes së firmware përdor hapat e mëparshëm për të gjeneruar një firmware të sigurt. Firmware i sigurt mund të ndahet në tre pjesë siç tregohet në Fig. 9: Header, firmware i koduar dhe një mesazh lëshimi. Titulli përmban madhësinë e firmware-it, numrin e versionit të firmware-it, ID-në e firmware-it, shumën e kontrolluar të koduar dhe do të ruhet në EEPROM. Mesazhi i lëshimit ofron një përshkrim themelor të firmware-it të sigurt dhe do të shkruhet në seksionin e memorjes flash të aplikacionit duke ndjekur firmware. Kur pajisja starton, bootloader-i mund të shkruajë mesazhin e lëshimit në UART për t'ia shfaqur përdoruesit.

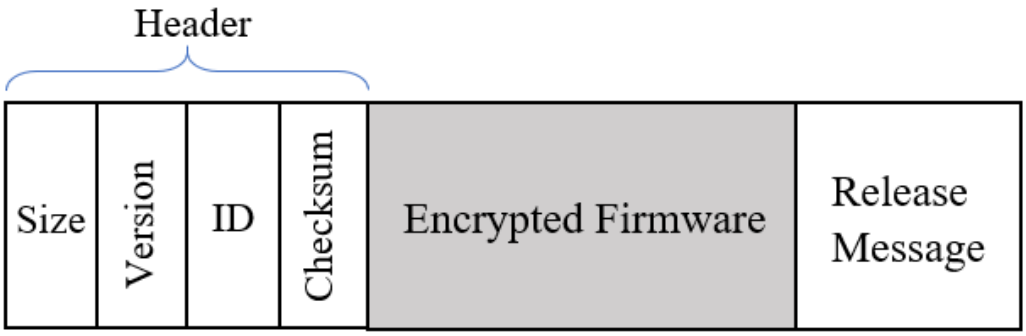


Fig. 9 - Struktura e sigurt e firmware-it

3) **Mjet i përditësimit të firmware-it:** Mjeti i përditësimit të firmware-it ndihmon ngarkuesin duke kopjuar firmware-in në memorjen flash të aplikacionit, e cila përmban dy faza: (i) Vegla e përditësimit të firmware-it dërgon kokën e firmware-it të sigurt te ngarkuesi i sigurt dhe pret përgjigja e verifikimit nga bootloader. Në këtë fazë, bootloader është në modalitetin e ngarkimit të firmware-it dhe do të kryejë verifikimin themelor të firmware, siç u diskutua më parë. (ii) Kur kalon verifikimi, mjeti i përditësimit të firmware-it dërgon firmware të koduar në bootloader dhe pret përgjigjen e verifikimit nga bootloader. Pasi bootloader të marrë me sukses firmware-in e koduar dhe të verifikojë integritetin e tij, mjeti i përditësimit të firmware-it përfundon. Bootloader deshifron firmware-in dhe shkruan firmware-in në seksionin e memorjes flash të aplikacionit.
4) **Mjet i rishikimit të firmware-it:** Vegla e leximit komunikon me firmware-in e sigurt përmes portës UART pas vendosjes, duke i lejuar shitësit të marrin një firmware të koduar për analizë. Për të shmangur hyrjen e paautorizuar në firmware, bootloader i sigurt do kërkojnë një emër përdoruesi dhe fjalëkalim që ishin paracaktuar nga shitësi. Për më tepër, të gjitha të dhënat e transferuara midis mjetit të rikthimit dhe bootloader-it janë të koduara nga çelësi i paracaktuar AES.

4.5.3 Kurthet e mundshme

a) Konfigurimi i gabuar: Konfigurimi i gabuar i çipit mund të çojë në pasoja të rënda. Një keq konfigurim i zakonshëm është dështimi për të aktivizuar siguresat ose bitët e bllokimit, e cila parandalon plasaritjen e bazuar në harduer pas vendosjes. Mundësimi i bajteve të siguresave mund të parandalojë që një sulmues të ketë qasje në memorje përmes portës JTAG / SPI. Konfigurimi i

