1

Пример хардверског изазова за такмичење из етичког хаковања

Маја Миљанић, др Лазар Карбунар

***Садржај* — За национално такмичење из етичког хаковања развијен је хардверски изазов - уређај са *ESP32-S3* микроконтролером, са намерно остављеним безбедносним пропустима које је потребно експлоатисати у оквиру четири задатка. Безбедносни пропусти на овом уређају симулирају рањивости које се често јављају на *IoT* и сродним уређајима: незаштићени портови, екстракција и реверзно инжењерство фирмвера, небезбедно чување криптографских тајни, несигурне лозинке и небезбедно бежично ажурирање.**

***Кључне речи* — безбедност хардвера, етичко хаковање, интернет интелигентних уређаја**

I. Увод

ТАКМИЧЕЊА из етичког хаковања облика *CTF* (*Capture The Flag*) пред такмичаре стављају задатке из различитих домена информационих технологија: реверзно инжењерство (енг. *rev*), бинарна експлоатација (енг. *pwn*), веб, форензика, *OSINT* (*Open Source Intelligence*), криптографија, потпуно компромитовање система (енг. *full pwn*), као и хардвер. Циљ сваког задатка је да такмичар пронађе заставицу (енг. *flag*), односно, карактеристичан стринг којим се доказује да је задатак решен експлоатисањем различитих рањивости.

За потребе националног такмичења у етичком хаковању у Србији развијен је хардверски изазов са карактеристикама *IoT* (интернет интелигентних уређаја, интернет ствари, енг. *Internet of Things*) уређаја због њихове велике популарности, али недовољне свести о томе да ови

[1]Маја Миљанић, Рачунарски факултет, Београд, Србија (e-mail: mmiljanic5223@raf.rs)
др Лазар Карбунар, Рачунарски факултет, Београд, Србија (e-mail: lkarbunar@raf.rs)

уређаји често садрже рањивости које могу компромитовати осетљиве податке корисника, угрозити критичну инфраструктуру, имовину, па чак и здравље људи. У истраживању о свести људи о безбедности *IoT* уређаја у Пољској, око 60% испитаника сматра да је удобност коју им пружа *IoT* уређај важнија од његове безбедности [1], док 60% испитаника у сличном истраживању спроведеном у Сједињеним Америчким Државама сматра да они ,,нису вредна мета за хакере'' [2]. Са друге стране, путем платформе *Shodan* могуће је претраживати и приступати незаштићеним уређајима повезаним на Интернет [3] - камерама за видео надзор, бејби мониторима, телевизорима и другим паметним уређајима који се налазе у домовима и пословним објектима. Стога, кључна мотивација за израду хардверског изазова и писање овог рада је да се такмичарима и широј јавности представе рањивости које се јављају на уређајима у свакодневној употреби.

Интернет интелигентних уређаја обезбеђује конекцију сваке особе са свим стварима, ма у ком тренутку и на ком месту се та особа и те ствари (уређаји) налазле [4]. *IoT* системи имају примену у паметним градовима, индустрији 4.0, здравству, пољопривреди, транспорту, угоститељству и управљању домаћинством. Уређаји могу имати различите улоге - неки су сензори који прикупљају информације о стању спољашње средине, као што су температура и влажност ваздуха, осветљење или ниво загађења, док су други актуатори који извршавају одређене радње, попут укључивања расвете, подешавања грејања или отварања вентила. Прикупљени подаци се углавном шаљу ка централном серверу (најчешће у облаку), где се чувају и обрађују. Крајњи корисници могу да приступе овим информацијама путем мобилних и веб апликација, а такође имају могућност да задају наредбе уређајима, чиме се омогућава интерактивна контрола и аутоматзација свакодневних процеса. Између уређаја и сервера често постоји још један слој - ивица мреже (енг. *edge*). Крајњи уређаји комуницирају са уређајима на ивици мреже, који доводе услуге и функције рачунара у облаку ближе крајњем уређају, омогућавајући велики проток, ниску латенцију и приступ информацијама у реалном времену [5].

