Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Útoky na hardvér pomocou indukovania chýb

Bakalárska práca

2023
Dennis Vita

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Útoky na hardvér pomocou indukovania chýb

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	FMFI.KI - Katedra informatiky
Školiteľ:	RNDr. Richard Ostertág, PhD.

Bratislava, 2023
Dennis Vita

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Dennis Vita
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Útoky na hardvér pomocou indukovania chýb
Attacking hardware using fault injection

Anotácia: Pomocou externých faktorov je možné ovplyvniť správanie hardvéru až do takej miery, že nekorektne vykoná nejakú operáciu. Takýmto spôsobom je možné zmeniť hodnoty prečítaných príznakov alebo cieľovú adresu skokov, či výsledky iných operácií. Toto neštandardné správanie je potom môžné využiť na cielený útoku na zvolený hardvér. Medzi tieto externé vplyvy patria napríklad zmena napájania (obvykle podpätie) alebo elektromagnetické rušenie.

Cieľom práce je zozbierať, podrobne popísať (vrátane potrebných zapojení, nástrojov, ...) a prakticky vyskúšať rôzne útoky. Napríklad:
• prečítanie obsahu pamäte s firmvérom aj pri zapnutej ochrane pred čítaním (code read protection)
• rôzne možnosti ovplyvnenia vykonávanie algoritmov (napríklad ovplyvnenie podmienených skokov, výsledkov matematických a logických operácií)
Výsledkom práce by malo byť aj overenie ako veľmi záleží na presnosti a časovaní pri takýchto útokoch na rôzny hardvér (napríklad Arduino, STM32) a či treba pri implementácii využiť rýchlosť a presnosť FPGA.
Ďalším cieľom práce je príprava nových úloh do súťaží typu Capture The Flag (CTF). Úlohy by pozostávali zo zadania, vzorového riešenia (zapojenie, program, obrázky, prípadne aj video riešenie).

Vedúci: RNDr. Richard Ostertág, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 30.09.2022

Dátum schválenia: 05.10.2022 doc. RNDr. Dana Pardubská, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Pri návrhu a implementácii softvéru sa často predpokladá, že hardvér, na ktorý bude softvérové dielo nasadené bude jednotlivé inštrukcie vykonávať bezchybne a presne tak, ako výrobca daného hardvéru uvádza. Za bežných podmienok je väčšinou tento predpoklad skutočne naplnený. Vplyv externých fyzikálnych faktorov, ako napríklad prudká a krátka zmena napätia, elektromagnetické rušenie a mnoho ďalších, však môže mať za následok chybné vykonanie aktuálnej inštrukcie a tým spôsobiť nedefinované správanie bežiaceho procesu. Takýmto spôsobom možno cielene vyvolať konkrétnu chybnú operáciu v správnom čase a zaútočiť tak na dané zariadenie, napríklad s účelom obísť bezpečnostný mechanizmus. Takýto typ útokov na hardvér nazývame indukovanie chýb (angl. fault injection).

Hlavným cieľom tejto práce je práve overenie či aj lacný hardvér môže byť použitý ako zdroj indukovania chyby, resp. čo ním možno dosiahnuť. Následnou analýzou potom určiť, aké typy chýb na úrovní jednotlivých inštrukcií možno dosiahnuť a ako presné časovanie je potrebné. Zároveň vyskúšame tieto znalosti aplikovať na štandardné programy písané v jazykoch na vyššej úrovni (napríklad C).

Kapitola 1

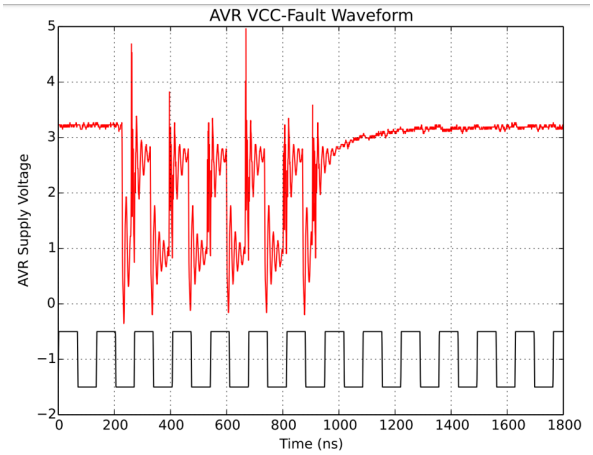
Techniky a princípy a ciele útokov

V tejto kapitole stručne zhrnieme najčastejšie techniky používané na indukovanie chýb na hardvéri a vysvetlíme princípy týchto techník.

Základným predpokladom úspešného útoku pomocou indukovania chýb je fyzický prístup k zariadeniu, ktoré je cieľom útoku. Väčšina takýchto útokov vyžaduje miernu modifikáciu hardvéru, aby bolo možné ovplyvniť jeho činnosť, napríklad odstránenie niektorých komponentov, ktoré by mohli útok sťažiť, prípadne úplne znemožniť. Ideálnou možnosťou pre útočníka je, ak môže komponent vybrať a prevádzkovať ho vo svojom prostredí. V niektorých prípadoch sa dokonca podarilo zariadenie poškodiť na trvalo tak, aby chyba spôsobovala cielený účinok v systéme, do ktorého bolo následne opäť nasadené. Častými obeťami takýchto útokov sú vnorené (angl. embedded) zariadenia, väčšinou ovládané pomocou MCU (Micro Controller Unit), známe aj pod názvom mikrokontrolér. Práve pri vnorených zariadeniach sú zvyčajne splnené všetky vyššie uvedené predpoklady.

