Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών

«ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ, ΑΣΦΑΛΕΙΑ ΚΑΙ ΑΝΑΔΥΟΜΕΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Ανάπτυξη Πλατφόρμας Εισαγωγής Σφαλμάτων για την Αξιολόγηση της Ασφάλειας Ενσωματωμένων Συστημάτων Development of a Fault Injection Platform for the Security Evaluation of Embedded Systems
Ονοματεπώνυμο Φοιτητή	ΔΗΜΗΤΡΙΟΣ ΝΑΣΟΥΦΗΣ
Πατρώνυμο	ΙΩΑΝΝΗΣ
Αριθμός Μητρώου	ΜΠΚΣΑ/ 19018
Επιβλέπων	ΜΙΧΑΗΛ ΨΑΡΑΚΗΣ, ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

Ιούλιος 2021

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Αναπλ. Καθηγητής Μιχαήλ Ψαράκης	Αναπλ. Καθηγητής Παναγιώτης Κοτζανικολάου	Αναπλ. Καθηγητής Κωνσταντίνος Πατσάκης

Ευχαριστίες

Από τη θέση αυτή επιθυμώ να ευχαριστήσω από καρδιάς:

- Τον επιβλέποντά μου Δρ. Αθανάσιο Παπαδημητρίου για την πολύτιμη βοήθειά του σε κάθε βήμα της προσπάθειάς μου. Επίσης, τον ευχαριστώ για την υπομονή του, καθώς και την άμεση υποστήριξη σε κάθε δυσκολία που συνάντησα στο πλαίσιο της εκπόνησης και συγγραφής της μεταπτυχιακής διατριβής μου.
- Τον επιβλέποντα αναπληρωτή καθηγητή μου κύριο Μιχαήλ Ψαράκη που συνέβαλε επικουρικά στην αξιολόγηση της μεταπτυχιακής διατριβής μου.
- Την σύζυγό μου για την όλη στήριξη και βοήθειά της σε όλη τη διάρκεια της προσπάθειάς μου.

Ιούλιος 2021
Νασούφης Ι. Δημήτριος

Περίληψη

Μία από τις πιο δημοφιλείς μεθόδους αξιολόγησης της ασφάλειας και της αξιοπιστίας των ενσωματωμένων συστημάτων είναι η εισαγωγή σφάλματος (Fault Injection - FI), στην οποία προσομοιώνεται η συμπεριφορά του συστήματος παρουσία σφαλμάτων. Με τη μέθοδο αυτή δίνεται, επίσης, η δυνατότητα σε κακόβουλους χρήστες να ακυρώσουν μηχανισμούς ασφάλειας (πχ. έλεγχος PIN) ή/και να αποκαλύψουν κρυπτογραφικά κλειδιά αλγόριθμων κρυπτογράφησης.

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής είναι η δημιουργία μίας πλατφόρμας για την εισαγωγή των σφαλμάτων, καθώς και για την καταγραφή τους και η αξιολόγησή της εισάγοντας σφάλματα σε μικροελεγκτή ενός ενσωματωμένου συστήματος, όπου εκτελείται κώδικας ελέγχου PIN (Personal Identification Number), εισάγοντας σφάλματα κατά την εκτέλεση του κώδικα.

Η πλατφόρμα λαμβάνει ως είσοδο μια ακολουθία εκτέλεσης συμβολικών εντολών (assembly instructions) ενός μικροελεγκτή (microcontroller) και ανάλογα με το μοντέλο εισαγωγής σφαλμάτων που έχει επιλέξει ο χρήστης, παράγει και εισάγει σφάλματα κατά την εκτέλεση του κώδικα. Τέλος, αποθηκεύει τα αποτελέσματα για περαιτέρω εξέταση κι αξιολόγηση.

Abstract

One of the most popular methods of evaluating the safety and reliability of embedded systems is Fault Injection (FI), which simulates the behavior of the system in the presence of errors. This method also allows malicious users to cancel security mechanisms (e.g., PIN control) and / or reveal cryptographic keys of encryption algorithms.

The object of this master's thesis is the development of a platform for the injection of errors, as well as for their recording and the evaluation of the platform by injecting errors in a microcontroller of an integrated system, where a PIN (Personal Identification Number) code is executed, injecting errors during execution of the code.

The platform receives as input a sequence of assembly instructions of a microcontroller and depending on the target error model chosen by the user, generates and injects errors during the execution of the code. Finally, it stores the results for further examination and evaluation

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Συντομογραφιών

COM: Communication port

COTS:	Commercial off-the-shelf
EMV:	Europay, Mastercard, and Visa
FPGA :	Field Programmable Gate Arrays
FI :	Fault Injection
GUI:	Graphical User Interface
GDB:	The GNU Project Debugger
IC:	Integrated Circuit
MCU:	MicroController Unit
OpenOCD:	Open On-Chip Debugger
PC:	Personal Computer
PED:	PIN Entry Device
PIN:	Personal Identification Number
QEMU:	Quick EMUlator
STM32:	STM32F103RB
SW:	System Workbench for STM32
VHDL:	Very high-speed integrated circuit Hardware Description Language

1 Εισαγωγή

Το ενσωματωμένο λογισμικό (embedded software) αναπτύσσεται με την προϋπόθεση ότι η εκτέλεση του υλικού (hardware) είναι πάντα ορθή. Όμως, μέσω της εισαγωγής στοχευμένων σφαλμάτων, μπορεί να τροποποιηθεί η ροή ελέγχου ή η ακεραιότητα ροής των δεδομένων ή ακόμα και τα ίδια τα δεδομένα ενός προγράμματος, το οποίο ενδέχεται να επηρεάσει την έξοδο του προγράμματος. Τα σφάλματα προέρχονται, συνήθως, από φαινόμενα που προκαλούνται με την πρόσκρουση σωματιδίων στην επιφάνεια ενός ολοκληρωμένου κυκλώματος (Integrated Circuit - IC) ή από τις επιδράσεις της ακτινοβολίας του περιβάλλοντος, π.χ. της κοσμικής ακτινοβολίας, στην λειτουργία των ηλεκτρονικών κυκλωμάτων. Η πιθανότητα εμφάνισης σφάλματος εξαρτάται από πολλούς παράγοντες, όπως οι περιβαλλοντικές συνθήκες, το υψόμετρο και η ποσότητα της ακτινοβολίας στη θέση λειτουργίας του IC.

Οι εφαρμογές σε ενσωματωμένα συστήματα και εξαρτήματα, όπως έξυπνες κάρτες, κινητά τηλέφωνα κλπ, πρέπει να «οπλιστούν» έναντι της παρουσίας σφαλμάτων – που οφείλονται είτε σε περιβαλλοντικές αιτίες είτε σε κακόβουλους χρήστες - ώστε να διασφαλιστεί η ασφαλής και αξιόπιστη λειτουργία τους. Η υλοποίηση μηχανισμών προστασίας και η αξιολόγηση της ασφάλειας των συστημάτων έναντι της παρουσίας σφαλμάτων, είναι ένα πρόβλημα που απασχολεί όλους τους εμπλεκόμενους φορείς στον κύκλο ζωής των συστημάτων. Αυτό έχει ως αποτέλεσμα την ανάπτυξη διαφόρων εργαλείων αξιολόγησης για τα διαφορετικά στάδια σχεδίασης και λειτουργίας των συστημάτων, μεταξύ αυτών και εργαλείων προσομοίωσης. Μια σημαντική δυσκολία στην ανάπτυξη και αξιολόγηση αυτών των εργαλείων είναι η έλλειψη αντιπροσωπευτικών κριτηρίων και μετρικών για την αξιολόγηση ή σύγκριση των ληφθέντων αποτελεσμάτων. Προς αυτήν την κατεύθυνση, στην παρούσα διατριβή αναπτύχθηκε μια πλατφόρμα εισαγωγής σφαλμάτων σε ενσωματωμένους επεξεργαστές, και αξιολογήθηκε με πειράματα σε ενσωματωμένες εφαρμογές.

Στην παρούσα διατριβή, αναπτύχθηκε μία πλατφόρμα εισαγωγής σφαλμάτων και αξιολογήθηκε με πειράματα σε μια εφαρμογή αυθεντικοποίησης με χρήση ελέγχου PIN (CheckPIN), που εκτελείται σε ένα ενσωματωμένο σύστημα (STM32 Nucleo-64 development board with STM32F103RB MCU. Με τη βοήθεια της πλατφόρμας εισαγωγής σφαλμάτων που έχουμε αναπτύξει, και η οποία έχει υλοποιηθεί σε περιβάλλον Matlab, παράγεται αρχικά μια λίστα σφαλμάτων. Εν συνεχεία, με χρήση του GNU Project Debugger, εισάγονται τα σφάλματα ένα-ένα κατά τη διάρκεια εκτέλεσης του κώδικα, στον μικροελεγκτή STM32 πλακέτας. Τέλος, συλλέγονται τα αποτελέσματα και αξιολογούνται ως προς την επίδρασή των σφαλμάτων στην εκτέλεση της εφαρμογής. Επιπλέον, προτείνεται μία λύση για την αύξηση της ασφάλειας, μέσω της τροποποίησης του κώδικα.

Η δομή της διατριβής έχει ως εξής: στο Κεφάλαιο 2 γίνεται μία αναφορά στα σφάλματα, στις κατηγορίες αυτών, καθώς και στα περιβάλλοντα εισαγωγής σφαλμάτων. Στο Κεφάλαιο 3 περιγράφεται η πλατφόρμα, που αναπτύχθηκε στο πλαίσιο της διατριβής, η εφαρμογή αυθεντικοποίησης CheckPIN και η διαδικασία αξιολόγησης που ακολουθήθηκε. Τέλος, στο Κεφάλαιο 4 παρουσιάζονται τα αποτελέσματα της αξιολόγησης και στο Κεφάλαιο 5 παρατίθενται συμπεράσματα και προτάσεις για την βελτίωση της ασφάλειας του ενσωματωμένου λογισμικού.

2 Γενικά περί εισαγωγής σφαλμάτων

Σε γενικές γραμμές, οι τεχνικές εισαγωγής σφαλμάτων (fault injection - fi) μπορούν να κατηγοριοποιηθούν σε αυτές που βασίζονται: α) στο υλικό (hardware-based), β) στο λογισμικό (software-based), γ) στην προσομοίωση (simulation-based), δ) στην εξομοίωση (emulation-based) και ε) υβριδικές (hybrid). Η διαδικασία δημιουργίας ενός περιβάλλοντος fi εξαρτάται από διάφορες παραμέτρους που επηρεάζουν τη συνοχή και τη σημασία των αποτελεσμάτων. Στην συνέχεια του κεφαλαίου γίνεται μία ανάλυση αυτών των παραμέτρων, για να επισημανθεί πως τυχόν σφάλματα στην μεθοδολογία μπορεί να επηρεάσουν την επιλογή αυτών των παραμέτρων και συνεπώς, να οδηγήσουν σε μη αξιόπιστες αξιολογήσεις του συστήματος - στόχου. Επίσης, αναλύονται οι τέσσερις βασικές αρχές που διέπουν την ορθή ανάπτυξη ενός περιβάλλοντος fi, και αφορούν: α) το μοντέλο σφαλμάτων και τη λίστα σφαλμάτων, β) το φορτίο εργασίας και τα δεδομένα που θα εφαρμοστούν στο σύστημα - στόχο, γ) τα αποτελέσματα που θα επιλεγούν και δ) τον τρόπο ερμηνείας των αποτελεσμάτων.

2.1 Κατηγορίες Σφαλμάτων

Ένα σφάλμα, ως απόκλιση από την προβλεπόμενη λειτουργία του συστήματος, μπορεί να προκύψει σε όλα τα στάδια της διαδικασίας σχεδιασμού και ανάπτυξης είτε να εισαχθεί από οποιονδήποτε κακόβουλο χρήστη για να παρακάμψει τους μηχανισμούς ασφαλείας ενός συστήματος. Τα περισσότερα σφάλματα, που εμφανίζονται πριν από την πλήρη ανάπτυξη ενός συστήματος, ανακαλύπτονται και εξαλείφονται μέσω δοκιμών. Τα σφάλματα που δεν εξαλείφονται μπορούν να μειώσουν κατά πολύ την αξιοπιστία και την ασφάλεια ενός συστήματος.

Μια γενική κατηγοριοποίηση των σφάλματων σχεδιασμού είναι η εξής:

- Φυσικά σφάλματα ή σφάλματα υλικού: προκύπτουν κατά τη λειτουργία του συστήματος και ταξινομούνται ανάλογα με τη διάρκειά τους, σε μόνιμα (permanent faults), προσωρινά (transient faults) και διαλείποντα / διακοπτόμενα (intermittent faults) [5].

- Σφάλματα λογισμικού: προέρχονται από λανθασμένο σχεδιασμό, κατά την ανάλυση των προδιαγραφών ή κατά την ανάπτυξη του κώδικα του λογισμικού. Πολλά από αυτά τα σφάλματα είναι λάθη στον κώδικα και εμφανίζονται κατά την εκτέλεση φορτίων εργασίας που είναι είτε πολύπλοκα ή μη συνηθισμένα και ως εκ τούτου δεν έχουν δοκιμαστεί ενδελεχώς.