bitëve të bllokimit mund të parandalojë që një sulmues të programojë flleshin dhe EEPROM duke përdorur Programimin Paralel të Tensionit të Lartë (HVPP).Bordi i programuesit AVR Dragon mund të komunikojë me ATmega1284P në modalitetin HVPP duke lidhur ndërfaqen e tij HV PROG me kunjat e shumta të programueshme I / O në çip. Me HVPP, nëse bitët e bllokimit nuk janë konfiguruar dhe bajtet e siguresave janë vendosur siç duhet, sulmuesit mund të përdorin softuerin Atmel Studio [17] për të rishkruar bajtat e siguresave përmes programuesit AVR Dragon për të mundësuar portat JTAG / SPI dhe për të lexuar firmware-in dhe Kujtesa EEPROM përmes këtyre portave. Kujtoj se informacioni i ndjeshëm është ruajtur në memorjen EEPROM, të tilla si çelësat e deshifrimit, kredencialet dhe informacionet e firmware-it. Konfigurimi i gabuar mund të lejojë që një sulmues të marrë informacione delikate dhe t'i modifikojë ato. Në fakt, edhe me bajtat e siguresave të aktivizuara dhe bitët e bllokimit, me bordin e programuesit, një sulmues mund të rishkruajë të gjithë firmware-in dhe EEPROM ndërsa ajo nuk mund t'i lexojë ato. Prandaj, ATmega1284P mund të mos jetë një çip ideal IoT nëse rishkrimi i flleshit është një shqetësim.

b) Sulmi i Glitch Clock: Për shumicën e qarqeve të integruara (IC), siç është një MCU në rastin tonë, sinjalet e orës përdoren për të sinkronizuar përbërësit e ndryshëm të brendshëm në një qark të integruar. Frekuenca e orës së sistemit të siguruar nga prodhuesi i çipit është frekuenca në të cilën sinjali i orës arrin çdo komponent IC në mënyrë korrekte, dhe të gjitha udhëzimet mund të ekzekutohen normalisht në këtë frekuencë. Mbingarkesa është një praktikë e rritjes së frekuencës së orës së sistemit përtej frekuencës së dhënë nga prodhuesi. Si rezultat, overclocking mund të përmirësojë performancën e CPU / MCU deri në një farë mase. Sidoqoftë, kjo nuk është e vërtetë botërisht për shkak të kufizimeve të pajisjeve. Nxehtësia shtesë mund të gjenerohet në këtë rast, gjë që mund të pengojë gjithashtu performancën. Prandaj, një IC mund të mos funksionojë si duhet kur ndodh overclocking. Një defekt i orës është një periudhë e shkurtër kohe kur frekuenca e orës së sistemit rritet papritur. Mund të çojë në ekzekutim të pasaktë të udhëzimeve dhe në dalje të paqëndrueshme të të dhënave. Një sulmues mund të kryejë sulmin e defektit të orës për të anashkaluar udhëzimet kriptografike të tilla si kriptimi dhe kontrolli i integritetit.

c) Rrjedhja e Çelësit: Një sulmues mund të marrë çelësin e kriptimit në mënyra të ndryshme për shkak të pakujdesisë së shitësve ose të metave ekzistuese të mjeteve. Për shembull, çelësi mund të kodohet fort në mjetin e mbrojtjes së firmware-it. Në këtë rast, një sulmues mund të kryejë procesin e kundërt të inxhinierisë dhe të nxjerrë çelësin nga fajlli binar. Edhe nëse çelësi është i koduar në

firmware të koduar, një tjetër grackë e mundshme përfshin përdorimin e një algoritmi të pasigurt kriptimi, i cili mund t'i nënshtrohet kriptanalizës. Në rrethana të tilla, një sulmues mund të thyejë kriptimin e firmware-it dhe të marrë çelësin e koduar që është ruajtur në EEPROM. Sapo sulmuesi të marrë çelësin, mund të vendosen sulme të ndryshme të tjera, të tilla si fabrikimi i një firmware me qëllim të keq. Si zgjidhje, shitësit duhet të mbajnë çelësat e tyre të enkriptimit në një vend të sigurt.