Хардвер *IoT* уређаја изграђен је од различитих микроконтролера, попут *ESP*, *STM*, *PIC*, *NXP* и других, уз мноштво сензора и актуатора као што

су фотоелектрични сензори, пироелектрични сензори, ултразвучни сензори, сензори вибрације, лидар, микрофон, камера, *RFID* читач, серво и степ мотори, екран, звучник и други. За комуникацију између компонената користе се различити видови серијске комуникације - *UART*, *SPI*, *I2C*, *USB* и други, а за комуникацију између крајњих уређаја и других слојева *IoT* система у употреби су технологије као што су *Wi-Fi*, *LoRaWAN*, *GSM*, *NB-IoT*, *BLE*, *Bluetooth Classic* и *Zigbee*, док се на апликативном нивоу користе протоколи попут *HTTP*, *MQTT* и *WebSocket*. Ова разноликост технологија отежава управљање безбедношћу у *IoT* системима јер свака појединачна компонента и технологија уноси сопствени скуп рањивости.

Листа *OWASP* (*Open Worldwide Application Security Project*) топ десет из 2018. године, као најчешће безбедносне пропусте у *IoT* системима, наводи пропусте попут слабих лозинки које се могу лако открити нападом „грубе силе'' (енг. *brute force*), небезбедне сервисе на уређајима, непостојање безбедног начина ажурирања уређаја и валидације фирмвера и друге [6]. Међу честе рањивости убрајају се и непостојање енкрипције при комуникацији преко мреже, непостојање енкрипције меморије, омогућен приступ командној линији и *UART*-у, што омогућава и измену функционисања самог уређаја [7].

Mirai је један од забележених напада на *IoT* систем приликом ког је *Mirai* малвером заражено 600.000 уређаја који су постали део ботнета који је вршио *DDoS* (*Distributed Denial of Service*) нападе [8]. Аутомобили такође имају функционалности *IoT* уреађаја, па, осим експлоатације рањивости *CAN* (*Controller Area Network*) магистрале [9], аутомобили могу бити хаковани и даљински. У [10] је објашњено како је преузета контрола над унутрашњом *CAN* мрежом возила коришћењем „Отвореног система за надзор возила'' (енг. *Open Vehicle Monitoring System*, *OVMS*) и демонстрирана је могућност да се утиче на безбедносно критичне системе: рад електричног погона, држање стања возила и позиционирање.

Полазна претпоставка овог рада је да развој хардверског изазова који реплицира уобичајене пропусте *IoT* уређаја може допринети подизању

свести о хардверским рањивостима код корисника и младих истраживача. Оригинални допринос овог рада огледа се у дизајну и имплементацији јединственог хардверског изазова намењеног националном такмичењу у етичком хаковању. Рад пружа увид у дизајн уређаја, начин интеграције рањивости и израду сценарија за такмичаре, чиме се обезбеђује практично окружење за изучавање безбедносних пропуста у *IoT* системима. На овај начин рад доприноси развоју методологије за едукацију у области хардверске безбедности.

У делу II биће описан развијени хардверски уређај. У деловима III-VI биће објашњена четири имплементирана задатка у оквиру којих такмичари треба да експлоатишу неке од рањивости које се често јављају код *IoT* уређаја, попут отворених портова за програмирање и дебаговање, небезбедно чување криптографских тајни, небезбедно бежично ажурирање система (енг. *over-the-air update*) и могућност реверзног инжењерства фирмвера.

II. ХАРДВЕРСКИ ИЗАЗОВ ЗА ТАКМИЧЕЊЕ ИЗ ЕТИЧКОГ ХАКОВАЊА

Хардверски задатак састоји се од уређаја са *ESP32-S3 Mini* микроконтролером. Иако не садржи све компоненте једног *IoT* уређаја, овај уређај ће послужити да демонстрира неке од најчешћих рањивости које се на таквим уређајима јављају. Осим микроконтролера, на уређају се налази *UART* (*Universal Asynchronous Receiver Transmitter*) порт, *USB C* (*Universal Serial Bus*) порт, четири дугмета од којих свако служи да покрене један од четири задатка и четири лед диоде које означавају покренути задатак (Сл. 1). Поред уређаја, такмичарима је на располагању и *USB-UART* конвертер (програматор) и *USB C* кабл.