Στην παρούσα διατριβή θα ασχοληθούμε με σφάλματα από επιθέσεις κακόβουλων χρηστών, με σκοπό την παραβίαση της ασφάλειας των ενσωματωμένων συστημάτων. Αυτά τα σφάλματα κατηγοριοποιούνται όπως παρακάτω:

- Μόνιμα σφάλματα (permanent ή hard faults): Στα μόνιμα σφάλματα, αλλάζει η τιμή ενός στοιχείου μόνιμα. Μπορεί να αλλοιωθούν τα δεδομένα ή ο κώδικας του λογισμικού. Ένα μόνιμο σφάλμα μπορεί να είναι πολύ ισχυρό όταν αλλάζει ένα μυστικό κλειδί. Ωστόσο, είναι πολύ δύσκολο να προκαλέσουμε μόνιμα σφάλματα. [23]

- Προσωρινά σφάλματα (transient ή soft faults): Τα προσωρινά σφάλματα δεν αλλάζουν μόνιμα τις τιμές των στοιχείων. Το κύκλωμα ανακτά την αρχική του κατάσταση μετά την επαναφορά ή όταν σταματήσει το ερέθισμα του σφάλματος. Ένα παροδικό σφάλμα μπορεί να διαταράξει την εκτέλεση κώδικα ή έναν συγκεκριμένο υπολογισμό. Επομένως, μια κλήση σε

μια υπορουτίνα μπορεί να παραλειφθεί, να αποφευχθεί μια δοκιμή, να εκτελεστούν διαφορετικές εκτελέσεις, να ληφθεί λανθασμένη τιμή ή να τροποποιηθεί ένας μετρητής προγράμματος [23].

Τόσο τα μόνιμα, όσο και τα προσωρινά σφάλματα επιτρέπουν εξίσου την παραβίαση της ασφάλειας ενός ενσωματωμένου συστήματος. Αυτό δίνει τη δυνατότητα σε έναν κακόβουλο χρήστη να προκαλέσει σφάλματα κατά τον υπολογισμό του κρυπτογραφικού αλγορίθμου και να εκμεταλλευτεί τα λανθασμένα αποτελέσματα για να εξαγάγει πληροφορίες σχετικά με το μυστικό κλειδί κρυπτογράφησης. Οι επιθέσεις εισαγωγής σφαλμάτων μπορούν να «σπάσουν» ένα μη προστατευμένο σύστημα πιο γρήγορα από οποιοδήποτε άλλο είδος επίθεσης πλευρικού καναλιού (side channel attack - SCA), όπως η απλή ανάλυση ισχύος (simple power analysis - SPA), η ανάλυση διαφορικής ισχύος (differential power analysis - DPA) ή η ηλεκτρομαγνητική ανάλυση (electromagnetic analysis - EMA). Για παράδειγμα, ο εισβολέας μπορεί να σπάσει έναν RSA-CRT (RSA with Chinese Remainder Theorem) αλγόριθμο με ένα λανθασμένο αποτέλεσμα και τους Data Encryption Standard (DES) και Advanced Encryption Standard (AES) με δύο [23]. Επίσης, έχει τη δυνατότητα, ο κακόβουλος χρήστης να παρακάμψει τους μηχανισμούς ελέγχου ασφαλείας που χρησιμοποιούν PIN ή να αλλάξει την ορθή λειτουργία ενσωματωμένων συστημάτων, τα οποία χρησιμοποιούνται κατά κόρον στην βιομηχανία για την ασφάλεια των εγκαταστάσεών τους και τη λειτουργία τους.

2.2 Μέθοδοι Εισαγωγής Σφαλμάτων

Υπάρχουν πολλοί τρόποι για την εισαγωγή σφαλμάτων. Παρακάτω περιγράφονται οι πιο κοινές μέθοδοι εισαγωγής σφάλματος, που χρησιμοποιούνται από τους κακόβουλους χρήστες, για να παραβιάσουν την ασφάλεια του συστήματος [23].

- Επίθεση μικροσφάλματος – δυσλειτουργίας (Glitch attack): Οι διακυμάνσεις στην τάση τροφοδοσίας ή στο εξωτερικό ρολόι μπορούν να προκαλέσουν προβλήματα στην λειτουργία μιας συσκευής. Για παράδειγμα, μπορεί να οδηγήσουν σε παράλειψη κάποιων εντολών (instructions) του επεξεργαστή ή παρερμηνεία αυτών ή ακόμα και να προκαλέσουν εσφαλμένη ανάγνωση δεδομένων. Για να το πετύχει αυτό, ο εισβολέας πρέπει να ελέγξει την ένταση και τη διάρκεια της δυσλειτουργίας (του παλμού glitch). Ένα πλεονέκτημα της επίθεσης μικροσφάλματος είναι ότι δεν μπορεί να ανιχνευθεί εύκολα. Για αυτό το λόγο αποτελεί μια από τις πιο κοινές μεθόδους για την παραβίαση πολλών κρυπτοσυστημάτων. Αντίθετα, το κύριο μειονέκτημα αυτού του τύπου επίθεσης είναι ότι, ο εισβολέας δεν μπορεί να εστιάσει σε συγκεκριμένα μέρη της συσκευής.

- Επίθεση θερμοκρασίας (Temperature attack): Σε ακραίες θερμοκρασίες, οι συσκευές δεν λειτουργούν σωστά. Μπορεί να συμβεί τυχαία τροποποίηση θέσεων μνήμης RAM ή αναντιστοιχία αναγνώσεων και εγγραφών σε μη πτητικές μνήμες (NVMs). Οι περισσότερες έξυπνες κάρτες έχουν ανιχνευτές υπέρβασης των ορίων θερμοκρασίας, αλλά μπορεί να συμβεί λάθος παραμετροποίηση του ανιχνευτή.

- Επίθεση φωτός (Light attack): Είναι από τις πιο ισχυρές επιθέσεις. Σε αντίθεση με μια επίθεση δυσλειτουργίας, σε αυτήν την περίπτωση ο εισβολέας μπορεί να επιλέξει τη θέση της επίθεσης στη συσκευή. Επειδή όλα τα ηλεκτρικά κυκλώματα είναι ευαίσθητα στο φως, λόγω φωτοηλεκτρικών εφέ, ο εισβολέας μπορεί να χρησιμοποιήσει το ρεύμα που προκαλείται από τα φωτόνια για να προκαλέσει σφάλματα. Για να το πετύχει αυτό, ο εισβολέας, πρέπει να

ελέγξει την ενέργεια, το μήκος κύματος, τη θέση και το χρόνο εκπομπής του φωτός. Ένα απλό φλας κάμερας μπορεί να εκτελέσει μια ελαφριά επίθεση. Το πλεονέκτημα αυτής της απλής μεθόδου είναι ότι απαιτεί πολύ φθηνό εξοπλισμό. Αλλά το βάθος διείσδυσης μιας ελαφριάς επίθεσης εξαρτάται από το μήκος κύματος του φωτός και το φλας της κάμερας παρέχει μόνο ορατά μήκη κύματος. Επιπλέον, ένα φλας κάμερας είναι δύσκολο να ελεγχθεί με ακρίβεια. Επομένως, ένα σύστημα λέιζερ μπορεί να είναι πιο αποτελεσματικό για να εκτελέσει κάποιος μια επίθεση φωτός. Επιτρέπει τη χρήση αρκετών διακριτών μηκών κύματος και τη στόχευση μιας πολύ μικρής περιοχής της συσκευής, καθιστώντας δύσκολη την προστασία, αν και οι περισσότερες έξυπνες κάρτες διαθέτουν ανιχνευτές φωτός και μεταλλικές ασπίδες. Επιπλέον, σήμερα είναι δυνατή η επίθεση με λέιζερ στην πίσω πλευρά του τσιπ, όπου συνήθως δεν εφαρμόζεται μηχανισμός προστασίας.

- Μαγνητική επίθεση (Magnetic attack): Είναι η χρήση εκπομπής ενός ισχυρού μαγνητικού παλμού κοντά στο τσιπ. Το μαγνητικό πεδίο δημιουργεί τοπικά ρεύματα στην επιφάνεια της συσκευής για να δημιουργήσει ένα σφάλμα. Αυτή η επίθεση μπορεί να πραγματοποιηθεί με χρήση εξοπλισμού χαμηλού κόστους - για παράδειγμα, μια βελόνα που τυλίγεται με σύρμα - και επιτρέπει την επίθεση σε μικρά τμήματα του τσιπ. Ωστόσο, πέραν του χαμηλού κόστους, η επίθεση με χρήση λέιζερ είναι πιο αποτελεσματική.

2.3 Βασική Δομή Περιβάλλοντος Εισαγωγής Σφαλμάτων

Ένα περιβάλλον εισαγωγής σφαλμάτων αποτελείται, σύμφωνα με το [5], από τα ακόλουθα συστατικά:

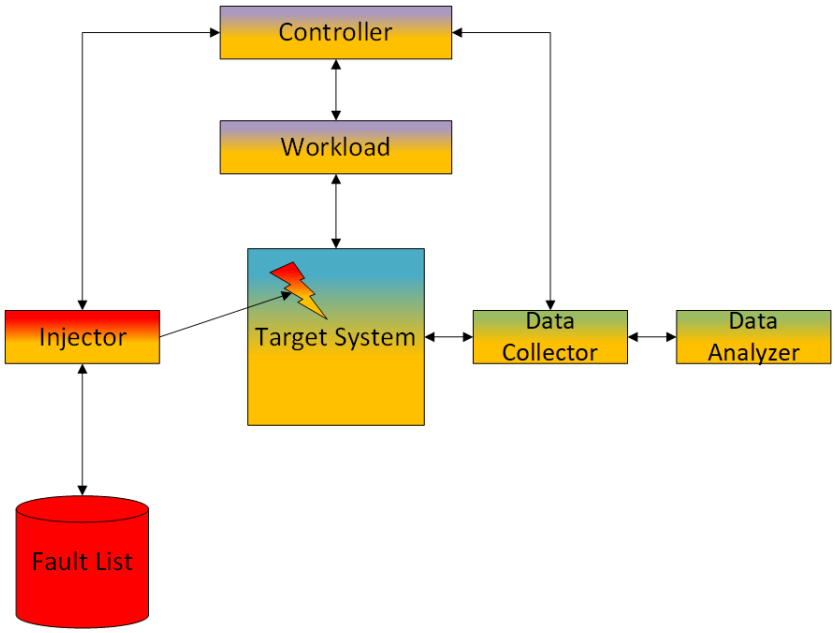
- *Fault injector*: Εισάγει τα σφάλματα στο σύστημα – στόχο, καθώς εκτελεί εντολές από τον Workload generator.
- *Fault library*: Αποθηκεύει διαφορετικούς τύπους σφαλμάτων, τοποθεσίες σφαλμάτων, χρόνους σφαλμάτων ή δομές λογισμικού.
- *Workload generator*: Δημιουργεί το φόρτο εργασίας για το σύστημα - στόχο ως είσοδο.
- *Workload library*: Αποθηκεύει δείγματα φόρτου εργασίας για το σύστημα - στόχο.
- *Controller*: Ελέγχει την εκτέλεση του πειράματος.
- *Monitor*: Παρακολουθεί την εκτέλεση των εντολών και ξεκινά τη συλλογή δεδομένων όποτε είναι απαραίτητο.
- *Data collector*: Εκτελεί άμεση συλλογή δεδομένων.
- *Data analyzer*: Πραγματοποιεί την επεξεργασία και ανάλυση των δεδομένων.

Ένας αποτελεσματικός τρόπος για να χαρακτηριστεί ένα περιβάλλον εισαγωγής σφάλματος είναι το μοντέλο F A R M, που προτάθηκε στο [9]. Τα τέσσερα χαρακτηριστικά του FARM είναι:

- το σύνολο των σφαλμάτων (Faults - "F") που εισάγονται στο σύστημα. Το F ονομάζεται επίσης λίστα σφαλμάτων. Κάθε σφάλμα χαρακτηρίζεται από ένα μοντέλο (π.χ. bit-flip, short, κ.λπ.), μια θέση (π.χ., ένα flip-flop, μία διεύθυνση μνήμης, ένα pin κ.λπ.) και το χρόνο

εισαγωγής (π.χ. μετά την εκτέλεση μιας συγκεκριμένης εντολής, μετά από ένα δεδομένο χρόνο, κ.λπ.). Το μέγεθος του χώρου σφαλμάτων είναι επομένως M x L x T, όπου M είναι το σύνολο των πιθανών μοντέλων σφάλματος (Model), L είναι το σύνολο των πιθανών θέσεων σφάλματος (Location) και T το σύνολο των πιθανών χρονικών στιγμών εισαγωγής σφάλματος (Time), που αντιστοιχούν στη διάρκεια κάθε πειράματος. Το μέγεθος του χώρου σφαλμάτων, ανάλογα με τις πιθανές τιμές των M, L, T, μπορεί να είναι πολύ μεγάλο. Το κύριο πρόβλημα στον καθορισμό της λίστας σφαλμάτων (F) είναι επομένως, η επιλογή ενός υποσυνόλου του συνολικού χώρου σφαλμάτων που μπορεί να εισαχθεί σε εύλογο χρονικό διάστημα, αλλά να εξακολουθεί να μπορεί να παρέχει στατιστικά σημαντικά αποτελέσματα.

- το σύνολο των σημείων ενεργοποίησης (Activation - "A"), που καθορίζει τον τρόπο λειτουργίας του συστήματος κατά τη διάρκεια του πειράματος. Αποτελεί το σύνολο των εισόδων που εφαρμόζονται στο σύστημα κατά τη διάρκεια κάθε πειράματος. Η επιλογή του A επηρεάζει άμεσα τη διάρκεια κάθε πειράματος και, κατά συνέπεια, το μέγεθος του χώρου σφαλμάτων (M x L x T) και της λίστας σφαλμάτων. Συχνά αυτό το μοντέλο επεκτείνεται, ώστε να περιλαμβάνει το σύνολο των φόρτων εργασίας (Workloads - "W"), π.χ. ένα σύνολο μετροπρογραμμάτων λογισμικού (software benchmarks) στην περίπτωση ενός συστήματος που βασίζεται σε μικροεπεξεργαστή.