d) Duke anashkaluar Verifikimin e Firmware-it: Tani demonstrojmë një dizajn të gabuar të procesit të përditësimit të firmware-it. Mjeti për përditësimin e firmware-it. e ngarkon firmware-in faqe për faqe në çip. Kjo është, çipi së pari merr të dhënat nga mjeti i përditësimit dhe i fsheh ato në një buffer. Mbrojtësi do të mbajë të dhënat derisa të arrijë madhësinë e një faqe, d.m.th., 256 bajte. Bootloader deshifron këtë faqe dhe e shkruaj atë në seksionin e flash memorjes të çipit. Ky proces përsëritet derisa i gjithë firmware të jetë shkruar në qip. Vegla e përditësimit të firmware-it më pas dërgon një mesazh përfundimi në çip. Më pas kryhet verifikimi i integritetit të firmware-it. Nëse verifikimi dështon, bootloader fshin memorjen flash të aplikacionit duke pasur parasysh që firmware i dëmtuar / me qëllim të keq është shkruar tashmë në seksionin e flash memorjes të aplikacionit. Kjo strategji e përditësimit të firmware-it më lart është me të meta. Nëse një sulmues merr mjetin e përditësimit të firmware-it dhe përpiqet të sulmojë një pajisje, ai mund ta ndryshojë mjetin në mënyrë që një firmware i padobishëm të ngarkohet në pajisje, por vegla e shtrembër e përditësimit të firmware-it nuk dërgon mesazhin e ndërprerjes në mënyrë që të mos kryhet verifikimi . Kjo do të anashkalojë procesin e verifikimit dhe një firmware i hedhurinave është shkruar në pajisje. Një mundësi për të mposhtur këtë lloj sulmi është të ndash të gjithë kujtesën e programit flash të aplikacionit në dy ndarje: ndarja e parë për firmware aktual dhe ndarja e dytë për firmware i ri. Mjeti i përditësimit ngarkon firmwarin e ri në ndarjen e dytë dhe verifikon integritetin e tij. Pasi të verifikohet integriteti, firmware i ri kopjohet në ndarjen e parë. Një përditësim në ajër (OTA) shpesh përdor këtë strategji të ndarjeve të shumta në procesin e përditësimit të firmwarit edhe pse procesi aktual OTA mund të jetë i ndryshëm.

e) Të metat ekzistuese në mjetin e kthimit të leximit: Kujtohuni që bootloader kërkon kredencialet e përdoruesit (d.m.th. emrin e përdoruesit dhe fjalëkalimin) kur shitësi përpiqet të përdorë mjetin e leximit për të komunikuar me të. Nëse shitësi ndërvepron në distancë me bootloader-in përmes internetit dhe kanalit të komunikimit nuk është i siguruar (p.sh., pa kriptim), një kundërshtar mund

të përgjojë përmbajtjen e komunikimit, duke përfshirë kredencialet e përdoruesit. Kundërshtari mund të përdorë kredencialet për të lexuar ose modifikuar firmware. Prandaj, mjeti i kthimit duhet të zbatojë praktikat e duhura të sigurisë në rrjet.

5 DISKUTIM

Tabela i përmbledh mënyrën e mbrojtjes së ndërfaqeve I / O të ATmega1284P. ATmega1284P nuk mund të çaktivizojë UART, i cili do të mbrohet nga një bootloader i sigurt. Bootloader mund të programohet dhe të kontrollojë bashkëveprimin me çipin përmes UART. Siç u diskutua më lart, ATmega1284P në të vërtetë nuk është një mikroprocesor ideal për aplikime të sigurta IoT. Me bordin e programuesit në Fig. 8, bootloader dhe firmware i ATmega1284P mund të mbishkruhen në mënyrë arbitrare edhe me bajte të siguresave të aktivizuara dhe bit bllokimi.

Table 1 - Masat e sigurisë ATMEGA1284P I/O

I/O	Funksionaliteti	Masat e Sigurisë për ndërfaqet I / O
JTAG	Ndërfaqja e provës për korrigjimin dhe programimin në çip. Flash, EEPROM, përdorimi dhe bitët e bllokimit mund të programohen përmes kësaj ndërfaqeje.	Byte siguresash + bit bllokues
SPI	Përdoret për komunikim serial dhe shkëmbim të dhënash me pajisjet periferike; Komunikoni me bordin e programuesit për të shkarkuar bootloader	Byte siguresash + bit bllokues
UART	Përdoret për komunikimin e të dhënave me pajisje të jashtme; Komunikoni me mjetin e përditësimit të firmware-it	Bootloader i sigurt me kontroll të integritetit dhe fjalëkalim