1.1 Zmena napätia

Jednou z najjednoduchších a veľmi často využívaných techník indukovania chýb je zmena napätia, obvykle podpätie (angl. power glitch). Princíp takéhoto útoku spočíva v skonštruovaní obvodu, ktorý bude mať kontrolu nad napájaním zariadenia, na ktoré chceme útočiť a vo vhodných časových intervaloch veľmi krátkej dĺžky (rádovo stovky nanosekúnd), vyvolať prudkú zmenu napätia. Takáto manipulácia môže spôsobiť nedefinované správanie na napájanom zariadení, najčastejšie možno očakávať, že ovplyvní aktívne časti hardvéru, napríklad obvody na procesore vykonávajúce jednotlivé inštrukcie. Častými symptómami takéhoto útoku sú napríklad preskočenie inštrukcie, nekorektné vyhodnotenie podmieneného skoku, chybný výsledok aritmetickej, či logickej operácie a pod. Najdôležitejšia časť takéhoto útoku je správne načasovanie a dĺžka intervalu zmeny napätia. Pokiaľ je tento časový úsek zmeny napätia príliš dlhý (viac



Obr. 1.1: Synchronizácia zmeny napätia s hodinami zariadenia. Červenou farbou je znázornená hodnota napätia v čase. V spodnej časti grafu (čierna farba) sú hodinové impulzy zariadenia [8].

ako 2-3 mikrosekundy), je veľká pravdepodobnosť, že nastane fatálne zlyhanie a následný reštart zariadenia. Práve požadovaná presnosť časovania je parameter, ktorý výrazne vplýva na technickú náročnosť útoku. V niektorých prípadoch

Pre implementáciu útoku využívajúceho techniku zmeny napätia sú zvyčajne potrebné dve základné súčasti. Jednou je obvod, ktorý dokáže dynamicky manipulovať s napájaním zariadenia, na ktoré cielime a druhým je generátor riadiacich impulzov pre tento obvod, ktorý dokáže generovať signály s dostatočnou presnosťou. V závislosti od zložitosti a požadovaných parametrov týchto dvoch komponentov sa odvíjajú aj náklady a náročnosť takéhoto útoku. V porovnaní s inými technikami však zmena napätia patrí k menej náročným na implementáciu a je preto veľmi často používaná.

Príklad jednoduchej implementácie takéhoto útoku môže byť napríklad použitie tranzistora, ktorým vieme spínať napájanie na mikrokontroléri. Takýmto spôsobom vieme na krátky okamih vypnutím tranzistora vyvolať podpätie na mikrokontroléri a následným zopnutím zase vrátiť napätie do bežného stavu, čo môže vyvolať chybné vykonanie jednej alebo viacerých nasledujúcich inštrukcií [5].

Útok založený na podobnom, ale mierne zložitejšom princípe využíva tzv. „Corwbars“ obvod. Princíp takéhoto obvodu spočíva v tom, že okrem hardvéru, zdroj napájania paralelne zapojíme cez tranzistor do „skratu“, ktorý vieme spínať pomocou tohoto tranzistora. Generovaním impulzov do tranzistora vieme na veľmi krátky čas vyskratovať napájací zdroj, čím spôsobíme prudké zmeny v napätí na mikrokontroléri. Aby bolo možné presnejšie zacieliť na konkrétnu časť vykonávaného kódu bolo ovládanie tohoto obvodu synchronizované s externými hodinami cieľového zariadenia, znázornené na obrázku 1.1 [8].

Pre použitie takýchto obvodov je potrebné zabezpečiť, exaktné časovanie generovaných riadiacich impulzov. To je možné zabezpečiť napríklad laboratórnym zdrojom

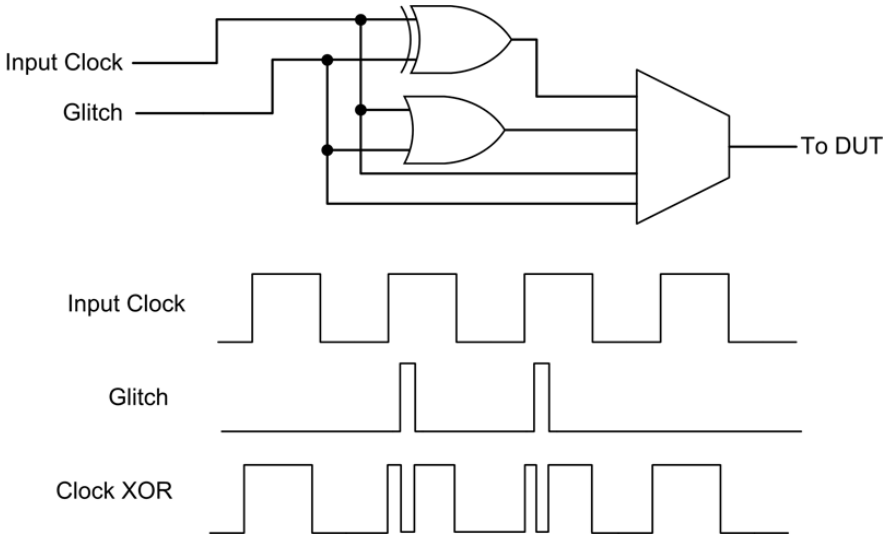
alebo použitím špecializovaného hardvéru, napríklad open source platformu ChipWhisperer [10]. Takéto riešenia však môžu vyžadovať netriviálne náklady. Otázkou je, či aj lacnejšie riešenie, napríklad mikrokontrolér ATMega328P, ktorý je súčasťou populárnych vývojových dosiek Arduino, dokáže dosiahnuť rovnako dobré výsledky. Jedným z cieľov tejto práce je práve vyskúšať takéto typy útokov pomocou mikrokontroléra ATMega328P.

1.2 Porucha hodín

Ďalšou často používanou technikou indukovania chýb je manipulácia taktovacích impulzov prichádzajúcich z externých hodín do zariadenia (angl. clock glitch). Takouto manipuláciou možno vytvoriť nepravidelný taktovací signál, ktorý môže spôsobiť nekorektné správanie hardvéru, ním riadeným. Riadiaca jednotka zariadenia obvykle reaguje na nábehovú (niekedy dobehovú) hranu signálu prichádzajúceho od hodín zariadenia. Napríklad pridanie hrán do takto generovaného signálu môže mať za následok, že sa začne vykonávať ďalšia inštrukcia skôr ako sa predošlá inštrukcia stihla korektne dokončiť, čo pri správnom načasovaní môže spôsobiť cielený efekt na vykonávajúci sa program. Aby bol takýto útok úspešný je potrebné, aby bola riadiaca jednotka zariadenia priamo taktovaná externým oscilátorom. Mnoho zariadení pomocou PLL (Phase Lock Loop) obvodu odvádza z externého signálu od hodín interný, ktorý má rádovo vyššiu frekvenciu. Proti takýmto zariadeniam preto útok touto technikou pravdepodobne nebude účinný[8].