Сл. 1. Изглед уређаја

Породица *ESP32-S3* микроконтролера обухвата неколико верзија 32-битних микроконтролера заснованих на 32-битном *LX7* двојезгарном чипу, са интегрисаним 2.4 GHz *Wi-Fi* модулом и *Bluetooth* (или *Bluetooth Low Energy*) модулом. Ови микроконтролери засновани су на *Xtensa* архитектури са 32-битним двојезгарним процесорима брзине до 240 MHz, са 512 KB *SRAM* и 384 KB *ROM* меморије на чипу. У зависности од модела, ови микроконтролери имају до 45 *GPIO* пинова, као и различите интерфејсе: *SPI*, *I2C*, *UART*, *PWM* и слично. Различити модели микроконтролера имају различите величине екстерне *PSRAM* меморије (2,8 или 16 MB) и екстерне флеш меморије (4, 8 или 16 MB). Мини верзије микроконтролера имају 8 MB флеш меморије на чипу. Микроконтролери се разликују и по томе да ли имају антену на чипу или само конектор за исту. За потребе хардверског изазова коришћен је *ESP32-S3 Mini* микроконтролер који има 39 *GPIO* пинова и уграђену антену.

За напајање уређаја довољно је 5 V, па га такмичар може напајати из лаптоп рачунара датим *USB C* каблом. Задаци се покрећу тако што се уређај укључи и притисне се одговарајуће дугме на уређају. Уколико такмичар жели да покрене неки други задатак, потребно је да искључи уређај, укључи га поново и одабере жељени задатак.

III. ЗАДАТАК 1

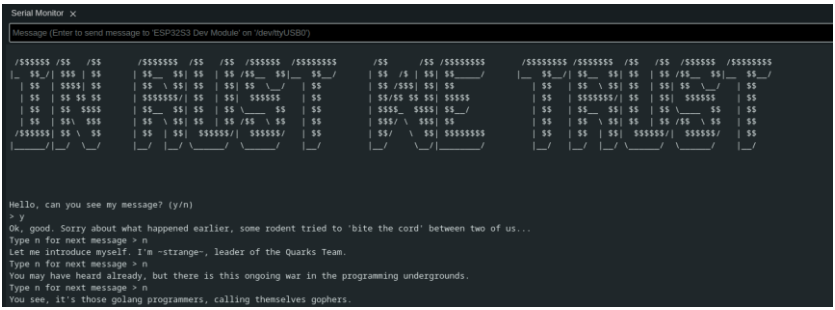
Овим задатком је приказано како незаштићен порт, у овом случају *UART*, доводи до компромитовања осетљивих информација. На уређају је порт видљив и пинови су означени. Потребно је да такмичар повеже уређај и свој рачунар користећи *USB-UART* конвертер и помоћу неког од програма за серијску комуникацију читањем и уписивањем текста у серијску конзолу дође до скривене заставице и на тај начин реши задатак.

UART је облик асинхроне комуникације у којем уређаји који размењују податке не деле заједнички тактни сигнал (енг. *clock*). Уместо тога, сваки уређај ради са сопственим тактом који мора бити подешен на исту вредност брзине преноса. Подаци се шаљу у облику карактера дужине 7 или 8 битова, при чему се на почетку сваког карактера шаље старт бит, а на крају један или два стоп бита. По потреби, пренос се може допунити битом парности за проверу грешака у комуникацији. Бауд (енг. *baud rate*) је број симбола послат у секунди и исту вредност морају користити и пошиљалац и прималац података, да би комуникација била синхронизована [11]. Због једноставности употребе, *UART* је један од најчешћих интерфејса за развој система и дебаговање.