Εικόνα 1. Τυπική δομή ενός περιβάλλοντος εισαγωγής σφαλμάτων [6]

- το σύνολο των καταγραφών (Readouts - "R"), που αντιστοιχεί στην καταγεγραμμένη συμπεριφορά του συστήματος. Τα δεδομένα που καταγράφονται στο R εξαρτώνται, σε μεγάλο βαθμό, από το σύστημα-στόχο και από τους μηχανισμούς που χρησιμοποιούνται για την παρακολούθηση της συμπεριφοράς του συστήματος. Για παράδειγμα, σε ένα σύστημα που βασίζεται σε μικροεπεξεργαστή, τα καταγεγραμμένα δεδομένα μπορεί να περιλαμβάνουν πρόσβαση στη μνήμη, τα τελικά αποτελέσματα της εφαρμογής ή τις εξαιρέσεις του συστήματος. Η ποιότητα και η λεπτομέρεια των καταγεγραμμένων δεδομένων μπορεί να επηρεάσει σημαντικά τη σημασία των τελικών αποτελεσμάτων της εισαγωγής σφάλματος, και επίσης, τη

διάρκεια του πειράματος. Η επιλογή του R πρέπει επομένως να είναι μια συνετή επιλογή μεταξύ ακρίβειας και επιβάρυνσης χρόνου ή μνήμης.

- το σύνολο των μέτρων (Measures - "M") που λαμβάνονται κατά την ανάλυση και επεξεργασία των καταγραφών ("R"), που αποθηκεύτηκαν κατά τη διάρκεια του πειράματος και που απαιτούνται για τον υπολογισμό της τελικής εκτίμησης αξιοπιστίας του συστήματος.

Δεδομένου του μοντέλου FARM, μια πλήρης διαδικασία εισαγωγής σφαλμάτων είναι μια συλλογή πειραμάτων, καθένα από τα οποία απαιτεί την εισαγωγή σφάλματος, f από το σύνολο των F, ενώ στο σύστημα δίνεται μια σειρά δεδομένων a που επιλέγεται από το A, σε ένα φόρτο εργασίας w από το W. Το σύνολο των μέτρων M λαμβάνεται με την επεξεργασία του συνόλου των μετρήσεων R, που συγκεντρώθηκαν κατά τη διάρκεια κάθε πειράματος. Στην Εικ. 1 παρουσιάζεται η τυπική δομή ενός περιβάλλοντος εισαγωγής σφάλματος.

2.4 Τεχνικές Εισαγωγής Σφάλματος

Για να μπορέσουμε να σχεδιάσουμε τεχνικές για την αποφυγή σφαλμάτων (fault mitigation) και την επαναφορά ενός κυκλώματος από την λανθάνουσα κατάσταση (error recovery), είναι απαραίτητο να εντοπιστούν οι ευαίσθητες περιοχές του λογισμικού κατά το σχεδιασμό. Σε γενικές γραμμές, εάν ένα σφάλμα εμφανίζεται σε μια ευαίσθητη περιοχή, έχει περισσότερες πιθανότητες να μετατραπεί σε ελάττωμα και να επηρεάσει την κανονική λειτουργία του κυκλώματος. Επίσης, μπορεί να το εκμεταλλευτούν κακόβουλοι χρήστες, οι οποίοι θα προσπαθήσουν να παραβιάσουν την ασφάλεια του συστήματος. Έτσι, με τον εντοπισμό των ευαίσθητων περιοχών και την αξιοποίηση κατάλληλων τεχνικών ανοχής σφαλμάτων, θα αποφευχθούν αυτές οι περιπτώσεις επιθέσεων.

Μία από τις πιο κοινές προσεγγίσεις για τον προσδιορισμό των ευαίσθητων περιοχών και τη μέτρηση της ασφάλειας και της αξιοπιστίας των κυκλωμάτων είναι η τεχνική εισαγωγής σφάλματος. Σε αυτήν την τεχνική, η ελαττωματική κατάσταση προσομοιώνεται με την εισαγωγή ενός σφάλματος στο κύκλωμα και την παρακολούθηση των επιπτώσεών του σε αυτό.

Στα [1], [5], [6] και [8] αναφέρονται διάφορες κατηγορίες - τεχνικές εισαγωγής σφάλματος, για την ανάλυση της αξιοπιστίας των ηλεκτρονικών συσκευών έναντι πιθανών σφαλμάτων. Αυτές οι τεχνικές έχουν διαφορετική πολυπλοκότητα, απαιτούν διαφορετικό χρονικό διάστημα για την εφαρμογή τους και έχουν ένα ευρύ φάσμα κόστους. Στη συνέχεια, χρησιμοποιώντας τις κατάλληλες αναλυτικές προσεγγίσεις, υπολογίζονται παράμετροι που χαρακτηρίζουν την ανοχή του κυκλώματος έναντι των σφαλμάτων, όπως για παράδειγμα ο ρυθμός αστοχίας του κυκλώματος.

Οι τεχνικές εισαγωγής σφάλματος μπορούν να ταξινομηθούν σε πέντε κύριες ομάδες, όπως φαίνεται και στην Εικ. 2:

α) hardware - based (physical)

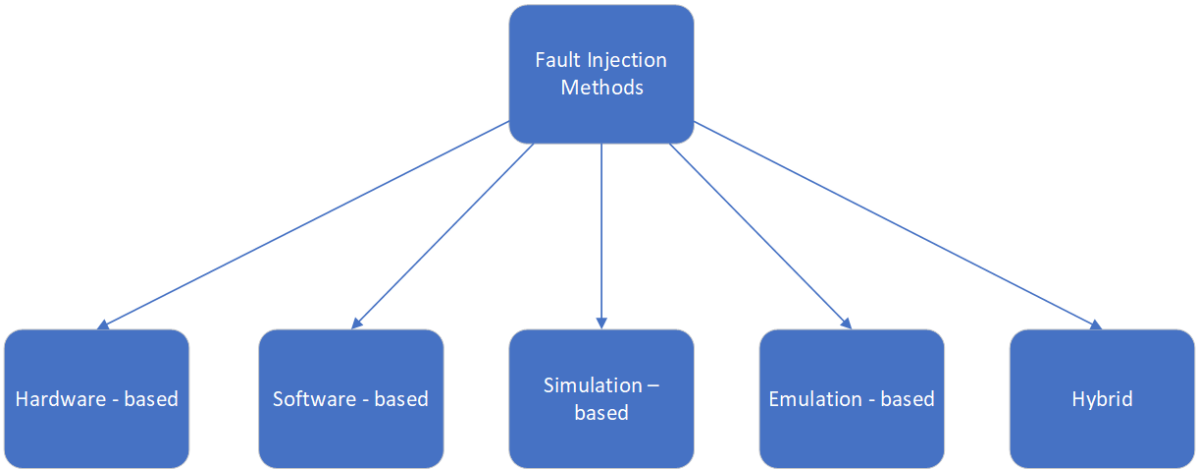
β) software - based

γ) simulation - based

δ) emulation - based

ε) hybrid

Τα πλεονεκτήματα και μειονεκτήματα, της κάθε μιας από τις παραπάνω τεχνικές, φαίνονται συνοπτικά στον Πίν. 1.



Εικόνα 2. Τεχνικές εισαγωγής σφάλματος

2.4.1 Hardware-based (physical) fault injection

Επιτυγχάνεται σε φυσικό επίπεδο, και παρεμβάλλεται στην λειτουργία του υλικού εφαρμόζοντας φυσικές διαταραχές ή μεταβάλλοντας τις συνθήκες περιβάλλοντος, για παράδειγμα χρησιμοποιώντας ακτινοβολία βαρέων ιόντων, ηλεκτρομαγνητικές παρεμβολές, κ.λπ. Αυτές οι διαταραχές μπορούν να προκαλέσουν ταλάντωση στην τάση τροφοδοσίας (διαταραχές τροφοδοσίας), να εισάγουν σφάλματα στις τιμές των μνημών με χρήση λέιζερ ή να τροποποιήσουν την τιμή των εισόδων/εξόδων του κυκλώματος.

Η εισαγωγή σφαλμάτων που βασίζεται στο υλικό, περιλαμβάνει την προσθήκη στο υπό αξιολόγηση σύστημα, ειδικά σχεδιασμένου υλικού, που επιτρέπει την εισαγωγή συγκεκριμένων τύπων σφαλμάτων στο σύστημα και την παρακολούθηση των εξόδων ώστε να εξεταστούν οι επιπτώσεις των σφαλμάτων στην λειτουργία του συστήματος. Ανάλογα με τα σφάλματα και τις θέσεις τους, οι μέθοδοι που βασίζονται στο υλικό, εμπίπτουν σε δύο κατηγορίες:

- Εισαγωγή σφάλματος με επαφή: υπάρχει άμεση φυσική επαφή με το σύστημα - στόχο, προκαλώντας διαταραχές στην τάση ή το ρεύμα στο τσιπ στόχου.

- Εισαγωγή σφάλματος υλικού χωρίς επαφή: δεν υπάρχει άμεση φυσική επαφή με το σύστημα - στόχο. Αντ 'αυτού, μια εξωτερική πηγή παράγει κάποιο φυσικό φαινόμενο, όπως ακτινοβολία βαρέων ιόντων ή ηλεκτρομαγνητικές παρεμβολές, που προκαλεί λανθασμένη λειτουργία στο τσιπ στόχο [5].

2.4.2 Software-based fault injection

Ο στόχος αυτής της τεχνικής είναι η αναπαραγωγή των σφαλμάτων σε επίπεδο λογισμικού. Τα σφάλματα λογισμικού είναι από τις κύριες αιτίες λανθασμένης λειτουργίας ενός συστήματος. Η μέθοδος αυτή περιλαμβάνει την τροποποίηση του λογισμικού, που εκτελείται στο υπό αξιολόγηση σύστημα, προκειμένου να διαφοροποιήσει την κατάσταση του συστήματος. Χρησιμοποιείται γενικά σε κώδικα που έχει μεθόδους (functions) που αλληλοεπιδρούν μεταξύ τους. Μπορούν να εισαχθούν όλα τα είδη σφαλμάτων, όπως σφάλματα στους καταχωρητές (registers) του επεξεργαστή, ή τη μνήμη ή σε πακέτα δικτύου που έχουν

αναπαραχθεί. Αυτά τα σφάλματα μπορούν να εισαχθούν σε προσομοιώσεις σύνθετων συστημάτων, όπου οι αλληλεπιδράσεις μεταξύ των μεθόδων είναι αντιληπτές, αλλά όχι οι λεπτομέρειες υλοποίησης, ή ακόμα και σε διεργασίες λειτουργικών συστημάτων, όπου τα αποτελέσματα των σφαλμάτων μελετώνται σε επίπεδο συστήματος.

Οι τεχνικές που εισάγουν σφάλματα λογισμικού εστιάζουν στις λεπτομέρειες υλοποίησης των εφαρμογών. Τα σφάλματα μπορεί να αφορούν λανθασμένους χρόνους εκτέλεσης εντολών, ελλιπή μηνύματα, επαναλήψεις μηνυμάτων, λανθασμένες τιμές μνήμης ή καταχωρητών, ελαττωματικές αναγνώσεις δίσκων και σχεδόν οποιαδήποτε άλλη ελαττωματική κατάσταση, στην οποία μπορεί να περιέλθει το λογισμικό. Στη συνέχεια, το σύστημα συνεχίζει τη λειτουργία του με σφάλμα για να εξεταστεί η συμπεριφορά του. Αυτές οι προσομοιώσεις διαρκούν περισσότερο επειδή ενσωματώνουν όλη τη λειτουργία και τις λεπτομέρειες του συστήματος, αλλά θα συλλάβουν με μεγαλύτερη ακρίβεια τις επιδράσεις του σφάλματος στο σύστημα - στόχο. Εδώ έλεγχος των αποτελεσμάτων πραγματοποιείται για να επαληθευτεί η αντίδραση του συστήματος σε σφάλματα και να καταγραφούν οι περιπτώσεις που τα σφάλματα δεν επηρέασαν την απόκριση του συστήματος. Η ανάλυση της συμπεριφοράς με σφάλματα λογισμικού γίνεται αργότερα στον κύκλο του σχεδιασμού για να προσδιοριστεί η ασφάλεια του τελικού συστήματος [5].

2.4.3 Simulation-based fault injection

Είναι η εισαγωγή σφαλμάτων σε μοντέλα υψηλού επιπέδου, συνήθως μοντέλα γλωσσών περιγραφής υλικού (hardware description language, HDL). Οι εισαγωγές σφαλμάτων που βασίζονται στην προσομοίωση υλοποιούνται με χαμηλότερο κόστος, αλλά η σημασία των αποτελεσμάτων τους εξαρτάται σε μεγάλο βαθμό από την ακρίβεια και το επίπεδο αφαίρεσης του μοντέλου του συστήματος - στόχου. Περιλαμβάνει την κατασκευή ενός μοντέλου προσομοίωσης, του υπό αξιολόγηση συστήματος - στόχου, συμπεριλαμβανομένου ενός λεπτομερούς μοντέλου προσομοίωσης του επεξεργαστή που χρησιμοποιείται. Τα μοντέλα προσομοίωσης αναπτύσσονται χρησιμοποιώντας μια γλώσσα περιγραφής υλικού, όπως η Very high speed integrated circuit Hardware Description Language (VHDL) ή η Verilog. Τα σφάλματα εισάγονται σε ένα HDL μοντέλο του συστήματος τροποποιώντας το μοντέλο ή χρησιμοποιώντας σενάρια προσομοίωσης. Αυτή η δυνατότητα επιτρέπει τη θεωρητική εισαγωγή σφαλμάτων σε οποιοδήποτε μέρος του συστήματος. Στη συνέχεια αναλύεται η συμπεριφορά του συστήματος χρησιμοποιώντας τον προσομοιωτή. Επιτρέπει την έγκαιρη αξιολόγηση της αξιοπιστίας του συστήματος. Επίσης, αξιολογούνται διαφορετικά επίπεδα αφαίρεσης, χρησιμοποιώντας διαφορετικές γλώσσες περιγραφής (π.χ. system-level ή HDL ή RTL description) [5].