Útok poruchou hodín môže byť realizovaný napríklad skonštruovaním kombinačného obvodu, ktorý pomocou logických hradiel najčastejšie AND, OR a XOR skladá rôzne výstupné signály zo vstupných. Vstupom do takéhoto obvodu môžu byť rôzne zdroje impulzov napríklad oscilátory s rôznymi frekvenciami, laboratórne zdroje, generátory signálov a ďalšie. Cieľom je rôznymi spôsobmi modulovať výstupný taktovací signál tak, aby v želaných okamihoch mal nesprávny tvar (nepravidelné takty, nesprávne hrany, vyššia frekvencia) a spôsobil tak poruchu na cieľovom zariadení. Pomocou logických hradiel možno vytvoriť rôzne vzorky signálu a následne napríklad využitím multiplexora automatizovane prepínať medzi jednotlivými výstupmi a tým dynamicky meniť výstupný hodinový signál. Na obrázku 1.2 je znázornená ukážka schémy takéhoto obvodu [6]. Takýto obvod dokonca často nie je potrebné zostaviť fyzicky, ale možno použiť aj programovateľné hradlové pole (FPGA), čo značne zjednoduší implementáciu útoku.

Poruchy, ktoré vieme touto technikou spôsobiť môžu byť rôzne, ale najčastejšie sa týkajú toku riadenia a dát, keďže hodiny generujú signál pre činnosť riadiacej jednotky procesora. Príkladmi takýchto porúch sú vynechanie inštrukcie, nekorektné načítanie



Obr. 1.2: Schéma obvodu pre generovanie nepravidelného signálu. Vo vrchnej časti je znázornená schéma obvodu, v spodnej sú rôzne výstupy jednotlivých kanálov multiplexora [6].

dát z pamäte a nekorektný zápis do registra PC (Program Counter), čo môže spôsobiť nesprávny skok vrámci vykonávaného programu. Konkrétna implementácia takéhoto útoku na AVR mikrokontrolér z rodiny ATMega, do ktorej patrí aj nami zvolený ATMega328P (bližšie popísaný v kapitole 2) ukázala, že možno týmto spôsobom vyvolať viaceré z vyššie spomenutých porúch [3].

1.3 Elektromagnetické rušenie

Treťou známou technikou indukovania chýb je elektromagnetické rušenie alebo EMFI (Electromagnetic Fault Injection). Vplyvom elektromagnetického poľa možno narúšať fungovanie hardvéru na cieľovom zariadení a tým ovplyvňovať jeho činnosť. Takýto typ útokov zvyčajne vyžaduje XYZ pohyblivú dosku riadenú počítačom, ktorá dokáže hýbať a s dostatočnou presnosťou nastaviť pozíciu antény, ktorá je zdrojom elektromagnetického rušenia. Generátorom impulzov do tejto antény následne možno v daných časových okamihoch spôsobovať želané rušenie [7].

Výhodou tejto techniky je možnosť s istou presnosťou (závisí od parametrov použitej elektroniky) útočiť na konkrétne časti obvodov na cieľovom čipe, napríklad pamäť, registre, či ALU. Nevýhodou je väčšia technická náročnosť útoku a obvykle drahší hardvér.

Existuje mnoho ďalších techník indukovania chýb na hardvéri (napríklad laserové vypaľovanie), ktoré dokážu byť oveľa presnejšie a spoľahlivejšie. Ich implementácia je však prudko náročnejšia, vyžaduje vysokú úroveň technickej odbornosti a veľmi vysoké finančné náklady. Analýza týchto techník preto presahuje rámec tejto práce a nebudeme

sa im podrobnejšie venovať.

1.4 Ciele útokov

Najčastejším cieľom útokov pomocou indukovania chýb je obídenie bezpečnostného mechanizmu. Jednoduchým príkladom je ovplyvnenie jednoduchej kontroly podmienky v programe vyvolaním chyby, ktorá spôsobí skočenie programu do nesprávnej vetvy, čo z vyššieho pohľadu môže mať efekt nesprávneho vyhodnotenia tejto podmienky. Zložitejšie útoky môžu cieliť na nesprávne vykonanie aritmetickej, či logickej operácie alebo nekorektné prečítanie dát z pamäte a následne chybné rozhodnutie algoritmu, čo spôsobí cielené obídenie daného bezpečnostného mechanizmu. Konkrétnym príkladom takýchto cieľov môže byť získanie neoprávneného prístupu k službám alebo dátam zariadenia, prečítanie obsahu pamäte s firmvérom aj pri zapnutej ochrane pred čítaním, útok na zavádzací softvér počítača, aby naštartoval systém z nepovoleného média a množstvo ďalších scenárov.

Ďalšia kategória cieľov týchto útokov sú zariadenia implementujúce kryptografické schémy. Tieto schémy sú často veľmi náchylné na implementačné chyby a preto dopady takýchto útokov môžu byť fatálne. Pomocou indukovania chýb možno napríklad prinútiť zariadenie, aby použilo nulový vektor ako kľúč pre symetrickú šifru. Potom už je jednoduché takto zašifrované dáta priamo dešifrovať. O niečo sofistikovanejšia, ale o to účinnejšia, metóda útokov na kryptografické konštrukcie je tzv. DFA (Differential Fault Analysis). Princíp tejto metódy spočíva v indukovaní chýb na kryptografický algoritmus a porovnávaním korektných a nekorektných výstupov určiť aké inštrukcie a dáta sa v zariadení spracovávaj. Takouto analýzou je následne možné napríklad určiť použitý kľúč v symetrickej šifre alebo dokonca efektívne faktorizovať verejný modulus RSA schémy. Praktická úspešnosť týchto útokov bola demonštrovaná proti implementáciám schém AES a RSA na vnorených zariadeniach [4].

1.5 Zraniteľnosti a obranné mechanizmy

Samotná implementácia a následná úspešnosť útoku často závisí aj od samotného algoritmu, na ktorý je útok cielený, a spôsobu jeho implementácie. Rôzny kód môže mať rôzne druhy zraniteľností a od toho sa odvíja ako veľmi môže byť náročná technická realizácia útoku. Niekedy nie je pre dosiahnutie cieleného efektu potrebné mieriť na jednú konkrétnu chúlostivú operáciu, ale stačí pokiaľ je výsledok série operácií nesprávny. Príkladom kódu, ktorý je náchylný voči chybám spôsobeným vplyvom prostredia je postupné inkrementovanie premennej v rámci cyklu, alebo séria viacerých priradení do rovnakej premennej počas výpočtu. Kedy presne nastane chyba neovplyvní výsledný

efekt útoku, za predpokladu, že chyba nespôsobí fatálne zlyhanie cieľového zariadenia. Pre tento typ útokov je často postačujúci lacný hardvér a nie je väčšinou potrebná synchronizácia zdroja chýb a cieľa.