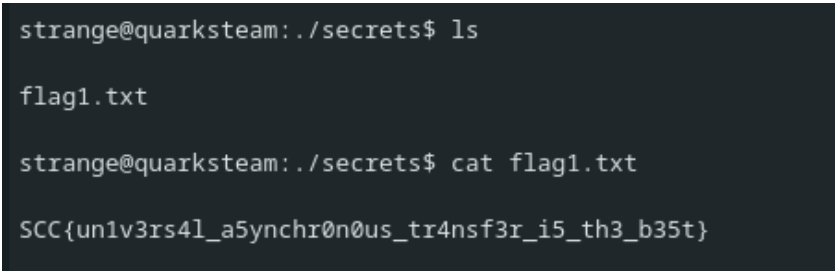
На уређају постоји група од 5 обележених пинова – *RX, TX, VCC, GND, IO0* – тако да се *UART* пинови могу једноставно идентификовати без коришћења унимера. Пин *IO0* се користи да пребаци уређај у режим програмирања када је потребно снимити фирмвер на уређај. Да би се уређај повезао са рачунаром, треба користити *USB-UART* конвертер. Иначе се за емулацију серијске комуникације могу користити и вишенаменски уређаји попут *Attify Badge* и *Bus Pirate*. Пошто се уређај напаја преко *USB C* порта, довољно је користити три пина и повезати их тако да су *RX*, *TX* и *GND* пинови на уређају спојени са *TX*, *RX* и *GND* пиновима на *USB-UART* конвертеру, респективно.

Следећи корак је читање серијског излаза. Пошто је *UART* асинхрони протокол комуникације, потребно је знати брзину преноса података коју уређај користи за слање података другом уређају. Такође је потребан програм за серијску комуникацију као што су *minicom*, *screen* или *Arduino Serial,* да би се подесила брзина преноса података и читао излаз.

Такмичар може пробати стандардне брзине преноса података као што су 9600, 19200 бауда итд. док не пронађе праву. Када такмичар исправно успостави везу између рачунара и уређаја, на серијској конзоли добија испис (Сл. 2). Затим је потребно да се у неколико наврата у серијску конзолу упише тражена вредност. Кориснику је затим понуђено да приступи терминалу (Сл. 3), након чега се може видети да се у тренутном директоријуму налази фајл *flag.txt*, чији се садржај може прочитати наредбом *cat* (Сл. 3).



Сл. 2. Изглед серијске конзоле након покретања првог задатка



Сл. 3. Проналажење и штампање заставице

Фирмвер уређаја написан је у програмском језику *Rust*. Кôд овог задатка састоји се од функција за уписивање и читање са *UART*-а. Након што такмичар успе да приступи терминалу уређаја и унесе наредбу за штампање садржаја фајла у ком се налази заставица (*cat flag.txt*), позива се функција која декодира замаскирани (енг. *obfuscated*) низ бајтова који

представљају заставицу и уписује их на *UART*. Како би се онемогућило директно читање заставица из изворног кода фирмвера (уколико такмичари ишчитају цео садржај флеш меморије), стринг литерали су замаскирани током компајлирања коришћењем библиотеке *cryptify*.