2.4.4 Emulation-based fault injection

Αυτή η τεχνική έχει παρουσιαστεί ως εναλλακτική λύση για τη μείωση του χρόνου, που δαπανάται στις χρονοβόρες εκτελέσεις των τεχνικών εισαγωγής σφάλματος με βάση την προσομοίωση. Βασίζεται στην χρήση της τεχνολογίας Field Programmable Gate Arrays (FPGA) ή στην χρήση μικροελεγκτών (MCU) για να εξομοιώσει την λειτουργία του συστήματος – στόχου και να επιταχύνει την προσομοίωσης σφαλμάτων. Επιτρέπει στον σχεδιαστή να μελετήσει την πραγματική συμπεριφορά ενός κυκλώματος στο περιβάλλον εφαρμογής του, λαμβάνοντας υπόψη τις αλληλεπιδράσεις σε πραγματικό χρόνο.

Το κύκλωμα προς αξιολόγηση υλοποιείται σε FPGA χρησιμοποιώντας μια τυπική διαδικασία σχεδίασης κυκλωμάτων που ξεκινά από μια περιγραφή υψηλού επιπέδου και

περιλαμβάνει τα στάδια της σύνθεσης, τοποθέτησης και διασύνδεσης. Το FPGA ή η MCU συνδέεται με έναν κεντρικό υπολογιστή, ο οποίος χρησιμοποιείται για τον καθορισμό εισαγωγής σφαλμάτων, την εμφάνιση των αποτελεσμάτων και γενικά για τον έλεγχο όλου του πειράματος [5].

2.4.5 Hybrid

Μια υβριδική προσέγγιση συνδυάζει δύο ή περισσότερες από τις άλλες τεχνικές εισαγωγής σφαλμάτων για την αξιολόγηση ενός συστήματος – στόχο. Για παράδειγμα, μπορεί να συνδυάσει την software-based εισαγωγή σφαλμάτων και την παρακολούθηση μέσω υλικού (hardware). Σε αυτήν την περίπτωση, η εκτέλεση πειραμάτων εισαγωγής σφαλμάτων βάσει υλικού ή λογισμικού μπορεί να προσφέρει σημαντικό όφελος από την άποψη του χρόνου εκτέλεσης των πειραμάτων ή / και μπορεί να μειώσει τον αρχικό χρόνο εγκατάστασης, πριν από την έναρξη των πειραμάτων. Επιπλέον, συνδυάζει την ευελιξία της εισαγωγής σφάλματος λογισμικού και την ακρίβεια της παρακολούθησης μέσω υλικού. Είναι κατάλληλη για τη μέτρηση εξαιρετικά μικρών καθυστερήσεων. Ωστόσο, δεδομένου του σημαντικού κέρδους στην ικανότητα ελέγχου και παρατηρησιμότητας, με μια προσέγγιση που βασίζεται σε προσομοίωση, μπορεί να είναι χρήσιμο να συνδυαστεί, μια τέτοια τεχνική, με μια από τις άλλες προκειμένου να αξιολογηθεί πληρέστερα το σύστημα – στόχος [5].

Πίνακας 1. Πλεονεκτήματα και μειονεκτήματα των τεχνικών [5]

Τεχνική	Πλεονεκτήματα	Μειονεκτήματα
Hardware-Based	• Υψηλή ανάλυση χρόνου για ενεργοποίηση και παρακολούθηση μέσω υλικού • Κατάλληλο για μοντέλα σφαλμάτων χαμηλού επιπέδου • Δεν απαιτείται τροποποίηση του συστήματος - στόχου για την εισαγωγή σφαλμάτων • Τα πειράματα είναι γρήγορα • Δεν απαιτείται ανάπτυξη ή επικύρωση μοντέλου • Δυνατότητα μοντελοποίησης μόνιμων σφαλμάτων στο επίπεδο του pin	• Μπορεί να προκαλέσει μόνιμη βλάβη στο σύστημα - στόχο • Περιορισμένη παρατηρησιμότητα (observability) και ελεγξιμότητα (controllability) • Μειωμένη φορητότητα • Περιορισμένο σύνολο σημείων εισαγωγής και περιορισμένο σύνολο σφαλμάτων • Απαιτεί υλικό ειδικού σκοπού για την εκτέλεση των πειραμάτων
Software-Based	• Μπορεί να χρησιμοποιηθεί σε εφαρμογές και λειτουργικά συστήματα • Τα πειράματα μπορούν να εκτελεστούν σε σχεδόν πραγματικό χρόνο • Δεν απαιτεί υλικό ειδικού σκοπού. • Χαμηλή πολυπλοκότητα, χαμηλό κόστος υλοποίησης και ανάπτυξης • Δεν απαιτείται ανάπτυξη ή επικύρωση μοντέλου • Μπορεί να επεκταθεί σε νέες κατηγορίες σφαλμάτων	• Περιορισμένο σύνολο σφαλμάτων, μόνο σε επίπεδο assembly instruction • Δεν μπορεί να εισάγει σφάλματα σε τοποθεσίες που δεν είναι προσβάσιμες από το λογισμικό • Απαιτεί τροποποίηση του πηγαίου κώδικα • Περιορισμένη παρατηρησιμότητα και ελεγξιμότητα • Πολύ δύσκολο να μοντελοποιήσουμε μόνιμα σφάλματα
Simulation-Based	• Μπορεί να υποστηρίξει όλα τα επίπεδα αφαίρεσης ενός συστήματος - στόχου • Δεν απαιτείται τροποποίηση του συστήματος - στόχου για την εισαγωγή σφαλμάτων • Πλήρης έλεγχος τόσο των μοντέλων σφαλμάτων, όσο και των μηχανισμών εισαγωγής	• Μεγάλη προσπάθεια ανάπτυξης • Μεγάλες απαιτήσεις σε χρόνο • Το μοντέλο δεν είναι άμεσα διαθέσιμο • Η ακρίβεια των αποτελεσμάτων εξαρτάται από το μοντέλο που χρησιμοποιείται • Δεν είναι δυνατή η εισαγωγή σφαλμάτων σε πραγματικό χρόνο σε ένα πρωτότυπο

Τεχνική	Πλεονεκτήματα	Μειονεκτήματα
	• Αυτοματοποίηση μέσω υπολογιστών με χαμηλό κόστος και δεν απαιτεί υλικό ειδικού σκοπού • Μέγιστη παρατηρησιμότητα και ελεγξιμότητα • Επιτρέπει την αξιολόγηση αξιοπιστίας σε διάφορα στάδια της διαδικασίας σχεδιασμού • Δυνατότητα μοντελοποίησης τόσο παροδικών, όσο και μόνιμων βλαβών	• Το μοντέλο ενδέχεται να μην περιλαμβάνει κανένα από τα σφάλματα σχεδιασμού που είναι πιθανό να υπάρχουν στο πραγματικό υλικό
Emulation - based	• Απαιτείται λιγότερος χρόνος σε σχέση με τις τεχνικές που βασίζονται σε προσομοίωση • Ο χρόνος πειραματισμού μπορεί να μειωθεί εφαρμόζοντας, εν μέρει ή πλήρως, την παραγωγή προκαθορισμένων δεδομένων εισόδου στο FPGA	• Το κόστος ενός συστήματος εξομοίωσης υλικού και / ή η πολυπλοκότητα υλοποίησης ενός FPGA • Χρησιμοποιείται μόνο για την ανάλυση των λειτουργικών συνεπειών ενός σφάλματος • Με τη χρήση των FPGA, κύριος περιορισμός είναι ο αριθμός I/O's του προγραμματιζόμενου υλικού • Αναγκαιότητα σύνδεσης υψηλής ταχύτητας, μεταξύ του κεντρικού υπολογιστή και της πλακέτας προσομοίωσης

2.5 Βασικές Αρχές

Σύμφωνα με το [6], κατά το σχεδιασμό και την εκτέλεση ενός πειράματος εισαγωγής σφαλμάτων θα πρέπει να ακολουθούνται τέσσερις βασικές αρχές ή τρόπος του λέγειν «αρετές». Αυτές οι τέσσερις αρετές επιλέχθηκαν επειδή είναι «φυσικές» και μπορούν να επιτευχθούν μέσω της ανθρώπινης προσπάθειας.

Η Εγκράτεια (Temperance) είναι η αρετή που προτείνεται για την επιλογή ενός αποτελεσματικού μοντέλου εισαγωγής σφαλμάτων (F). Η ανάγκη διαπραγμάτευσης μεταξύ πολλών περιορισμών απαιτεί την καλύτερη επιλογή, λαμβάνοντας υπόψη τον προϋπολογισμό, τους χρονικούς περιορισμούς και την αντιπροσωπευτικότητα της λίστας σφαλμάτων. Έτσι, θα πρέπει να επιλεγεί ένα ρεαλιστικό μοντέλο σφαλμάτων, λαμβάνοντας υπόψη το πραγματικό σύστημα - στόχο και τις συνθήκες λειτουργίας του. Ο ισχυρισμός ότι ο χώρος σφαλμάτων είναι άπειρος θα κάνει τα αποτελέσματα εντελώς άχρηστα, διότι δεν θα είναι δυνατό να αξιολογηθεί αν το εισαχθέν σύνολο σφαλμάτων, είναι στατιστικά αντιπροσωπευτικό ολόκληρου του χώρου σφαλμάτων. Υπάρχουν πάντα ισοδύναμα σφάλματα σε σχέση με το χρόνο (ίδια θέση, αλλά διαφορετικό χρόνο εισαγωγής) ή / και χώρο (διαφορετική θέση σφάλματος, αλλά ίδια επίδραση στο σύστημα), τα οποία πρέπει να εντοπίζονται, για να ελαχιστοποιηθεί ο αριθμός των πραγματικών εισαγωγών. Τέλος, θα πρέπει να εφαρμόζονται ουσιαστικές μέθοδοι δειγματοληψίας σφαλμάτων, αντί μιας απλής ομοιόμορφης επιλογής:

- με στατιστική δειγματοληψία, εάν είναι γνωστή η κατανομή χώρου σφαλμάτων.
- λαμβάνοντας υπόψη την ειδική δομή του υλικού και του λογισμικού που βρίσκεται υπό αξιολόγηση:

(α) τα σφάλματα δεν πρέπει ποτέ να εισάγονται σε μη χρησιμοποιημένα μέρη του συστήματος – στόχου.

(β) η εισαγωγή πρέπει να επικεντρώνεται στα πιο κρίσιμα μέρη του συστήματος - στόχου.

- εάν τα σφάλματα που πρόκειται να εισαχθούν επιλέγονται τυχαία από μια μεγαλύτερη λίστα, το πείραμα θα πρέπει να επαναλαμβάνεται με διαφορετικές λίστες σφαλμάτων, έως ότου επιτευχθεί το απαιτούμενο επίπεδο εμπιστοσύνης.

Η δικαιοσύνη (Justice) είναι απαραίτητη κατά την επιλογή του συνόλου των σημείων ενεργοποίησης (A). Πρέπει να αντικατοπτρίζουν τις πραγματικές συνθήκες λειτουργίας και όχι να επιλεχθούν απλώς για να λειτουργήσει το πείραμα, λαμβάνοντας υπόψη, τόσο τον σκοπό του πειράματος, όσο και τη λίστα σφαλμάτων. Εάν ο σκοπός είναι η αξιολόγηση της αξιοπιστίας του υλικού, είναι απαραίτητο να επιλεγεί ένας φόρτος εργασίας, ικανός να ενεργοποιήσει την πλειονότητα (πιθανώς το 100%) των λειτουργιών του συστήματος - στόχου. Αν ο σκοπός είναι να αξιολογηθεί το λογισμικό, ο φόρτος εργασίας πρέπει να εγγυηθεί την υψηλότερη δυνατή κάλυψη των λειτουργιών του λογισμικού, επιλέγοντας ένα σύνολο φόρτου εργασίας αντιπροσωπευτικό της πραγματικής λειτουργίας.

Η σύνεση (Prudence) είναι απαραίτητη για την επιλογή των καταγραφών (R). Η καταγραφή της συμπεριφοράς του συστήματος μπορεί να είναι μια πολύ χρονοβόρα εργασία που επηρεάζει άμεσα τη διάρκεια και την ακρίβεια των πειραμάτων. Οι μετρήσεις εξαρτώνται σε μεγάλο βαθμό από τον τύπο του περιβάλλοντος εισαγωγής σφαλμάτων. Γι΄ αυτό το λόγο θα πρέπει να επιλέγεται ένα εργαλείο εισαγωγής σφαλμάτων, ικανό να λειτουργεί στο ίδιο επίπεδο αφαίρεσης στο οποίο πρέπει να ληφθούν τα αποτελέσματα. Τα σημεία καταγραφής πρέπει να επιλέγονται προσεκτικά, ώστε να αντιπροσωπεύουν με ακρίβεια το πραγματικό τμήμα του συστήματος – στόχου, που είναι κρίσιμο για τα αποτελέσματα του πειράματος.