V iných prípadoch je potrebné cieliť na konkrétnu operáciu, čo zvyčajne vyžaduje vyššiu presnosť a synchronizáciu zariadení počas útoku. Príkladom takejto zraniteľnosti je zneužitie jednoduchej optimalizácie kompilátora [9]. Kompilátor pri prekladaní cyklu, v ktorom sa postupne dekrementovala iterovaná premenná a postupne čítal vybraný úsek pamäte, použil inštrukciu BNE (Branch if Not Equal) miesto BLT (Branch if Less Than). Takáto optimalizácia je z teoretického pohľadu správna, keďže nemení význam algoritmu a test na nulu možno efektívnejšie vykonať ako všeobecné porovnanie. Keď v takto upravenom programe docielime vynechanie tejto inštrukcie práve vtedy, keď má dôjsť k ukončeniu cyklu, spôsobíme tým, že takýto program bude pokračovať v cykle až kým nepretečie hodnota iterovanej premennej. Takýto útok bol demonštrovaný a autorovi sa podarilo prečítať firmvér na mikrokontroléri napriek tomu, že k nemu nemal oprávnený prístup. V prípade, že by kompilátor použil inštrukciu BLT, by bolo potrebné pravidelne v každej iterácií spôsobiť takúto chybu, čo je pre útočníka náročnejšie [9].

Existuje viacero obranných mechanizmov, ktoré dokážu takéto útoky skomplikovať, niekedy až znemožniť. Medzi aktívne mechanizmy patrí detekcia takýchto útokov s využitím špecializovaného hardvéru, alebo softvéru a následne spustiť bezpečnostné opatrenie, napríklad reštart zariadenia. Aktívna detekcia útokov zvyšuje výkon zariadenia, čo najmä v prípade vnorených systémov môže byť problematické riešenie. Jednoduchší, ale niekedy postačujúci, je pasívny spôsob obrany, napíklad úprava kódu tak, aby bol menej chúlostivý voči chybám. Pridanie náhodných oneskorení do programu môže sťažiť situáciu útočníkovi, ktorý potrebuje presne načasovať útok na konkrétnu operáciu. Ďalšími opatreniami môže byť nepoužívanie binárnych premenných pri vetvení programu a rovnako nepoužívanie triviálnych konštánt (0, súvislý vektor jednotiek). Nastavenie konkrétnej netriviálnej hodnoty (0xAA, 0xB9, ...) vplyvom chyby je náročnejšie, niekedy zanedbateľne nepravdepodobné a možno tak takémuto útoku zabrániť.

Kapitola 2

Hardvér

V tejto kapitole popíšeme a rozoberieme podstatné aspekty zvoleného hardvéru. Pri útokoch pomocou indukovania chýb je často potrebná vysoká presnosť časovania v stovkách až desiatkach nanosekúnd. Lacnejší hardvér často takúto presnosť nie je schopný dosiahnuť. Hlavným cieľom tejto práce je overiť túto skutočnosť a zistiť akú presnosť časovania dokáže takýto hardvér poskytnúť. Finančné náklady na všetky súčiastky použité pri implementovaných útokoch sa pohybujú rádovo v desiatkach eúr.

2.1 ATMega328P

Najdôležitejším zariadením pre účely tejto práce je Atmel mikrokontrolér s osem bitovou architektúrou AVR, dnes vlastnený firmou Microchip Technology Inc. Tento mikrokontrlér je súčasťou viacerých modelov vývojových dosiek Arduino, ktoré sú často využívané pri vývoji jednoduchých vnorených zariadení. Zároveň zvykne byť použitý aj pri nasadení skutočných produktov, najmä vnorených systémov s nízkou spotrebou. Mikrokontrolér ATMega328P bol aj preto zvolený ako cieľom útokov implementovaných v tejto práci, ale hlavným dôvodom bola jeho veľmi nízka cena pohybujúca sa okolo desať eúr. Techniky útokov použité v tejto práci môžu s nezanedbateľnou pravdepodobnosťou trvalo poškodiť cielený hardvér, preto sme sa rozhodli použiť viacej rovnakých čipov. Konkrétne boli testované štyri exempláre formátu ATMega328P-PU z dvoch (po dvojiciach) rôznych sérií výroby. Okrem toho bol použitý aj ako zdroj útokov generujúci raidiace impulzy do rôznych obvodov využívajúcich niektoré techniky spomenuté v kapitole 1, konkrétne ako súčasť dosky Adrunino Nano.

2.1.1 Kľúčové vlastnosti

Zvolený formát čipu ATMega328P-PU je v púzdre THT (Through Hole Technology), čo znamená, že piny na čipe sú vyvedené vo forme „kolíkov“ a je možné čip zapojiť bez potreby pájkovania pomocou nepájivého kontaktného poľa (angl. solderless breadbo-

ard). Takéto zapojenie potom možno jednoducho a bezprostredne modifikovať počas experimentovania. Ďalšou vlastnosťou je možnosť pripojenia externého oscilátora s frekvenciou maximálne 16 MHz, čo znamená väčšiu flexibilitu nastaviteľných parametrov jednotlivých útokov. Na čipe sa nachádza aj interný oscilátor s frekvenciou 8 MHz a možno ho využiť miesto zapojenia s externým [1].

Čo sa týka softvéru, pre programovanie mikrokontroléra ATMega328P potrebujeme kompilátor nízko-úrovňového jazyka (napríklad C) pre architektúru AVR. Nahratie programu do trvácnej flash-pamäte na mikrokontroléri možno zabezpečiť napríklad pomocou ISP (In-System Programming) programátora. Pre tento účel, je potrebné, aby na mikrokontroléri bol prítomný zavádzací program (angl. bootloader), ďalej len zavádzač, ktorý dokáže komunikovať s ISP programátorom. Jednoduchý a automatizovaný spôsob pre vykonanie týchto operácií poskytuje aj integrované vývojové prostredie (IDE) Arduino IDE, ktoré sme sa rozhodli použiť. Arduino IDE interne používa kompilátor AVR-GCC a ISP programátor AVRDUDE, ktoré sú voľne dostupné aj samostatne.