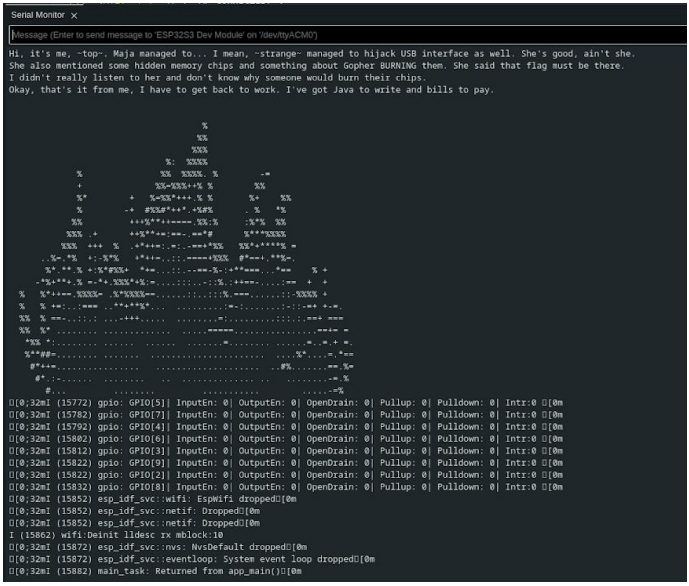
IV. ЗАДАТАК 2

У овом задатку је приказано како сензитивни подаци на уређају могу бити складиштени на небезбедан начин, па их је могуће компромитовати. У овом случају се заставица налази у ефјуз (енг. *eFuse*) меморији. Ефјуз је тип фјуз (енг. *fuse*) меморије интегрисане у сам чип, која се електрично програмира [12]. Њена главна карактеристика је то што омогућава једнократно програмирање, јер се упис података реализује кроз трајну физичку измену унутрашње структуре. Једном испрограмирани ефјуз битови више не могу бити измењени, што обезбеђује трајност, интегритет и безбедност података који се у њима чувају [13]. Ефјуз меморија има посебан значај у процесима провере аутентичности фирмвера приликом безбедног покретања уређаја (енг. *secure boot*), као и у поузданом складиштењу криптографских кључева [13].

ESP32-S3 Mini микроконтролер користи ефјуз технологију. Кључ за енкрипцију флеш меморије чува се у ефјуз меморији и заштићен је од даљег уписа или софтверског читања. Хеш вредност јавног кључа за потписивање фирмвера, који је потребан за верификацију фирмвера током процеса безбедног покретања фирмвера, такође се чува у ефјуз меморији. Блокови ефјуз меморије могу се конфигурисати тако да буду заштићени од читања. Међутим, на овом уређају заставица је сачувана у ефјуз меморији, али тај блок меморије није заштићен од читања.

Испис на конзоли приликом покретања другог задатка наговештава да решење може имати везе са ефјуз меморијом наглашавањем речи *спаљивање* (енг. *burn*) која се односи на трајно програмирање ефјуз меморије (Сл. 4). За проналажење заставице потребно је ишчитати ефјуз меморију, што је могуће извршити помоћу алата *espefuse*:

```
espefuse.py dump --port /dev/ttyACM0
```



Сл. 4. Изглед серијске конзоле након покретања другог задатка

Резултат ишчитавања ефјуз меморије је текст који је хекс енкодиран. Декодирањем текста и променом редоследа бајтова (енг. *endianess*), добија се заставица. Имплементација овог задатка се, осим кода за испис текста задатка на конзоли, састоји од уписивања вредности заставице у ефјуз наредбом:

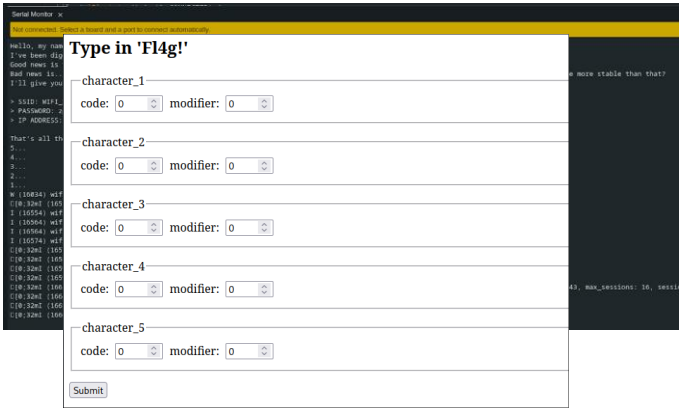
```
espefuse.py -p /dev/ttyACM0 burn_block_data --offset 8 BLOCK3 output.bin
```

V. ЗАДАТАК 3

Уређај у овом задатку представља *USB HID*, конкретно, уређај емулира тастатуру. Ову особину поседују и малициозни уређаји за инјекционе нападе путем тастатуре (енг. *keystroke injection attack*), као популарни уређај *Rubber Ducky*.