Τελευταία και πιο σημαντική αρχή είναι το θάρρος (Courage) κατά την επεξεργασία των αποτελεσμάτων και τη λήψη μέτρων για την αύξηση της αξιοπιστίας (M). Είναι πολύ εύκολο να εξαχθούν «βιαστικά» συμπεράσματα, που δεν υποστηρίζονται από στατιστικό υπόβαθρο. Τα πραγματικά και χρήσιμα αποτελέσματα απαιτούν την ανάλυση των ανεπεξέργαστων δεδομένων με θάρρος, χωρίς το φόβο προσθήκης επιπλέον πειραμάτων ή τροποποίησης της μεθοδολογίας για τη λήψη σημαντικών αποτελεσμάτων. Έτσι, συνιστάται η χρήση πραγματικών στατιστικών στοιχείων και όχι μόνο ένας ελάχιστος αριθμός αποτελεσμάτων, απλά για να φαίνονται επαγγελματικά, με:

- τη μη παροχή απόλυτων αποτελεσμάτων, αλλά σε συνδυασμό με εκτίμηση σφάλματος και εμπιστοσύνης.
- τη συσχέτιση των αποτελεσμάτων με τις συνθήκες λειτουργίας.
- την προσεκτική γενίκευση των αποτελεσμάτων.
- το να λαμβάνεται υπόψη η πιθανότητα εμφάνισης σφάλματος. Ένα πολύ καταστροφικό σφάλμα με πολύ χαμηλή πιθανότητα εμφάνισης, μπορεί να μην είναι τόσο κρίσιμο.
- με την κριτική αξιολόγηση των αποτελεσμάτων, προκειμένου γίνει κατανοητό εάν αντικατοπτρίζουν την πραγματική συμπεριφορά του συστήματος ή προκαλούνται από σφάλμα στη ρύθμιση της εισαγωγής σφάλματος.

2.6 Σχετικές Προσεγγίσεις

Προκειμένου να υποστηρίξουν τις διαδικασίες ανάπτυξης και σχεδιασμού, έχουν προταθεί διάφορα εργαλεία, είτε για την ανάλυση της ανοχής των εφαρμογών σε σφάλματα, είτε για τη βελτίωση της ασφάλειας των εφαρμογών, με την προσθήκη αντιμέτρων λογισμικού. Όλα αυτά τα εργαλεία εφαρμόζονται αποκλειστικά σε συγκεκριμένα μοντέλα σφαλμάτων και επίπεδα κώδικα.

Στην εργασία [8], οι συγγραφείς μοντελοποιούν τις επιθέσεις σφαλμάτων λογισμικού έξυπνων καρτών, σε επίπεδο πηγαίου κώδικα, και στη συνέχεια, προσομοιώνουν αυτές τις επιθέσεις για να ξεχωρίσουν τις επιβλαβείς. Μερικές από αυτές τις επιθέσεις είναι φυσικές διαταραχές των στοιχείων του chip, που προκαλούν λανθασμένη συμπεριφορά στην εκτέλεση κώδικα. Μια επιτυχής λειτουργική επίθεση μπορεί να αποκαλύψει ένα μυστικό ή να δώσει μια ανεπιθύμητη εξουσιοδότηση. Η εργασία επικεντρώνεται στις επιθέσεις ροής ελέγχου. Τέτοιες επιθέσεις είναι καλοί υποψήφιοι για το προτεινόμενο μοντέλο, που παρουσιάζεται στο άρθρο και μπορούν να χρησιμοποιηθούν για να ελέγξουν την ανθεκτικότητα του προγράμματος.

Στην εργασία [9], παρουσιάζεται μια ανάλυση πλευρικού καναλιού (side channel analysis) από ηλεκτρομαγνητικές εκπομπές σε αλγόριθμους VERIFY PIN. Για να εισαχθεί ένας κωδικός PIN, ένας χρήστης έχει περιορισμένο αριθμό δοκιμών. Επομένως, η κύρια δυσκολία της επίθεσης είναι να πετύχει με πολύ λίγα ίχνη. Έτσι, παρουσιάζεται μια νέα πραγματική απειλή, η οποία είναι εφικτή σε μια χαμηλού κόστους και φορητή πλατφόρμα. Επιπλέον, σ' αυτό το άρθρο φαίνεται ότι ορισμένες προστασίες για αλγόριθμους VERIFY PIN, έναντι επιθέσεων εισαγωγής σφάλματος, εισάγουν νέες ευπάθειες σε σχέση με την ανάλυση πλευρικών καναλιών.

Στην εργασία [10], διερευνάται η ευπάθεια ασφαλείας του προσωπικού αριθμού αναγνώρισης (PIN) ή αριθμητικών κωδικών πρόσβασης, σε συσκευές εισόδου PIN (PED - Pin Entry Device). Αυτές οι συσκευές χρησιμοποιούνται συνήθως σε διάφορες βιομηχανικές και καταναλωτικές ηλεκτρονικές εφαρμογές (όπως είσοδος σε σημεία ελέγχου ασφαλείας, ATM κλπ.) και βασίζονται σε σύντομες ακολουθίες δεδομένων (PIN ή κωδικούς πρόσβασης) ως μέσο επαλήθευσης της νομιμότητας ενός χρήστη. Σε αυτό το άρθρο, προτείνεται μια επίθεση πλευρικού καναλιού σε ένα τυχαίο PIN 4-6 ψηφίων και μια μέθοδος επαλήθευσης χρήστη με PIN. Τα διαστήματα μεταξύ δύο πληκτρολογήσεων εξάγονται από την ακουστική εκπομπή και χρησιμοποιούνται ως χαρακτηριστικά για την εκπαίδευση μοντέλων μηχανικής εκμάθησης. Το μοντέλο επίθεσης έχει 60% πιθανότητα να ανακτήσει το PIN και 88% ακρίβεια στην αναγνώριση του χρήστη. Οι μέθοδοι επίθεσης μπορούν να πραγματοποιήσουν ανάκτηση PIN, χρησιμοποιώντας το ακουστικό πλευρικό κανάλι, με χαμηλό κόστος. Ως αντίμετρο, η μέθοδος αυτή, μπορεί να βελτιώσει την ασφάλεια των συσκευών εισαγωγής PIN.

Στην εργασία [11] περιγράφεται και επιδεικνύεται ένα ελάττωμα του πρωτοκόλλου EMV (Europay Mastercard and Visa), το οποίο εξασφαλίζει συναλλαγές πιστωτικών και χρεωστικών καρτών με έλεγχο ταυτότητας, τόσο της κάρτας, όσο και του πελάτη που την κατέχει, μέσω ενός συνδυασμού κρυπτογραφικών κωδικών ελέγχου ταυτότητας, ψηφιακών υπογραφών και της εισαγωγής PIN. Το ελάττωμα αυτό επιτρέπει στους κακόβουλους χρήστες να χρησιμοποιούν μια γνήσια κάρτα για να κάνουν μια πληρωμή, χωρίς να γνωρίζουν το PIN της κάρτας και να παραμένουν μη ανιχνεύσιμοι, ακόμη και όταν ο έμπορος έχει απευθείας σύνδεση στο διαδίκτυο με το τραπεζικό δίκτυο. Ο απατεώνας εκτελεί μια επίθεση man-in-the-middle, για να

εξαπατήσει, από τη μία, το τερματικό ώστε να πιστέψει ότι το PIN επαληθεύτηκε σωστά και από την άλλη την κάρτα ώστε να πιστέψει πως δεν έχει εισαχθεί καθόλου PIN. Εξετάζεται, επίσης, πώς προέκυψαν τα ελαττώματα, γιατί παρέμειναν άγνωστα παρά την ευρεία ανάπτυξη του EMV και πώς μπορούν να διορθωθούν. Εφόσον βρέθηκε και επικυρώθηκε μια πρακτική επίθεσης κατά της βασικής λειτουργικότητας του EMV, συμπεραίνεται ότι το πρωτόκολλο είναι αναξιόπιστο. Αυτή η αποτυχία είναι σημαντική στον τομέα του σχεδιασμού πρωτοκόλλων και έχει επίσης σημαντικές επιπτώσεις στη δημόσια πολιτική, υπό το φως των αυξανόμενων αναφορών απάτης με κλεμμένες κάρτες.

Στην εργασία [12] παρουσιάζεται μια πλατφόρμα εισαγωγής σφαλμάτων βασισμένη στην πλατφόρμα εξομοίωσης ανοικτού-κώδικα QEMU που υποστηρίζει εμπορικούς επεξεργαστές, οι οποίοι χρησιμοποιούνται ευρέως στον τομέα των ενσωματωμένων συστημάτων. Αυτή η πλατφόρμα επιτρέπει την ανάλυση, σε επίπεδο συστήματος, των αντιμέτρων λογισμικού παρουσιάζοντας την προσομοίωση σφαλμάτων υλικού υψηλού επιπέδου με στόχο, για παράδειγμα, θέσεις μνήμης, καταχωρητές ή τη σωστή εκτέλεση των εντολών. Υποστηρίζει τη δημιουργία ρεαλιστικών σεναρίων επίθεσης σφαλμάτων. Η εργασία επιδεικνύει την πρακτικότητα της προσέγγισης παρουσιάζοντας δύο υποδειγματικές περιπτώσεις χρήσης.

Οι υλοποιήσεις της έξυπνης κάρτας είναι επιρρεπείς σε επιθέσεις διαταραχής, που επιφέρουν την αλλαγή της κανονικής συμπεριφοράς αυτών, προκειμένου να δημιουργηθούν εκμεταλλεύσιμα σφάλματα. Οι διαταραχές θα μπορούσαν να πραγματοποιηθούν με διαφορετικά μέσα, όπως ακτίνες λέιζερ που περιλαμβάνουν δαπανηρές και πολύπλοκες πλατφόρμες εισαγωγής σφάλματος. Έτσι, στην εργασία [18], προτείνεται η ενσωμάτωση του μηχανισμού εισαγωγής σφαλμάτων κατευθείαν στην έξυπνη κάρτα. Το πρόγραμμα προσομοίωσης σφαλμάτων ενσωματώνεται στο λογισμικό chip και τα αποτελέσματά του μπορούν να αναλυθούν με παρατηρήσεις πλευρικού καναλιού, κάτι που δεν συμβαίνει με τυχόν υπάρχοντες προσομοιωτές σφαλμάτων. Σε αυτό το άρθρο, παρουσιάζεται αυτή η νέα ιδέα και ο αρχιτεκτονικός σχεδιασμός της και εφαρμόζεται σε ένα πραγματικό προϊόν έξυπνης κάρτας. Τέλος, για την επικύρωση αυτής της προσέγγισης, μελετάται η λειτουργική και πλευρική επίδραση της εισαγωγής σφάλματος σε έναν τυπικό αλγόριθμο που παρέχεται από την έξυπνη κάρτα.

Μια πρόσφατη επίθεση σε έξυπνες κάρτες βασίζεται σε εισαγωγή σφάλματος που τροποποιεί τη συμπεριφορά της εφαρμογής. Στην εργασία [19], προτείνεται μια αξιολόγηση της επίδρασης της διάδοσης και της δημιουργίας εχθρικών εφαρμογών μέσα στην κάρτα. Παρουσιάζονται διάφορα αντίμετρα και μοντέλα έξυπνων καρτών. Στη συνέχεια, αξιολογείται η ικανότητα αυτών των αντιμέτρων να εντοπίζουν τα σφάλματα και η καθυστέρηση της ανίχνευσης.

Οι έξυπνες κάρτες Java είναι συσκευές που υπόκεινται σε επιθέσεις υλικού και λογισμικού. Επομένως, πρέπει να ενσωματωθούν αρκετά αντίμετρα για να αποφευχθούν οι επιπτώσεις αυτών των επιθέσεων. Πρόσφατα, προέκυψε η ιδέα συνδυασμού λογικών επιθέσεων με φυσικές επιθέσεις για την παράκαμψη της επαλήθευσης bytecode. Για παράδειγμα, σωστές και νόμιμες εφαρμογές κάρτας Java μπορούν να τροποποιηθούν στην κάρτα, χρησιμοποιώντας δέσμη λέιζερ και έτσι γίνονται μεταλλαγμένες εφαρμογές, με διαφορετική συμπεριφορά. Αυτή η εσωτερική αλλαγή θα μπορούσε να οδηγήσει σε παράκαμψη του ελέγχου και της προστασίας και συνεπώς να προσφέρει παράνομη πρόσβαση σε μυστικά δεδομένα και λειτουργίες εντός του chip. Στην εργασία [20], προτείνεται ένα σύνολο

αντιμέτρων, που μπορούν να ενεργοποιηθούν από τον προγραμματιστή χρησιμοποιώντας τον μηχανισμό επισημάνσεων, όπου δημιουργούνται Java επισημάνσεις (annotations), οι οποίες χρησιμοποιούνται από τον διερμηνέα (Java interpreter) μίας εικονικής μηχανής (VM). Στη συνέχεια, το VM αλλάζει σε «ασφαλή λειτουργία», όταν συναντά μια συνάρτηση (function) που έχει επισημανθεί ως ευαίσθητη. Αυτά τα αντίμετρα είναι αποτελεσματικά, αλλά και προσιτά για τον τομέα των έξυπνων καρτών, όπως φαίνεται από την αξιολόγηση που παρατίθεται σε αυτήν την εργασία.

Τέλος, η εισαγωγή προσωρινών σφαλμάτων, ως τρόπος επίθεσης κρυπτογραφικών εφαρμογών, έχει μελετηθεί σε μεγάλο βαθμό την τελευταία εικοσαετία. Έχουν ήδη παρουσιαστεί αρκετές επιθέσεις που χρησιμοποιούν την ηλεκτρομαγνητική εισαγωγή σφάλματος σε αρχιτεκτονικές υλικού ή λογισμικού. Στους μικροελεγκτές, η εισαγωγή σφάλματος ηλεκτρομαγνητικά οδηγεί σε παράλειψη των εντολών εκτέλεσης κώδικα ή των κλήσεων υπορουτίνας. Η συνεισφορά της εργασίας [21] είναι διπλή: παρουσιάζει μια εμπεριστατωμένη μελέτη των επιπτώσεων της εισαγωγής σφάλματος με ηλεκτρομαγνητικά μέσα σε έναν σύγχρονο μικροελεγκτή και προτείνει ένα σχετικό μοντέλο σφαλμάτων.