Një çip për një aplikim të sigurt IoT duhet të mbështetesë sigurinë nga pesë aspekte: **hardueri, sistemi operativ / firmware, softueri, rrjeti dhe të dhënat e gjeneruara dhe të mirëmbajtura brenda sistemit**. **(i) Hardware security:** Siguria e pajisjes është kritike kur kundërshtari mund të ketë qasje fizike në pajisjet IoT. Portet e të tilla si UART / JTAG duhet të çaktivizohen në produktet përfundimtare. **(ii) Siguria e sistemit operativ (OS) / firmware:** Mekanizmat janë të nevojshëm për të parandaluar që firmware të ndryshohet nga hakimi fizik dhe malware. Sistemi i fajllave, proceset, aktivitetet e kujtesës dhe portet e rrjetit duhet të monitorohen vazhdimisht për të zbuluar aktivitete dinamike me qëllim të keq. **(iii) Siguria e softuerit:** Praktikat e sigurta të kodimit janë të nevojshme për të zvogëluar dobësitë në aplikimet e IoT. **(iv) Siguria dhe privatësia e rrjetit:** Një sistem IoT është një sistem në rrjet, dhe i gjithë sistemi duhet të sigurohet nga skaji në skaj. **(v) Siguria e të dhënave:** Të dhënat e ndjeshme duhet të kriptohen dhe kriptimi i flashit preferohet nëse pajisjet vendosen në natyrë.

6 PËRFUNDIM

Gjatë këtij punimi unë kam hulumtuar për metodat e mundshme të hardware hacking, tiparet më të rëndësishme të hardware hacking , sulmet e mundshme fizike të cilat realizohet nëpërmjet interfejsave të zakonshëm si dhe sulmet nga distanca.Ky hulumtim do t'i ndihmojë shitësit të kuptojnë rëndësinë e mekanizmit të sigurt të përditësimit të firmware-it për sistemet IoT të bazuara në MCU dhe të shmangë kurthet e mundshme.

Sulmet e firmware-it dhe masat përkatëse të mbrojtjes kanë marrë mjaft vëmendjen e fundit për shkak të rritjes shpërthyese të tregut të IoT. Interneti i Gjërave (IoT) po rritet gjithnjë e më shumë me hapa të shpejtë, Por kjo rritje dramatik përkeqëson gjithashtu dy rreziqe themelore në IoT. Së pari, hakerat në IoT mund të përpiqen të fitojnë kontrollin e pajisjeve të mundësuara nga interneti, duke shkaktuar pasoja negative në botën fizike. Duke qenë se objektet me lidhje interneti variojnë nga pajisjet shtëpiake dhe automobilat deri te komponentët kryesorë të infrastrukturës, ky rrezik është potencialisht i rëndë. Në të vërtetë, në vitet e fundit, hakerat kanë fituar kontrollin e makinave, trenave dhe digave, dhe disa ekspertë mendojnë se edhe aeroplanët komercialë mund të rrezikohen. Së dyti, pajisjet IoT paraqesin një rrezik jashtëzakonisht të madh për vetë stabilitetin e internetit, pasi ato janë të prekshme për t'u hackuar. Ky punim heton çështjet e sigurisë që mund të ekzistojnë në procesin e përditësimit(update) të firmware-it të aplikacioneve IoT të bazuara në mikrokontrollues. Ne demonstrojmë që mekanizmi i përditësimit të firmware-it të pajisjes popullore të cilësisë së ajrit PurpleAir i nënshtrohet si sulmit fizik ashtu edhe sulmit në distancë. Për t'iu kundërvënë këtyre sulmeve, ne hetojmë mekanizmat e sigurt të përditësimit të firmware-it për një mikrokontrollues të trashëguar ATmega1284P dhe diskutojmë kurthet që mund të ndodhin gjatë implementimeve. Gjate ketij studimi ne kemi vrejtur që një çështje kryesore e pajisjes PurpleAir është se ajo përdor çipin ESP8266, i cili nuk ka ndonjë mekanizëm të integruar hardverik për sigurinë. Për shembull, nuk ka asnjë mënyrë për të çaktivizuar portin e tij UART gjë e cila e bënë të pasigurt paisjen dhe mundë ti nënshtrohet shumë lehtë sulmit fizik. Për të parandaluar që sulmuesi të përdorë sulme fizike dhe të marrë të dhëna senzitive, të tilla si çelësi i deshifrimit ose firmware-in, ndërfaqet e korrigjimit, të tilla si portet JTAG, SPI dhe UART, duhet të çaktivizohen ose të paktën të kenë qasje të kufizuar (p.sh. të mbrojtura nga një fjalëkalim i fortë).

Pasi një analize me kujdes në mekanizmin PurpleAir OTA, identifikojmë një dobësi të rëndë: pajisja nuk e vërteton serverin dhe serveri nuk e vërteton pajisjen, qka dmth se nuk ka vërtetim

reciprok midis pajisjes dhe serverit. Për më tepër, komunikimi është në tekst të thjeshtë. Dhe ky vërtetimit reciprok ngre çështje të sigurisë.