За решавање овог задатка користи се *USB C* порт. Када се изазов покрене, уређај укључује *Wi-Fi* приступну тачку и на конзоли се приказује локална *IP* адреса уређаја. Оно што се веома често дешава је да *IoT* уређаји долазе са креденцијалима произвођача (или без икаквих креденцијала) када је у питању конфигурација (енг. *provisioning*). Довољно је повезати рачунар или мобилни телефон на *Wi-Fi* приступну тачку уређаја која се исписује на серијској конзоли (Сл. 5) и приступити веб страници на датој адреси 192.168.71.1. Да би се добила заставица, потребно је уређају послати поруку која садржи текст „Fl4g!“ користећи форму на веб страници (Сл. 6). Ако се неки текст пошаље кроз форму и истовремено отвори текст едитор на рачунару на који је уређај повезан, послати текст би требало да буде исписан у едитору. Да уређај емулира тастатуру, може се видети и у излазу наредбе *dmesg*. Исправан унос подразумева *USB HID* кодове за сваки карактер, са опционим модификатором, након чега се на веб страници исписује заставица. *USB HID* кодови за карактере које је потребно унети (Fl4g!) су 9, 15, 33, 10 и 30 респективно. За карактер F, осим кода 9, који представља мало слово, односно, f, потребно је послати и модификатор 225 (леви шифт тастер на тастатури), а исто је потребно урадити и за карактер !.

Сл. 5. Приказ серијске конзоле након покретања трећег задатка



Сл. 6. Форма на веб страници

За имплементацију *USB HID* уређаја коришћена је *tinyUSB* библиотека. Функција *key_down* која прима листу тастера (до шест истовремено) и један модификатор (нпр. шифт) проверава да ли је *USB* уређај прикључен, помоћу функције *tud_mounted()*, и ако јесте, шаље *HID* извештај (*tud_hid_n_keyboard_report*) рачунару, симулирајући притисак тастера на тастатури (Сл. 7).

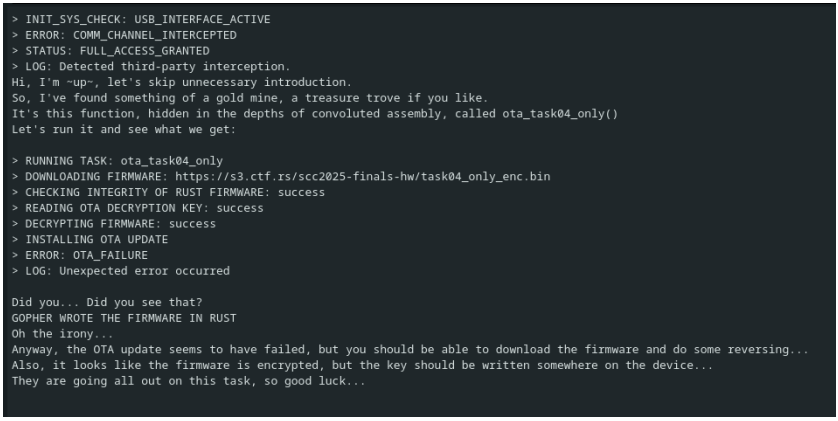
```
pub fn key_down(key: &Vec<u8>, modifier: u8) {
    let mut keys_slice: [u8; 6] = [0; 6];
    if key.len() <= 6 {
        for i in 0..key.len() {
            keys_slice[i] = key[i];
        }
    }

    unsafe {
        if tud_mounted() {
            if modifier == HID_KEY_SHIFT_LEFT {
                tud_hid_n_keyboard_report(0, 1, 0, [HID_KEY_SHIFT_LEFT, key[0], 0, 0, 0, 0].as_mut_ptr());
            } else if modifier == HID_KEY_SHIFT_RIGHT {
                tud_hid_n_keyboard_report(0, 1, 0, [HID_KEY_SHIFT_RIGHT, key[0], 0, 0, 0, 0].as_mut_ptr());
            } else {
                tud_hid_n_keyboard_report(0, 1, 0, [0, key[0], 0, 0, 0, 0].as_mut_ptr());
            }
        }
    }
}
```