2.1.2 Architektúra AVR

Mikrokontrolér ATMega328P implementuje osem bitovú architektúru AVR. Niektoré aspekty tejto architektúry sú dôležité pre pochopenie niektorých častí kódu písaných v asembleri v rámci tejto práce, preto ich stručne opíšeme. AVR je zaraďovaná do triedy RISC (Reduced Instruction Set Computer) architektúr [2]. Poskytuje tridsaťdva (R0 – R31) osembitových všeobecných registrov, z nich niektoré so špeciálnym významom pri vybraných inštrukciách. V tejto práci používame tieto registre len v základnom, všeobecnom význame. Pre prístup do pamäte slúžia dve základné inštrukcie LD (Load), ST (Store) a ich varianty.

Ovládanie GPIO (General Purpose Input Output) pinov na mikrokontroléri je zabezpečené pomocou špeciálnych vstupno-výstupných registrov. Každý pin má priradený jeden zo štyroch tzv. portov (ozn. A – D) a v rámci portu jeden z ôsmich bitov (0 – 7). Každý pin je potom ovládaný troma registrami DDRX, PINX a PORTX, kde X je písmeno označujúce port priradený pinu. Bit priradený pinu potom určuje index konkrétneho bitu (0 označuje najmenej významný bit) v týchto troch registroch (rovnaký index pre každý register), ktorým možno ovládať daný pin. Význam jednotlivých bitoch v registroch je potom nasledovný: DDR (Data Direction register) určuje smer toku dát, pokiaľ je príslsný bit v tomto registri nastavený na 0, pin je vo vstupnom móde, pokiaľ je nastavený na 1, pin je vo výstupnom móde. Register PIN má význam vo vstupnom móde a v danom bite udržiava hodnotu bitu zo vstupu na danom pine. Register PORT má naopak hlavný význam vo výstupnom móde a hodnota v príslušnom bite je výstupom na priradenom pine. Tieto vstupno-výstupné registre možno adresovať priamo pomocou vyhradeného adresného priestoru v pamäti (memory-mapped I/O), alebo s

Tabuľka 2.1: Ukážka nastavenia vstupno-výstupných registrov v architektúre AVR.

Inštrukcia	Komentár
sbi 0x04, 5	nastavenie bitu 5 na 1 v registri DDRB
sbi 0x05, 5	nastavenie bitu 5 na 1 v registri PORTB
cbi 0x05, 5	nastavenie bitu 5 na 0 v registri PORTB

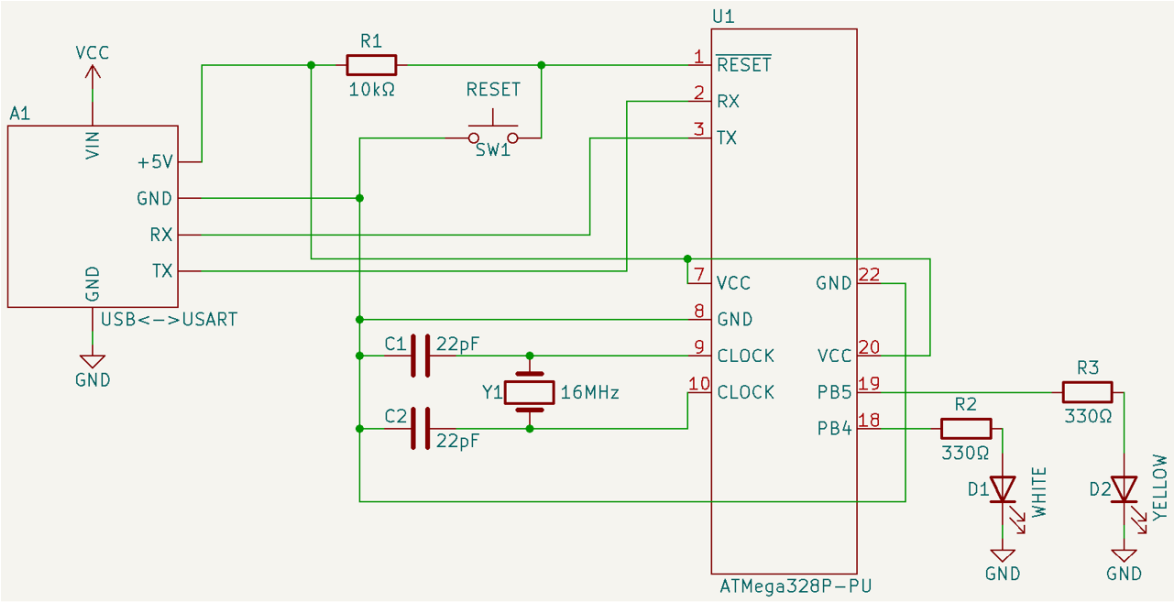
využitím špeciálnych inštrukcií (port-mapped I/O).

Pre lepšie porozumenie uvedieme príklad. Chceme nastaviť GPIO pin 13 ako výstupný a na výstupe chceme periodicky prepínať medzi logickými hodnotami 0 a 1, napríklad s účelom blikať LED diódou. GPIO pin 13 má priradený port B, bit 5. Potrebujeme preto nastaviť v registeri DDRB bit 5 na 1, čím nastavíme pin do výstupného módu. Následné prepínaním medzi hodnotami 0 a 1 na bitu 5 v registri PORTB dosiahneme želaný efekt na GPIO pine 13. Nstavenie hodnoty konkrétneho bitu vo vstupno-výstupnom registri možno docieliť napríklad pomocou vstupno-výstupných inštrukcií SBI (Set Bit in I/O Register) a CBI (Clear Bit in I/O register), ktoré nastavia daný bit na 1 resp. 0 [2]. Tieto inštrukcie používajú vstupno-výstupné adresy daných registrov, ktoré v našom prípade sú 0x04 (DDRB) a 0x05 (PORTB) [1]. Ukážky jednotlivých inštrukcií v asembleri sa nachádzajú v tabuľke 2.1. V prípade, že by sme chceli hodnoty v registroch nastaviť pomocou inštrukcií LD a ST (prípadne ich variánt), použili by sme adresy týchto registrov v mape pamäti – 0x24, resp. 0x25 [1].