Një pajisje e sigurt IoT ka nevojë për veçori të sigurisë së pajisjes.Dhe si përfundim për sigurinë e paisjes MITER Cyber Academy propozon kërkesa për hartimin e një sistemi të sigurt për përditësimin e firmware-it pa u lidhur në internet dmth duke e bërë përditësimin lokalisht në mënyrë që firmware të përditësohet sigurt.Sa i përket sulmit në distancë për ta mposhtur këtë sulm që bëhet gjate përditësimit të firmware-it, duhet të përdoret vërtetimi i ndërsjellë dhe kriptimi.Për të garantuar një përditësim të sigurt të firmware-it, ne do të përfshijmë kriptimin dhe vërtetimin gjatë procesit të përditësimit. Në mënyrë të veçantë, shitësi do të kriptojë firmware-in përpara se të lëshohet, ndërsa çelësi i dekriptimit duhet të jetë i sigurt brenda secilit produkt. Vetëm produkti mund të deshifrojë firmware-in. Produkti duhet të verifikojë integritetin e firmware-it gjithashtu.

7 REFERENCAT

[1] J.G. Grand, R.G.Russell, EN-Hardware Hacking - Have Fun While Voiding Your Warranty, Second Edition, Canada, 2004.

[2] EDUCBA, 2020 [https://www.educba.com/hardware-hacking/]

[3] Ctaylor, NOVEMBER 14, 2013 [https://www.sparkfun.com/news/1314]

[4] H.G. Graceful, 2019 [https://gracefulsecurity.com/an-introduction-to-hardware-hacking/]

[5] Ravi, "Basics of microcontrollers history, structure and applications.", https://www.electronicshub.org/microcontrollers-basics-structure -applications, November 13, 2017.

[6] Espressif-Systems, "Esp8266: Hardware design guidelines." https://www.espressif.com/sites/default/files/documentation/esp8266 h ardware design guidelines en.pdf, 2018.

[7] F. Ahlberg and A. Gratton, "esptool.py." https://github.com/espressif/e sptool, April 16, 2019.

[8] E. S. C. Platform, "Esp8266," Espressif Systems, 2013.

[9] "Esp8266 architecture and arduino gui." https://annefou.github.io/IoT i ntroduction/02-ESP8266/index.html, Nov 2019.

[10] Espressif-Systems., "Esp8266 sdk: Getting started guide." https://www.espressif.com/sites/default/files/documentation/2a-esp 8266-sdk getting started guide en.pdf, April 16, 2019.

[11] E. S. I. Team, "Esp8266 fota introduction." https://www.espressif.com/ sites/default/files/99c-esp8266 ota upgrade en v1.6.pdf, 2016.

[12] S. Hanna, R. Rolles, A. Molina-Markham, P. Poosankam, J. Blocki, K. Fu, and D. Song, "Take two software updates and see me in the morning: The case for software security evaluations of medical devices.," in HealthSec, 2011.

[13] J. Li, C. Chang, D. Shi, W. Xia, and L. Chen, “A new firmware upgrade mechanism designed for software defined radio based system,” in Proceedings of the 2012 International Conference on Information Technology and Software Engineering, pp. 277–283, Springer, 2013.

[14] J. D. Tenbarge, R. Timmerman, M. Nierzwick, R. E. Reinke, D. Birtwhistle, J. R. Long, R. P. Sabo, P. E. Pash, and D. B. Markinsohn, “Firmware update in a medical device with multiple processors,” Apr. 19 2012. US Patent App. 12/905,498.

[15] J. D. Tenbarge, R. Timmerman, M. Nierzwick, R. E. Reinke, D. Birtwhistle, J. R. Long, R. P. Sabo, P. E. Pash, and D. B. Markinsohn, “Firmware update in a medical device with multiple processors,” Apr. 19 2012. US Patent App. 12/905,498.

[16] Microchip, “The atmel avr dragon debugger.” http://ww1.microchip. com/downloads/en/devicedoc/atmel-42723-avr-dragon userguide.pdf, 2016.

[17] Microchip., “Atmel studio 7.” https://www.microchip.com/mplab/avr-s upport/atmel-studio-7, June 15, 2019.

[18] P. Bryan, L. Lan, Z. Yue, D. Rajib, L. Zhen, B. Mostafa, and F. Xinwen, “On misconception of hardware and cost in iot security and privacy,” in IEEE International Conference on Communications, 2019.