Сл. 7. Функција *key_down*

VI. ЗАДАТАК 4

У овом изазову приказано је више пропуста који се јављају код *IoT* уређаја, а један од њих је небезбедно бежично ажурирање. Такмичар може доћи у поседство фајла у ком се налази енкриптован фирмвер. Кључ за декрипцију фирмвера често се, као и у овом случају, чува на уређају и то у постојаној меморији (енг. *non volatile storage, nvs*). У задатку овај део флеш меморије није енкриптован, па се кључ за декрипцију фирмвера може компромитовати, што је озбиљан безбедносни пропуст. Када се приступи кључу, могуће је декриптовати фирмвер, а корак након тога је реверзно инжењерство фирмвера написаног у програмском језику *Rust*. Програм над којим је потребно применити технике реверзног инжењерства представља игрицу - лавиринт - а циљ је да такмичар спроведе играча од улаза до излаза из лавиринта, управљајући играчем уз помоћ 4 дугмета на уређају. При покретању задатка, на конзоли се,

између осталог, исписује линк са ког се преузима фирмвер за бежично ажурирање (Сл. 8). Након тога, на конзоли се приказује и партициона шема уређаја (Сл. 9).



Сл. 8. Изглед серијске конзоле након покретања четвртог задатка

Name	Type	SubType	Offset	Size
nvs	data	nvs	0x9000	0x4000
otadata	data	ota	0xd000	0x2000
phy_init	data	phy	0xf000	0x1000
factory	app	factory	0x10000	2M

Сл. 9. Партициона шема

Након што је енкриптовани фирмвер преузет, такмичар треба да га декриптује, за шта је потребан кључ, а очигледно место за складиштење кључа је постојана меморија, чија се почетна адреса (0x9000) и величина (0x4000) могу пронаћи у партиционој шеми. Наредбе за читање садржаја постојане меморије су следеће:

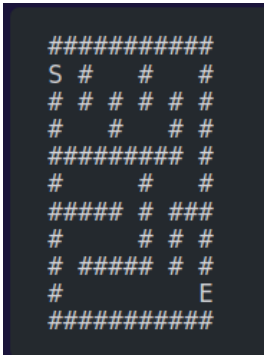
```
esptool.py --port /dev/ttyACM0 read_flash 0x9000 0x4000 nvs.bin
```

```
python ~/esp/esp-
idf/components/nvs_flash/nvs_partition_tool/nvs_tool.py nvs.bin
-d minimal
```

За декрипцију фирмвера је потребно знати и која је криптографска шема коришћена за енкрипцију фирмвера. Међутим, *Espressif* има свој алат за енкрипцију фирмвера: *esp_encrypted_img*. У *readme* фајлу пројекта се може пронаћи тачна команда за декрипцију фирмвера, која као аргументе узима енкриптован фирмвер и приватни кључ:

```
python esp_enc_img_gen.py decrypt <output_encrypted.bin>
<key_file.pem> <decrypted_output.bin>
```

За реверзно инжењерство фирмвера може се користити неки од уобичајених алата, попут програма *Ghidra*. По завршетку анализе, *Ghidra* генерише велики број функција, а даљим прегледом кода уочава се функција *start_task04*. Структура решења је заснована на лавиринту димензија 11x11, што се може закључити из чињенице да га чини 121 карактер (Сл. 10). Карактер *S* означава почетну позицију, док *E* означава крај. Декомпајлирани код региструје догађаје клика на дугме, који се касније користе за навигацију кроз лавиринт (Сл. 11). Након доласка до излаза из лавиринта исписује се заставица.



Сл. 10. Лавиринт

```
LAB_42006fac:
  uStack_10 = esp_idf_hal::task::notification::Notification::new();
  uVar4 = esp_idf_hal::task::notification::Notification::notifier(&uStack_10);
  uVar5 = esp_idf_hal::task::notification::Notification::notifier(&uStack_10);
  uStack_c = uVar5;
  uVar6 = esp_idf_hal::task::notification::Notification::notifier(&uStack_10);
  uStack_8 = uVar6;
  uVar7 = esp_idf_hal::task::notification::Notification::notifier(&uStack_10);
  uStack_4 = uVar7;
  iVar8 = esp_idf_hal::gpio::PinDriver<T,MODE>::subscribe_nonstatic(pcStack_50,uVar4);
  iVar9 = iVar8;
  if ((iVar8 != 0) ||
     (iVar9 = esp_idf_hal::gpio::PinDriver<T,MODE>::subscribe_nonstatic(pcStack_54,uVar5),
     iVar9 != 0)) goto LAB_42007328;
  iVar9 = esp_idf_hal::gpio::PinDriver<T,MODE>::subscribe_nonstatic(ppcStack_58,uVar6);
```