2.1.3 Základné zapojenie mikrokontroléra

V tejto časti predstavíme zapojenie mikrokontroléra ATMega328P v THT púzdre, ktoré neskôr využijeme pri implementácií vybraných útokov v kapitole 3. Zapojenie spočíva v pripojení základných súčiastok, ktoré zabezpečujú základné funkcie mikrokontroléra, konkrétne pripojíme zdroj napätia, externý oscilátor, tlačidlo pre resetovanie, prevádzanie signálu medzi rozhraniami USB a USART pre komunikáciu s počítačom a dve LED diódy, ktoré možno použiť pre výstupné signály. Zoznam a popis všetkých použitých súčiastok:

- ATMega328P – THT púzdro, umožňuje použiť kontaktné nepájivé pole
- prevodník medzi USB a USART – použitý modul s čipom CP2102, na doske je vyvedený výstup z napájania USB
- kontaktné nepájivé pole – umožňuje jednoducho prepájať súčiastky bez potreby spájkovania
- spínacie tlačidlo – uzemní RESET pin, čo spôsobí reset mikrokontroléra



Obr. 2.1: Schéma zapojenia mikrokontroléra ATMega328P.

- 16MHz kryštál – pre zapojenie externého oscilátora
- kondenzátory – 2 kusy, kapacita 22pF, záťažové kondenzátory ku kryštálu
- rezistory – 2-krát 330Ω k LED diódam, 1-krát 10kΩ – zdvíhací odpor RESET pinu
- LED diódy – 2 kusy, použili sme rôzne farby (žltá a biela)

Súčiastky zapojíme podľa schémy na obrázku 2.1. Prevodník medzi USB a USART možno okrem bežnej sériovej komunikácie použiť aj na naprogramovanie mikrokontroléra pomocou ISP programátora, za predpokladu, že mikrokontrolér obsahuje zavádzač. Pred nahratím nového programu do mikrokontroléra je potrebné stlačiť zapojené tlačidlo a tým ho resetovať. Po opätovnom zapnutí sa na krátky čas (niekoľko sekúnd) spustí zavádzač na mikrokontroléri, čo umožňuje komunikáciu s ISP programátorom. Komunikáciu so zavádzačom je potrebné iniciovať v rámci tohoto časového okna, v opačnom prípade zavádzač spustí program nahratý v pamäti mikrokontroléra a bude potrebné pre komunikáciu so zavádzačom mikrokontrolér opätovne resetovať.

Kapitola 3

Implementácia vybraných útokov

V tejto kapitole podrobne popíšeme vybrané útoky na konkrétny hardvér, ktoré sa nám podarilo uskutočniť.

3.1 Útok na firmvér zo súťaže CTF

Ako prvé sme sa pokúsili zreprodukovať popísaný útok na firmvér, ktorý bol realizovaný v rámci súťaži CTF (Capture The Flag) [5]. Firmvér po spustení periodicky posiela cez rozhranie USART (Universal Synchronous / Asynchronous Reciever and Transmit) správu „Lock“. Po úspešnom útoku by mal poslať „tajný flag“, ktorého obsah je cieľom tejto súťaže. Potrebný hardvér pre realizáciu tohoto útoku je nasledovný:

- Arduino UNO R3 – použili sme precízny klon, obsahuje vyberateľný čip ATMega328P v THT púzdre
- Arduino Nano – taktiež precízny klon, použité na ovládania tranzistora ako zdroj indukovania chýb
- kontaktné nepájivé pole – umožňuje jednoducho prepájať súčiastky bez potreby spájkovania
- bipolárny tranzistor NPN – model 2N2222A v THT púzdre
- rezistory – použili sme 2-krát 330Ω, potrebný odpor sa môže líšiť v závislosti od použitého tranzistora
- prepojovacie kábliky typu M-M (Male to Male), niekoľko (približne 15) kusov rôznej dĺžky

Čo sa týka softvéru je potrebný ISP programátor, ktorý dokáže nahrať firmvér na mikrokontrolér ATMega328P a kompilátor jazyka C pre architektúru AVR. Použili

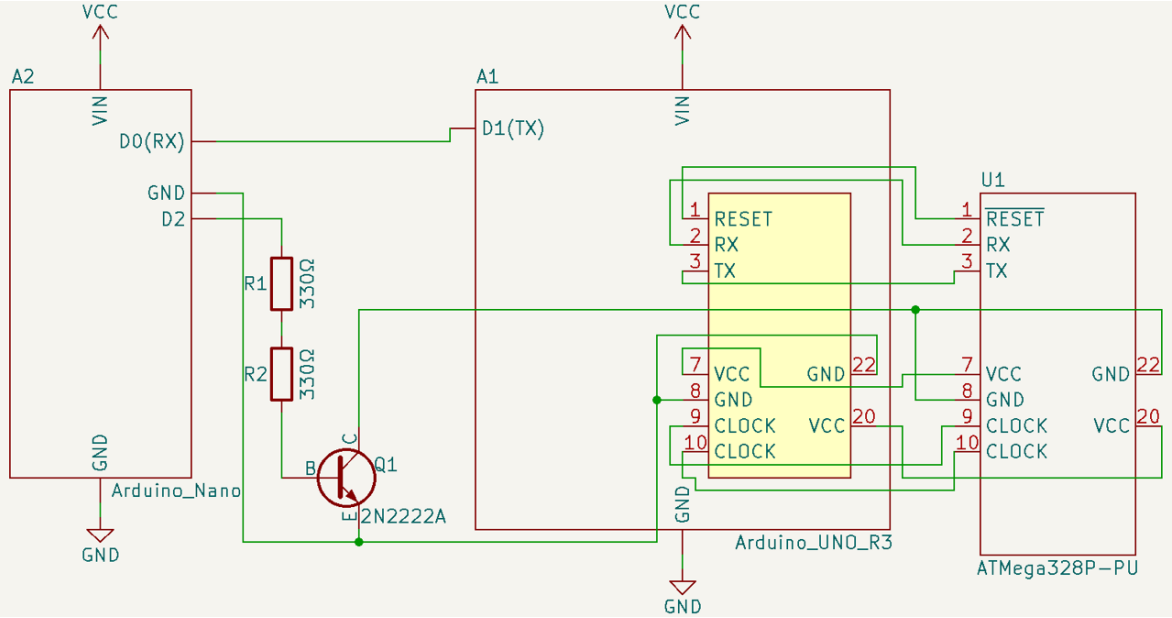
sme integrované vývojové prostredie (IDE) Arduino IDE, ktoré poskytuje automatizovaný spôsob kompilovania a nahrávania firmvéru, podporujúce aj náš ATMega328P. Firmvér, na ktorý chceme útočiť bol poskytnutý priamo v skompilovanej forme a pre nahranie takéhoto kódu sme priamo použili open source projekt AVRDUDE, ktorý interne používa aj Arduino IDE.