3 Πλατφόρμα Εισαγωγής Σφαλμάτων

Σε αυτό το κεφάλαιο παρουσιάζεται η διαδικασία αξιολόγησης της πλατφόρμας εισάγοντας σφάλματα σε ενσωματωμένο λογισμικό, που ονομάστηκε CheckPIN, και η οποία ακολουθεί μία emulation – based προσέγγιση. Ξεκινάει μια περιγραφή της πλατφόρμας εισαγωγής σφαλμάτων, ακολουθεί η περιγραφή του κώδικα που εκτελείται στον μικροελεγκτή του STM32 και τέλος, γίνεται μία αναφορά στα πλεονεκτήματα και τα μειονεκτήματα της πλατφόρμας.

3.1 Περιγραφή Πλατφόρμας

Πριν την αξιολόγηση του κώδικα, προηγήθηκε η ανάπτυξη της πλατφόρμας αξιολόγησης και η ανάπτυξη του ίδιου του, προς αξιολόγηση, κώδικα.

Η πλατφόρμα που αναπτύχθηκε είναι βασισμένη στο [3]. Στην Εικ. 3 απεικονίζεται η δομή και η λειτουργία της.

Σε ότι αφορά στο υλικό, αποτελείται από έναν ηλεκτρονικό υπολογιστή (PC) και την πλακέτα του STM32, τα οποία συνδέονται με σειριακή σύνδεση (COM).

Σε ότι αφορά στο λογισμικό, χρησιμοποιούνται τα εξής πακέτα:

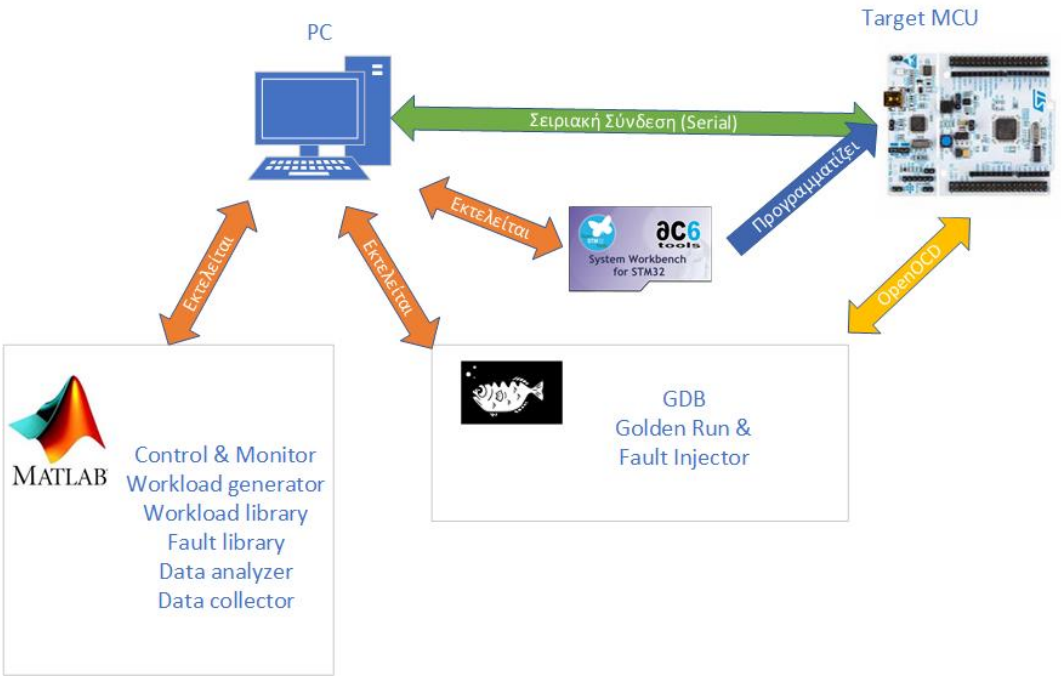
- Matlab R2020b, για την ανάπτυξη του γραφικού περιβάλλοντος (Graphical User Interface - GUI) της πλατφόρμας και για την εκτέλεση των λειτουργιών που περιγράφονται παρακάτω,
- System Workbench (SW4STM32), για τον προγραμματισμό και την επικοινωνία με τη πλακέτα STM32. Η εργαλειοθήκη System Workbench, που ονομάζεται SW4STM32, είναι ένα δωρεάν περιβάλλον ανάπτυξης λογισμικού πολλαπλών λειτουργιών που βασίζεται στο Eclipse, το οποίο υποστηρίζει την πλήρη γκάμα των μικροελεγκτών STM32 και τις σχετικές πλακέτες και είναι διαθέσιμη στον ιστότοπο www.openstm32.org.
- GDB (The GNU Project Debugger), για καταγραφή των εντολών, που εκτελούνται από τον επεξεργαστή και την εισαγωγή των σφαλμάτων (Fault injector) στον κώδικα.
- OpenOCD 0.10.0, για τη λογική σύνδεση της πλακέτας του STM32 με το GDB.

Σύμφωνα με την Ενότητα 2.3, το PC, μέσω του Matlab, χρησιμοποιείται για την εκτέλεση των παρακάτω λειτουργιών:

- τον έλεγχο της αξιολόγησης (Controller),
- την παρακολούθηση της εκτέλεσης των εντολών (Monitor),
- τη συλλογή δεδομένων και την ανάλυσή τους (Data collector and analyzer),
- τη δημιουργία φόρτου εργασίας (Workload generator) στην υπό αξιολόγηση πλακέτα και τέλος
- τη δημιουργία και αποθήκευση λιστών σφαλμάτων (Fault library).

Ο έλεγχος και η παρακολούθηση της αξιολόγησης γίνονται μέσω του GUI (Εικ. 4). Από το GUI της πλατφόρμας δίνεται η δυνατότητα των παρακάτω λειτουργιών:

I. Εκτέλεση του SW4STM32 για τον προγραμματισμό της πλακέτας του μικροελεγκτή STM32, καθώς και την επεξεργασία του κώδικα που εκτελείται στον μικροελεγκτή (Εικ. 5)..



Εικόνα 3. Πλατφόρμα Αξιολόγησης

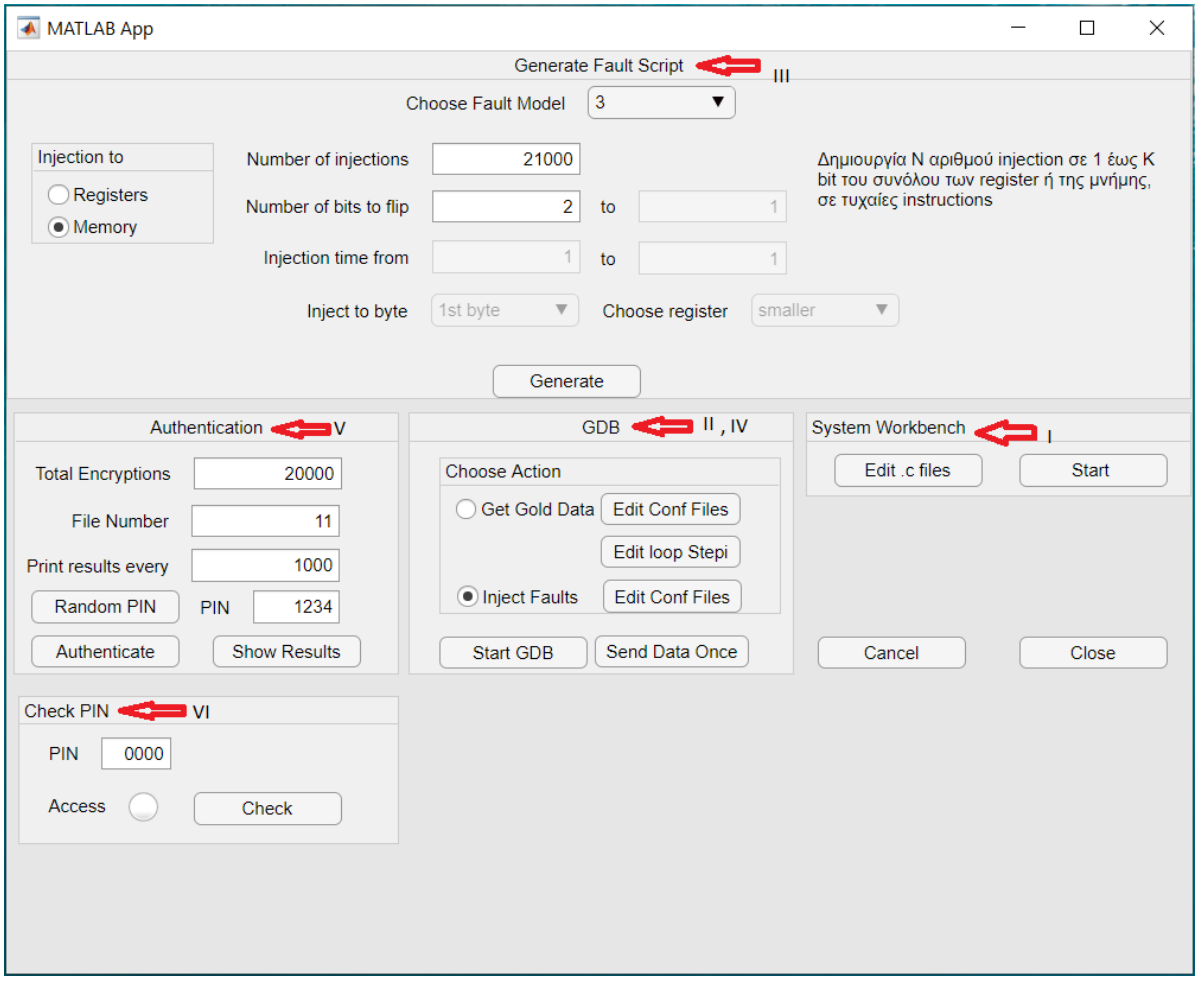
II. Ενεργοποίηση του GDB (και του OpenOCD ταυτόχρονα) για την καταγραφή των εντολών που υλοποιούνται, κατά την εκτέλεση του κώδικα, στην πλακέτα του STM32 (Εικ. 6).

III. Δημιουργία λιστών σφαλμάτων, σύμφωνα με τέσσερα (4) μοντέλα εισαγωγής σφαλμάτων.

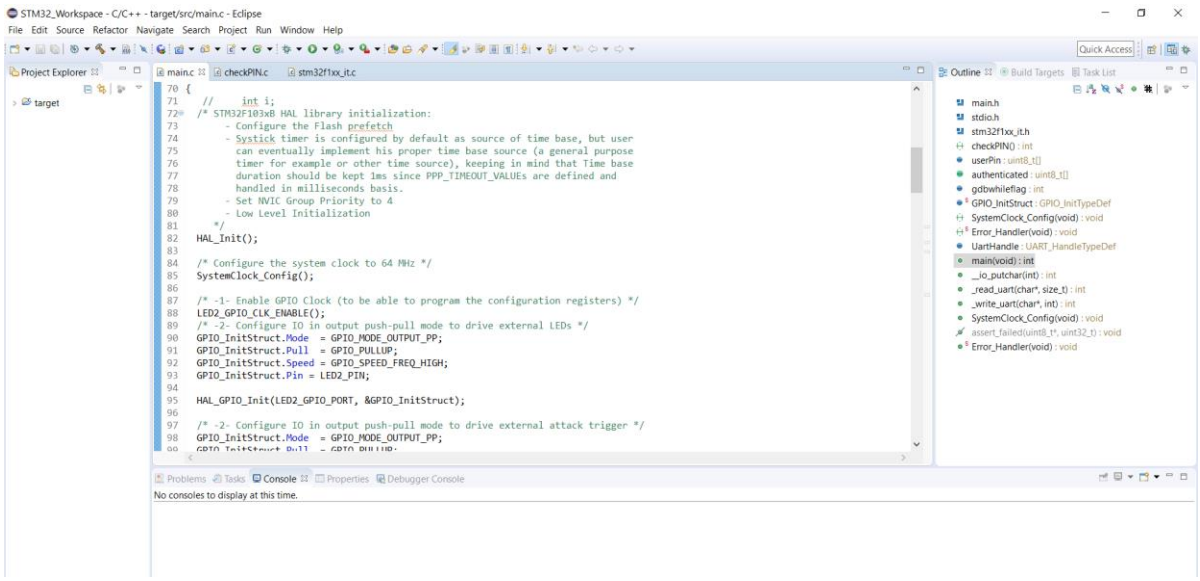
IV. Ενεργοποίηση του GDB, το οποίο επιτρέπει στην πλατφόρμα να βλέπει τι συμβαίνει «μέσα» σε ένα άλλο πρόγραμμα ενώ εκτελείται, (και του OpenOCD ταυτόχρονα) για την εισαγωγή σφαλμάτων, είτε στη μνήμη (SRAM) είτε στους καταχωρητές (registers).

V. Δημιουργία δεδομένων (input) για εισαγωγή στον μικροελεγκτή, έτσι ώστε να καταστεί δυνατή η εισαγωγή των σφαλμάτων. Αποστέλλει είτε το PIN που εισάγει ο χρήστης, είτε δημιουργεί και αποστέλλει τυχαία PIN.