Сл. 11. Декомпајлирани код

VII. ЗАКЉУЧАК

На основу функционалних карактеристика развијеног хардверског изазова, као и анализе образовних аспеката појединачних задатака, може се закључити да је полазна хипотеза у значајној мери потврђена. Изазов омогућава такмичарима да се на практичан начин упознају са типичним рањивостима *IoT* уређаја, што је потврђено кроз њихову интеракцију са системом и кроз поређење са реалним сценаријима. С обзиром на то да је ово први хардверски изазов који се појавио на такмичењу у Србији, задаци стављени пред такмичаре били су нешто лакши од оних који се јављају на такмичењима вишег ранга (рецимо, Европско такмичење у етичком хаковању). Додајући томе и ограничен буџет за израду хардвера, постављени задаци не обухватају напредније нападе попут напада споредним каналима (енг. *side channel attack*) и инјекције грешака (енг. *fault injection*), који би захтевали и додатан хардвер - осцилоскоп, па чак и развој додатног уређаја за инјекцију сметњи. Једна од уочених мана је непостојање дугмета за ресет уређаја, па, када такмичар у току рада жели да покрене неки други задатак, мора да искључи напајање и укључи уређај поново, што ће у наредним верзијама бити унапређено додатком ресет дугмета. Будуће истраживање и рад у овој области обухватиће развој рањивих уређаја и додатних алата за експлоатацију, како би било могуће вршити експлоатацију споредних канала и вршити инјекцију грешака. Поред тога, планира се проширење истраживања на друге платформе и архитектуре ван *ESP32-S3* окружења, као и на шири спектар хардверских и софтверских рањивости на *IoT* и сродним уређајима.

Литература

[1] I. Grobelna, M. Grobelny, and G. Bazydło, "User awareness in IoT security. A survey of Polish users," *AIP Conf. Proc.*, 2018. Available: https://doi.org/10.1063/1.5079136

[2] J. M. Haney, S. M. Furman, and Y. Acar, *User Perceptions of Smart Home Privacy and Security*, Research Report, 2020. Available: https://doi.org/10.6028/nist.ir.8330

[3] *What is Shodan? – Shodan Help Center*, Shodan, n.d. [Online]. Available: https://help.shodan.io/the-basics/what-is-shodan

[4] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in *Proc. 10th Int. Conf. Frontiers Inf. Technol.*, 2012. Available: https://doi.org/10.1109/fit.2012.53

[5] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, 2019. Available: https://doi.org/10.1016/j.future.2019.02.050

[6] *OWASP Internet of Things Project*, OWASP, n.d. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project

[7] M. Stanislav and T. Beardsley, *HACKING IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities*, 2015.

[8] P. Lea, *IoT and Edge Computing for Architects: Implementing Edge and IoT Systems from Sensors to Clouds with Communication Systems, Analytics, and Security*. Packt Publishing, 2020.

[9] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of CAN bus security challenges," *Sensors*, vol. 20, no. 8, p. 2364, 2020. Available: https://doi.org/10.3390/s20082364

[10] S. Jafarnejad, L. Codeca, W. Bronzi, R. Frank, and T. Engel, "A car hacking experiment: When connectivity meets vulnerability," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2015. Available: https://doi.org/10.1109/glocomw.2015.7413993

[11] D. S. Dawoud and P. Dawoud, *Serial Communication Protocols and Standards*. River Publishers, 2022.

[12] N. Robson *et al*., "Electrically programmable fuse (eFuse): From memory redundancy to autonomic chips," in *Proc. IEEE Custom Integrated Circuits Conf.*, 2007. Available: https://doi.org/10.1109/cicc.2007.4405850

[13] Abhilasha, "EFuse: The basics," *Medium*, Oct. 21, 2023. [Online]. Available: https://abhi24.medium.com/efuse-the-basics-6fcf1dbf6def