3.1.1 Postup útoku

Postup je zhodný s popisom pôvodného útoku [5]. Po prvé potrebujeme nahrať firmvér, na ktorý budeme útočiť na mikrokontrolér ATMega328P, ktorý je súčasťou dosky Arduino UNO. Doska obsahuje prevodník z USB na USART, čo zjednodušuje celý postup. Prevodník na doske pripojíme k počítaču a nahráme firmvér zo súťaže priamo pomocou softvéru AVRDUDE [5]. Súbor „fiesta.hex“ so skompilovaným firmvérom, ktorý je potrebné nahrať sa nachádza v prílohe 4, prevzatý zo súťaže [5].

Ďalším krokom je hardvérové zapojenie. Mikrokontrolér ATMega328P vyberieme z dosky Arduino UNO, keďže doska obsahuje stabilizátor napätia, ktorý by útok znemožňoval. Následne zapojíme tranzistor, tak aby ho bolo možné spínať pomocou Arduino Nano – emitor prepojíme so zemou na doske a bázu prepojíme s pinom, ktorý ovláda tranzistor (v našom prípade pin D2). Medzi doskami prepojíme piny zabezpečujúce sériovú komunikáciu USART smerom z UNO do Nano, aby bolo možné prečítať vypísaný „flag“, pre tento účel je potrebné prepojiť aj zem medzi doskami. Vybratý čip ATMega328P v púzdre THT umiestnime na kontaktné pole a prepojíme (káblikmi) naspäť s pôvodnou doskou Arduino UNO nasledovné piny: 1, 2, 3, 7, 9, 10 a 20. Tieto zabezpečujú základné potreby pre fungovanie mikrokontroléra – napájanie (zem týmto spôsobom neprepájame), externé hodiny (kryštál), sériová komunikácia. Piny 8 a 22 (zem), prepojíme s kolektorom tranzistora – zopnutím tranzistora pomocou ovládacieho pinu Arduino Nano, takto vieme odpájať a pripájať ATMega328P so zemou. Podrobná schéma celého zapojenia sa nachádza na obrázku 3.1.

Ďalej je potrebné naprogramovať Arduino Nano, aby pomocou ovládania tranzistora na krátko odpojilo napájanie ATMega328P od zeme a následne opäť pripojilo. Dĺžka časového intervalu, počas ktorého je tranzistor vypnutý, musí byť dostatočne dlhá, aby indukovala chybu na procesore, ale nie príliš dlhá, aby sa mikrokontrolér resetoval. Vyskúšali sme do Arduino Nano nahrať program použitý aj v pôvodnom útoku [5], ktorý bol implementovaný v prostredí Arduino IDE. Algoritmus vypne tranzistor a vykoná niekoľko prázdnych iterácií for-cyklu, po ktorom tranzistor opäť zapne. Následne sa pokúsi prečítať výstup zo sériového portu na ATMega328P. V prípade, že sa podarí prečítať „flag“, útok bol úspešný. V prípade, že sa „flag“ nepodarí prečítať, zväčší počet iterácií for-cyklu a postup zopakuje [5]. Ukážka časti kódu, ktorá ovláda tranzistor, je v algoritme 3.1.



Obr. 3.1: Schéma zapojenia útoku na firmvér zo súťaže CTF [5]. Žltý obdĺžnik znázorňuje nepájivé púzdro na doske Arduio UNO, z ktorého bol vybratý mikrokontrolér ATMega328P.

Algoritmus 3.1: Ovládanie tranzistora, ktorý spína napájanie na ATMega328P. Prevzaté zo zdrojového kódu pôvodného útoku [5].

```
int waste = 0;
digitalWrite(powerPin, LOW);
for (int i = 0; i<glitchDelay; i++){ waste++; }
digitalWrite(powerPin, HIGH);
glitchDelay += 10;
```

3.1.2 Výsledok, analýza a vylepšenie útoku

Útok bol vyskúšaný na všetky štyri čipy ATMega328P. Na dvoch z nich sa útok úspešne podaril s rovnakým podobným výsledkom ako v pôvodnom popise [5] a „flag“ sa podarilo prečítať už pri nulovom počte iterácií for-cyklu. Postačoval najkratší možný výpadok – vypnutie a okamžité zapnutie tranzistora pomocou funkcií poskytnutých prostredím Arduino IDE. Na druhých dvoch exemplároch (z inej série výroby) útok nebol úspešný a aj pri tomto najkratšom možnom výpadku sa mikrokontrolér resetoval. Rozhodli sme sa preto napätie na mikrokontroléri počas útoku podrobne analyzovať pomocou osciloskopu. Pre tento účel sme sa rozhodli ATMega328P zapojiť na kontaktom nepájivom poli bez dosky Arduino UNO. Postup tohoto zapojenia sme popísali v kapitole 2. Takéto zapojenie umožňuje meniť pasívne elektronické súčiastky v zapojení, čo poskytuje väčšiu flexibilitu vo voľbe parametrov týchto súčiastok pri analýze.

Pomocou osciloskopu sa podarilo podrobne určiť priebeh zmeny napätia na mikrokontroléri v čase s presnosťou na rádovo stovky nanosekúnd. Výsledkom bolo, že interval, počas ktorého nastalo podpätie bol príliš dlhý (približne 2 ms), čo pri útoku na 2 čipy zo štyroch spôsobilo reset mikrokontroléra. Doska Arduino Nano, ktorá ovládala tranzistor obsahuje tiež mikrokontrolér ATMega328P, s externým oscilátorom s frekvenciou 16 MHz. V kapitole 2 sme spomenuli ukážku nastavenia výstupnej logickej hodnoty na 1, resp. 0 pomocou jedinej inštrukcie SBI, resp. CBI. Obé tieto inštrukcie dokáže procesor vykonať počas dvoch taktov vďaka dvojfázovej pipeline (načítanie a vykonanie inštrukcie) [1]. Pri frekvencii 16 MHz to znamená, že vypnutie a opätovné zapnutie tranzistora by teoreticky malo trvať 1/4 mikrosekundy. Ďalším zaujímavým pozorovaním je, že časový interval podpätia sa nepredlžoval so zväčšovaním počtu iterácií „prázdneho“ for-cyklu. Dôvodom môže byť optimalizácia kompilátora, ktorý sa oprávnene rozhodol zdanlivo „zbytočný“ for-cyklus odstrániť.