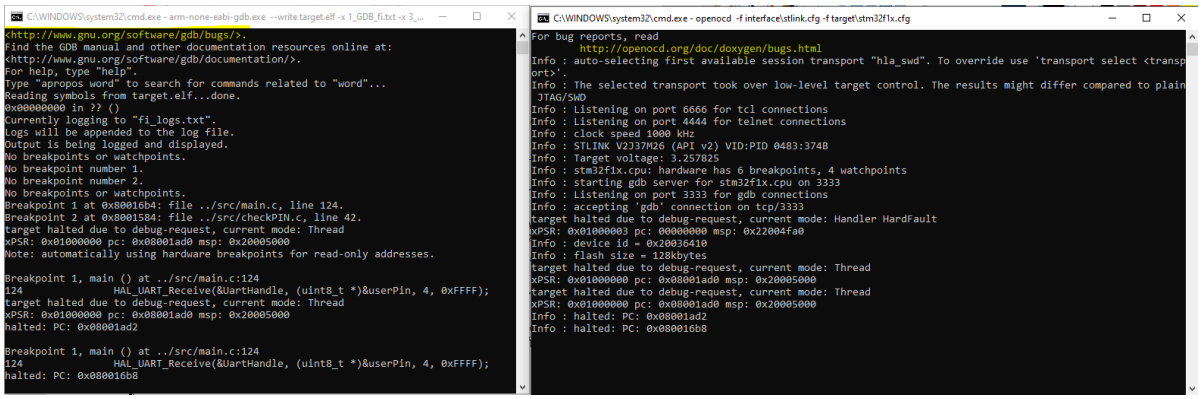
VI. Δοκιμή και έλεγχο σωστής λειτουργίας του κώδικα που εκτελείται στον STM32. Εισάγοντας κάποιο PIN, γίνεται έλεγχος κι αν είναι το σωστό, το LED γίνεται πράσινο και κόκκινο σε περίπτωση λάθους.



Εικόνα 4. Graphical User Interface



Εικόνα 5. System Workbench for STM32



Εικόνα 6. GDB και OpenOCD

Τα βήματα που εκτελεί ο GDB για την καταγραφή των εντολών που «τρέχει» ο επεξεργαστής κατά την εκτέλεση του προγράμματος (Golden Run) και για την εισαγωγή των σφαλμάτων, φαίνονται στον παρακάτω πίνακα (Πίν. 2). Τα αρχεία για την παραμετροποίηση του GDB, μπορούν να τροποποιηθούν μέσω της πλατφόρμας. Επίσης, αξίζει να επισημανθεί ότι, μετά από κάθε εισαγωγή σφάλματος, ο GDB επαναφέρει τον μικροελεγκτή στην αρχική του κατάσταση, έτσι ώστε κάθε σφάλμα να εισάγεται σε μια νέα εκτέλεση της εφαρμογής, χωρίς επιρροές από το προηγούμενο σφάλμα.

Πίνακας 2. Βήματα του GDB για την καταγραφή των εντολών και την εισαγωγή σφάλματος

Διαδικασία	Αλγόριθμος
Καταγραφή Αρχικών Εντολών	1. Πήγαινε στο σημείο διακοπής (ορίζεται με breakpoint) 2. Εκτέλεσε και κατέγραψε βήμα – βήμα το πρόγραμμα έως το τέλος του κώδικα υπό αξιολόγηση (όπου το "βήμα" σημαίνει μία εντολή μηχανής) 3. Συνέχεια εκτέλεσης κώδικα
Εισαγωγή Σφάλματος	1. Πήγαινε στο σημείο διακοπής 2. Εκτέλεσε βήμα – βήμα το πρόγραμμα, έως την επιθυμητή χρονική στιγμή εισαγωγής σφάλματος 3. Εισαγωγή σφάλματος 4. Συνέχεια 5. Επαναφορά

Από την πλατφόρμα υπάρχει η δυνατότητα παραγωγής λιστών σφαλμάτων σύμφωνα με τέσσερα διαφορετικά μοντέλα εισαγωγής σφαλμάτων, όπως φαίνεται και στον Πίν. 3.

Πίνακας 3. Μοντέλα εισαγωγής σφαλμάτων

Μοντέλο	Περιγραφή
1	Δημιουργία N αριθμού σφαλμάτων, με τη μέθοδο bit-flip, σε 1 έως 32 bit των καταχωρητών ή της μνήμης, σε τυχαίες χρονικές στιγμές. Αν υπάρχουν δύο καταχωρητές στην εντολή (instruction), επιλέγεται ή ο μεγαλύτερος ή ο μικρότερος ή τυχαία ένας από τους δύο
2	Δημιουργία N αριθμού σφαλμάτων, με τη μέθοδο bit-flip, σε 1 έως 8 bit των καταχωρητών ή της μνήμης, από την K έως την L χρονική στιγμή. Επιλογή

Μοντέλο	Περιγραφή
	του byte που θα γίνει η εισαγωγή (1ο, 2ο, 3ο, 4ο, 2 πρώτα bytes, or 2 τελευταία bytes). Αν υπάρχουν δύο καταχωρητές στην εντολή (instruction), επιλέγεται ή ο μεγαλύτερος ή ο μικρότερος ή τυχαία ένας από τους δύο
3	Δημιουργία N αριθμού σφαλμάτων, με τη μέθοδο bit-flip, σε K τυχαία bits (1 ≤ K ≤ 736 για τους καταχωρητές ή 163.840 για τη μνήμη) από το σύνολο των bits των καταχωρητών ή της μνήμης, σε τυχαίες χρονικές στιγμές
4	Δημιουργία N αριθμού σφαλμάτων, με τη μέθοδο bit-flip, από το K έως το L bit (K ≤ L και 1 ≤ L ≤ 736 για τους καταχωρητές ή 163.840 για τη μνήμη) του συνόλου των bit των καταχωρητών ή της μνήμης, σε τυχαίες χρονικές στιγμές

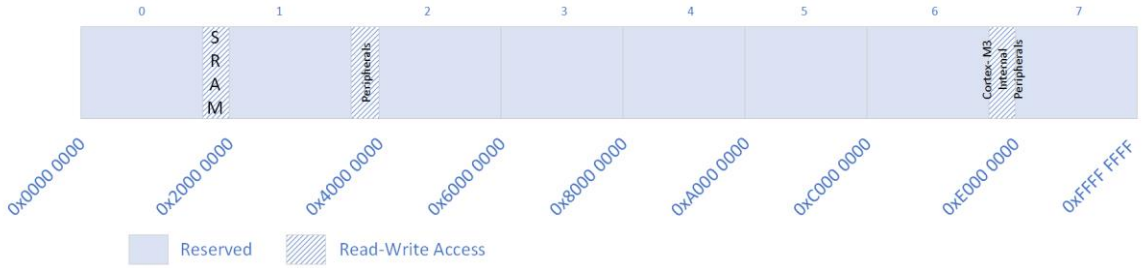
Στα δύο πρώτα μοντέλα γίνεται εισαγωγή σφαλμάτων μόνο στους καταχωρητές γενικού σκοπού, οι οποίοι εμπλέκονται στις εντολές που εκτελούνται από τον επεξεργαστή, ενώ στα δύο τελευταία η εισαγωγή γίνεται στο σύνολο των καταχωρητών, οι οποίοι έχουν 32-bit μέγεθος έκαστος ή 736 bits στο σύνολό τους, όπως αυτοί φαίνονται στον Πίν. 4.

Πίνακας 4. Περιγραφή καταχωρητών [16]

Καταχωρητής	Περιγραφή
r0 – r12	32-bit καταχωρητές γενικού σκοπού για πράξεις δεδομένων
Stack Pointer (SP)	Είναι ο καταχωρητής r13. Σε λειτουργία νήματος (Thread mode), το bit [1] του καταχωρητή CONTROL υποδεικνύει στο δείκτη στοίβας να χρησιμοποιήσει: • 0 για τον Main Stack Pointer (MSP) • 1 για τον Process Stack Pointer (PSP)
Link Register (LR)	Είναι ο καταχωρητής r14. Αποθηκεύει την διεύθυνση επιστροφής για υπορουτίνες, κλήσεις συναρτήσεων και εξαιρέσεων
Program Counter (PC)	Είναι ο καταχωρητής r15. Περιέχει την διεύθυνση της τρέχουσας εντολής του προγράμματος
Program status register (xPSR)	Συνδυάζει τους: • Application Program Status Register (APSR) • Interrupt Program Status Register (IPSR) • Execution Program Status Register (EPSR)
Priority mask register (primask)	Αποτρέπει την ενεργοποίηση όλων των εξαιρέσεων με διαμορφώσιμη προτεραιότητα
Base priority mask register (basepri)	Καθορίζει την ελάχιστη προτεραιότητα για επεξεργασία εξαιρέσεων
Fault mask register (faultmask)	Αποτρέπει την ενεργοποίηση όλων των εξαιρέσεων εκτός από τη διακοπή χωρίς μάσκα (NMI)
Control	Ελέγχει τη στοίβα (stack) που χρησιμοποιείται και το επίπεδο προνομίων για εκτέλεση λογισμικού, όταν ο επεξεργαστής βρίσκεται σε λειτουργία νήματος (Thread mode)

Στην Εικ.7 φαίνεται ο διαχωρισμός της μνήμης του STM32. Η πλατφόρμα παράγει λίστες σφαλμάτων μόνο για το κομμάτι της SRAM, η οποία ξεκινάει από τη διεύθυνση 0x2000 0000 και έχει μέγεθος 20 KB, διότι αυτό το κομμάτι δεν έχει προστασία ανάγνωσης και εγγραφής. Οπότε

είναι δυνατή η εισαγωγή σφαλμάτων σε 5.120 λέξεις μνήμης (των 32 bit η καθεμία) ή 163.840 bits στο σύνολό τους.



Εικόνα 7. Μνήμη STM32 [16]

Αναλυτικότερες οδηγίες χρήσης και λεπτομέρειες λειτουργίας της πλατφόρμας υπάρχουν στο εγχειρίδιο χρήσης [22].

3.2 Περιγραφή εφαρμογής αυθεντικοποίησης CheckPIN

Για την αξιολόγηση της πλατφόρμας εισαγωγής σφαλμάτων δημιουργήθηκε μία εφαρμογή, η οποία λαμβάνει σαν είσοδο ένα PIN από το χρήστη, ελέγχει αν είναι σωστό και επιστρέφει μήνυμα αυθεντικοποίησης ή όχι στον χρήστη. Σκοπός της επίθεσης είναι να επιτευχθεί αυθεντικοποίηση με την εισαγωγή λανθασμένου PIN.

Στην πλακέτα στου STM32 εκτελείται κώδικας, υλοποιημένος σε C με βάση το [4], ο οποίος αποτελείται από δύο συναρτήσεις, όπως φαίνεται στην Εικ. 8.

Η πρώτη συνάρτηση byte2byteCheck ελέγχει το PIN, που πληκτρολογείται από το χρήστη και εάν είναι σωστό επιστέφει 1, διαφορετικά 0. Το τετραψήφιο PIN, που πληκτρολογείτε (userPIN), χωρίζεται σε τέσσερα byte και ελέγχεται, το κάθε byte, αν είναι ίσο με το κάθε byte του PIN που είναι αποθηκευμένο (cardPIN). Γίνεται, επίσης, έλεγχος για το αν ελέγχθηκαν και τα τέσσερα bytes. Έτσι για την ανίχνευση της εισαγωγής σφάλματος, που συνίσταται σε παράλειψη σύγκρισης, η συνάρτηση, σαν αντίμετρο, μετράει εάν ο αριθμός των συγκρίσεων, που έγιναν, είναι ίσος με τον αριθμό τέσσερα (4). Αν όλα είναι σωστά, η συνάρτηση, επιστρέφει 1, αλλιώς 0.

Η δεύτερη συνάρτηση, checkPIN, λαμβάνει σαν είσοδο το αποτέλεσμα της πρώτης (συνάρτησης) και επιστρέφει την απάντηση στο χρήστη, αν έχει αυθεντικοποιηθεί (authenticated) ή όχι. Χρησιμοποιούνται δύο σταθερές (True, False) τεσσάρων bytes και με συγκεκριμένες τιμές, διαφορετικές από 0 και 1, οι οποίες κωδικοποιούν booleans.

Κάποιο σφάλμα μπορεί να προκαλέσει την διακοπή εκτέλεσης του κώδικα (Hang) ή κάποια εξαίρεση (Exception) (βλέπε Πιν.5, 1 - 8). Στην περίπτωση εξαίρεσης, αναγνωρίζεται η αιτία της εξαίρεσης και παράγεται το ανάλογο μήνυμα, ώστε να κατηγοριοποιήσουμε τα αποτελέσματα που συλλέγονται κατά την εισαγωγή των σφαλμάτων. Τα μηνύματα αυτά φαίνονται στον Πίν. 5 [14, 15].