Rozhodli sme sa preto časti kódu, ktoré ovládajú tranzistor prepísať do jazyku asemblera s využitím C Inline Assembly, ktorý je podporovaný aj kompilátorom prostredia Arduino IDE. Volania funkcie „digitalWrite“ sme teda nahradili ekvivalentnou konštrukciou pomocou inštrukcií SBI a CBI. Následne sme útok zopakovali s takto upraveným programom nahratým na Arduino Nano. Výsledkom bolo, že podpätie na mikrokontroléri trvalo pripližne 1/2 mikrosekundy, čo je štyrikrát menej ako predtým. Rozdiel medzi teoretickým časom (1/4 ms), bol pravdepodobne spôsobený nedokonalosťou tranzistora a vplyvom ďalších fyzikálnych faktorov. Pri takto krátkom čase už nenastal reset žiadného z testovaných čipov, ale útok bol opäť neúspešný. Interval bol pravdepodobne príliš krátky a na žiadnom z čipov sa neprejavila chyba. Ďalej sme preto upravili kód napísaním vlastnej procedúry oneskorenia v asembleri (opäť s využitím C Inline Assembly). Procedúra pozostáva z inicializácie registra na kladnú hodnotu (osem-bitový parameter) a cyklu. V cykle postupne dekrementujeme tento register a ná-

Algoritmus 3.2: Jednoduchá procedúra oneskorenia v asembleri. {IN} označuje vstupný parameter – 8-bitová konštanta, alebo hodnota v registeri.

```
mov r24, {IN}    // 1 takt
loop:
    dec r24      // 1 takt
    brne loop    // 2 takty pri vykonani skoku, 1 inak
```

Tabuľka 3.1: Porovnanie vylepšeného útoku zo súťaže CTF na rôznych čipoch ATMega328P. Čísla v tabuľke udávajú vstupný parameter (počet cyklov) procedúry oneskorenia. Na čipy zo série 2128BQY pôvodný útok nebol úspešný.

Séria čipu	Min (úspešný útok)	Max (úspešný útok)	Ideál (vždy úspešný útok)
2139E4A	6	12	9–10
2139E4A	4	11	10
2128BQY	3	11	7
2128BQY	4	12	8–9

sledne vykonáme podmienený skok na začiatok cykla, pokiaľ výsledok dekrementu bol nenulový. Pseudokód procedúry v jazyku asemblera uvádzame v algoritme 3.2. Takáto procedúra umožňuje parametrizovať oneskorenie s presnosťou na trojce taktov (cyklus procedúry trvá tri takty). Po tejto úprave sa útok úspešne podaril na všetkých štyroch testovaných čipov. Potrebné oneskorenia pre úspešný útok na každom z exemplárov sú zhrnuté v tabuľke 3.1. Väčšiu presnosť by bolo možné dosiahnúť vsnutím presného počtu inštrukcií NOP, medzi vypnutím a zapnutím tranzistora. Počet inštrukcií NOP, by však musel byť známy v čase kompilácie, čo by znemožnilo dynamicky upravovať dĺžku oneskorenia za behu.

Kapitola 4

Analýza výsledkov

V tejto kapitole analyzujeme výsledky a účinnosť uskutočnených útokov na daný hardvér.

Záver

Na záver už len odporúčania k samotnej kapitole Záver v bakalárskej práci podľa smernice: „V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.“

Všimnite si správne písanie slovenských úvodzoviek okolo predchádzajúceho citátu, ktoré sme dosiahli príkazom `\uv`.

V informatických prácach niekedy býva záver kratší ako dve strany, ale stále by to mal byť rozumne dlhý text, v rozsahu aspoň jednej strany. Okrem dosiahnutých cieľov sa zvyknú rozoberať aj otvorené problémy a námety na ďalšiu prácu v oblasti.

Abstrakt, úvod a záver práce obsahujú podobné informácie. Abstrakt je kratší text, ktorý má pomôcť čitateľovi sa rozhodnúť, či vôbec prácu chce čítať. Úvod má umožniť zorientovať sa v práci skôr než ju začne čítať a záver sumarizuje najdôležitejšie veci po tom, ako prácu prečítal, môže sa teda viac zamerať na detaily a využívať pojmy zavedené v práci.

Literatúra

[1] Atmel Corporation. *ATMega328P Datasheet*, 2015. Rev. 7810D–AVR–01/15.

[2] Atmel Corporation. *AVR Instruction Set Manual*, 2016. Rev. Atmel-0856L-AVR-Instruction-Set-Manual_Other-11/2016.

[3] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 105–114, 2011.

[4] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[5] Christoffer Claesson. Voltage glitching on the cheap, 2022. [Citované 2022-12-7] Dostupné z `https://blog.securitybits.io/2019/06/voltage-glitching-on-the-cheap/`.

[6] NewAE Technology Inc. Tutorial A2 introduction to glitch attacks (including glitch explorer), 2019. [Citované 2023-1-31] Dostupné z `https://wiki.newae.com/V4:Tutorial_A2_Introduction_to_Glitch_Attacks_(including_Glitch_Explorer)`.

[7] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 77–88, 2013.

[8] Colin O'Flynn. Fault injection using crowbars on embedded systems. *Cryptology ePrint Archive*, 2016.

[9] Colin O'Flynn. Building against fault injection attacks, 2020. [Citované 2022-1-31] Dostupné z `https://circuitcellar.com/research-design-hub/building-against-fault-injection-attacks/`.

[10] Colin O'Flynn and Zhizhang David Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 243–260. Springer, 2014.

Príloha A: obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádza zdrojový kód programu a súbory s výsledkami experimentov. Zdrojový kód je zverejnený aj na stránke `http://mojadresa.com/`.

Ak uznáte za vhodné, môžete tu aj podrobnejšie rozpísať obsah tejto prílohy, prípadne poskytnúť návod na inštaláciu programu. Alternatívou je tieto informácie zahrnúť do samotnej prílohy, alebo ich uviesť na obidvoch miestach.

Príloha B: Používateľská príručka

V tejto prílohe uvádzame používateľskú príručku k nášmu softvéru. Tu by ďalej pokračoval text príručky. V práci nie je potrebné uvádzať používateľskú príručku, pokiaľ je používanie softvéru intuitívne alebo ak výsledkom práce nie je ucelený softvér určený pre používateľov.

V prílohách môžete uviesť aj ďalšie materiály, ktoré by mohli pôsobiť rušivo v hlavnom texte, ako napríklad rozsiahle tabuľky a podobne. Materiály, ktoré sú príliš dlhé na ich tlač, odovzdajte len v electronickej prílohe.