3.3 Πλεονεκτήματα – Μειονεκτήματα Πλατφόρμας

Όπως αναφέρθηκε και παραπάνω, η πλατφόρμα εισαγωγής σφαλμάτων ακολουθεί την emulation – based τεχνική, βασισμένη σε μία MCU για την υλοποίησή της. Η MCU που

χρησιμοποιήθηκε είναι ο STM32. Οπότε διατηρεί τα πλεονεκτήματά της τεχνικής που ακολουθείται, αλλά και τα πλεονεκτήματα της χρήσης της MCU. Παρακάτω φαίνονται τα πλεονεκτήματα και τα μειονεκτήματα της πλατφόρμας αναλυτικά

Πλεονεκτήματα	Μειονεκτήματα
• Εύκολη στην παραμετροποίηση (απλή γνώση MATLAB) • Χρησιμοποιείται τόσο για έλεγχο ασφάλειας, όσο και αξιοπιστίας • Το κόστος ανάπτυξης – δημιουργίας είναι σχετικά μικρό • Μπορεί να γίνει εισαγωγή σφάλματος σε οποιοδήποτε είδος κώδικα, όχι μόνο για έλεγχο PIN • Το GDB είναι πολύ καλό για μεγάλα προγράμματα	• Η γλώσσα του GDB δεν είναι και τόσο φιλική προς το χρήστη • Το GDB δεν είναι το βέλτιστο για μικρά προγράμματα • Μεγαλύτερος κώδικας υπο αξιολόγηση, περισσότερος χρόνος για την υλοποίηση

```c
#include "main.h"
extern int gdbwhileflag;
uint8_t  authenticated[4];
uint8_t cardPin[4] = {0x02, 0x07, 0x00, 0x0F};
int size = 4;

int byte2byteCheck(uint8_t* a1, uint8_t* a2)
{
    int i;
    int status = 0;
    int diff = 0;
    for(i = 0; i < size; i++) {//Check PIN
        if(a1[i] != a2[i]) {
            diff = 1;
        }
    }
    if(i+1 != size) {
        status =  0;
    }
    if(diff == 0) {
        status = 1;
    }
    else {
        status = 0;
    }
    return status;
    //gdbwhileflag = 0;
}

uint8_t* checkPIN( uint8_t* userPin)
{
    int i;
    int check = byte2byteCheck(userPin, cardPin);
    uint8_t const TRUE[4] = {0xF8, 0xF8, 0xF8, 0xF8};
    uint8_t const FALSE[4] = {0xF7, 0xF7, 0xF7, 0xF7};
    gdbwhileflag = 1;
        if( check == 1) {// Authentication();
            for(i=0;i<4;i++){
                authenticated[i] = TRUE[i];
            }
       } else {
    gdbwhileflag = 0;
            for(i=0;i<4;i++){
                authenticated[i] = FALSE[i];
            }
    }
    return authenticated;
}

```

Εικόνα 8. CheckPIN.c

Πίνακας 5. Πιθανά μηνύματα που επιστρέφει ο STM32

Α/Α	Όνομα	Περιγραφή	Μήνυμα που αποστέλλεται
1	NMI	(Non-maskable interrupt) διακοπή χωρίς μάσκα	00FFFFFF
2	HardFault	Όλες οι κατηγορίες σφαλμάτων	11FFFFFF
3	MemManage	Σφάλμα διαχείρισης μνήμης	22FFFFFF
4	BusFault	Σφάλμα πρόσβασης στη μνήμη	33FFFFFF
5	UsageFault	Απροσδιόριστη οδηγία ή παράνομη κατάσταση	44FFFFFF
6	SVCall	Κλήση υπηρεσίας συστήματος μέσω εντολών SWI	55FFFFFF
7	Debug	Παρακολούθηση εντοπισμού σφαλμάτων	66FFFFFF

Α/Α	Όνομα	Περιγραφή	Μήνυμα που αποστέλλεται
8	PendSV	Εκκρεμή αίτημα για υπηρεσία συστήματος	77FFFFFF
9	Hang	Διακοπή εκτέλεσης κώδικα	9F9F9F9F
10	No Access	Μη αυθεντικοποίηση	F7F7F7F7
11	Access	Αυθεντικοποίηση	F8F8F8F8

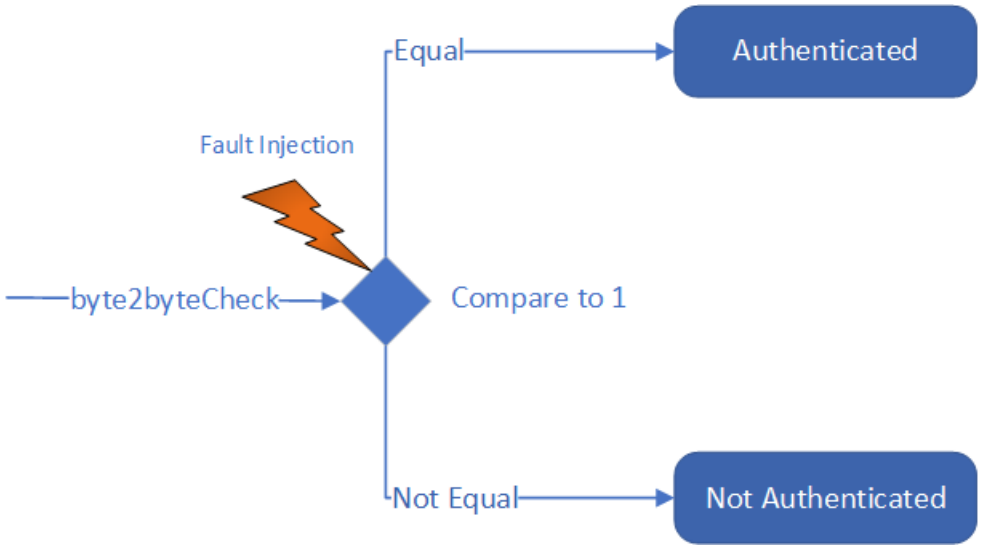
4 Διαδικασία Αξιολόγησης της Πλατφόρμας

Το πρώτο βήμα στην διαδικασία αξιολόγησης είναι η επιλογή του μέρους του κώδικα, στο οποίο θα εισαχθούν τα σφάλματα. Η εισαγωγή σφαλμάτων στο σύνολο του κώδικα θα οδηγούσε σε μία διαδικασία αξιολόγησης χρονοβόρα και σε μία λίστα πιθανών σφαλμάτων πολύ μεγάλου μεγέθους. Η επιλογή γίνεται με τη βοήθεια μίας μεταβλητής (gdbwhileflag), η οποία είναι μέρος του κώδικα. Η μεταβλητή αυτή αρχικοποιείται στον κώδικα με τιμή μηδέν (gdbwhileflag = 0), τίθεται ίση με τη μονάδα (gdbwhileflag = 1) στο σημείο ακριβώς πριν τη γραμμή που επιλέχθηκε να αξιολογηθεί και γίνεται πάλι μηδέν (gdbwhileflag = 0), στο σημείο που τελειώνει ο, προς αξιολόγηση, κώδικας.

Έτσι επιλέχθηκε το κομμάτι του κώδικα που βρίσκεται στη γραμμή τριάντα επτά (37), όπως φαίνεται και στην Εικ.8. Δηλαδή, η δήλωση υπό όρους (conditional statement) «if (check == 1)», στην οποία αντιστοιχούν επτά (7) εντολές (instructions) ή βήματα που πρέπει να εκτελεστούν από τον επεξεργαστή, όπως αυτά φαίνονται στον Πίν. 6. Επιλέχθηκε αυτό το κομμάτι του κώδικα, διότι υπάρχει μεγάλη πιθανότητα, εισάγοντας σφάλμα στη διαδικασία της σύγκρισης, να ακολουθείται ο κλάδος «Equal», ασχέτως του αποτελέσματος της σύγκρισης, όπως φαίνεται και στην Εικ. 9.

Πίνακας 6. Εντολές επεξεργαστή

Α/Α	Εντολή	Περιγραφή
1	ldr r3, [r7, #16]	Load the contents of the memory address [r7+16] to register r3
2	com r3, #1	Compare the value in r3 with 1
3	bne.n	Select branch if not equal
4	ldr r3, [pc, #64]	Load register r3 with a value from a PC-relative memory address
5	movs r2, #0	Copy the value of 0 into r2
6	str r2, [r3, #0]	Store the value of r2 at the memory address [r3+0]
7	movs r3, #0	Copy the value of 0 into r3



Εικόνα 9. Ροή εκτέλεσης κώδικα

Δεύτερο βήμα της διαδικασίας αξιολόγησης είναι η επιλογή του μοντέλου σφαλμάτων. Δοκιμάστηκαν όλα τα μοντέλα σφαλμάτων και από το σύνολο των μοντέλων που υποστηρίζει η πλατφόρμα, επιλέχθηκε το τρίτο μοντέλο σφαλμάτων. Υπενθυμίζεται ότι το τρίτο μοντέλο αφορά τη δημιουργία N σφαλμάτων, με τη μέθοδο bit-flip, σε K τυχαία bits από το σύνολο των bits των καταχωρητών ($1 \leq K \leq 736$) ή της μνήμης ($1 \leq K \leq 163.840$), σε τυχαίες χρονικές στιγμές. Στην επιλογή αυτή, συντέλεσε το γεγονός ότι, ακολουθώντας αυτό το μοντέλο, υπάρχει η δυνατότητα εισαγωγής σφαλμάτων στο σύνολο των καταχωρητών του Πίν. 5 και όχι μόνο σε ορισμένους από αυτούς.

Καθώς το πλήθος των σφαλμάτων που μπορεί να παραχθούν είναι πολύ μεγάλο, έπρεπε να καθοριστεί το μέγεθος του δείγματος (n), δηλαδή ο αριθμός των σφαλμάτων που θα πρέπει να εισαχθούν, έτσι ώστε να μπορούμε να καταλήξουμε σε ασφαλή συμπεράσματα, σύμφωνα με το [13]. Έτσι λοιπόν, αν και ο αρχικός πληθυσμός είναι πάντα πεπερασμένος, ο συνολικός αριθμός σφαλμάτων (N) μπορεί να είναι τεράστιος. Το N εξαρτάται από το κύκλωμα (αριθμός των στοιχείων της μνήμης που μπορούν να τροποποιηθούν), από το μοντέλο σφάλματος (αναμενόμενη πολλαπλότητα και κατανομή των λανθασμένων bit για ένα σφάλμα σε μια δεδομένη στιγμή) και από την εφαρμογή (αριθμός κύκλων του φόρτου εργασίας).

Στην παρούσα αξιολόγηση ο αρχικός πληθυσμός (B) προκύπτει από τον συνδυασμό του συνολικού αριθμού των bits των καταχωρητών ή της μνήμης, ήτοι 736 ή 163.840 bits αντίστοιχα, του αριθμού των κύκλων του φόρτου εργασίας, δηλαδή του αριθμού των εντολών που εκτελεί ο επεξεργαστής (instruction - i) και στην προκειμένη περίπτωση ισούται με επτά (7) (βλέπε Πίν. 6) και της πολλαπλότητας (m) των σφαλμάτων, δηλαδή του αριθμού των bits που θα επηρεάσει (ή διαφορετικά θα μεταβάλει την τιμή) κάθε σφάλμα. Στην πλατφόρμα μας, η πολλαπλότητα μπορεί να πάρει τιμές στο εύρος ένα (1) έως δέκα (10). Επιλέχθηκε αυτή η πολλαπλότητα διότι θεωρείται εξαιρετικά απίθανο να συμβούν πάνω από δέκα (10) σφάλματα ταυτόχρονα. Το σύνολο των τρόπων για να επιλεγούν m στοιχεία από ένα σύνολο B στοιχείων, στην ουσία ο συνολικός αριθμός σφαλμάτων (N), υπολογίζεται με τη βοήθεια του διωνυμικού συντελεστή, σύμφωνα με την εξίσωση (1).

$$N = \frac{B!}{m!\,(B-m)!} \quad (1)$$

Οπότε το n (ο αριθμός των σφαλμάτων που επιλέγονται τυχαία για εισαγωγή) μπορεί να υπολογιστεί σύμφωνα με την εξίσωση (2) και εξαρτάται από:

- τον αρχικό πληθυσμό (N)
- το εκτιμώμενο ποσοστό (p) των ατόμων στον πληθυσμό που έχουν ένα δεδομένο χαρακτηριστικό (π.χ. την εκτιμώμενη πιθανότητα σφαλμάτων που οδηγούν σε αποτυχία). Αυτή η παράμετρος καθορίζει το τυπικό σφάλμα και κυμαίνεται από 0 έως 1. Έχει αποδειχθεί ότι, για p = 0,5 επιτυγχάνεται η ορθή επιλογή μεγέθους δείγματος, το οποίο επαρκεί για να διασφαλιστεί το αναμενόμενο περιθώριο σφάλματος με το αναμενόμενο επίπεδο εμπιστοσύνης, ανεξάρτητα από την πραγματική τιμή της αναλογίας.
- το περιθώριο σφάλματος (e), το οποίο είναι ίσως ο πιο ευαίσθητος παράγοντας της εξίσωσης. Η μείωση αυτού του περιθωρίου αυξάνει κατά πολύ το απαιτούμενο μέγεθος δείγματος. Δίνεται ως ποσοστό και μπορεί αν πάρει τιμές από 0 έως 1, πχ. e = 0,05 αντιστοιχεί σε 5% περιθώριο σφάλματος.
- το επίπεδο εμπιστοσύνης (t). Αυτό το επίπεδο είναι η πιθανότητα, ότι η ακριβής τιμή βρίσκεται στην πραγματικότητα εντός του διαστήματος σφάλματος. Υπολογίζεται σύμφωνα με τον στατιστικό πίνακα της Κανονικής κατανομής. Η αύξηση του επιπέδου εμπιστοσύνης (δηλαδή του t) έχει μικρότερο αντίκτυπο στο n, απ' ότι η μείωση του περιθωρίου σφάλματος (e).
- Τον αριθμό των εντολών (I - instructions) που εκτελούνται στον μικροεπεξεργαστή.

$$n = \frac{i * N}{1 + e^2 * \frac{(N-1)}{\left(t^2 * p * (1-p)\right)}} \quad (2)$$

Μετά τον υπολογισμό των δεδομένων της αξιολόγησης και αφού θέσαμε σαν επιθυμητό περιθώριο σφάλματος 0,9% (e = 0,009) και διάστημα εμπιστοσύνης 99% (t = 2,5758), καταλήξαμε στον παρακάτω πίνακα, όπου φαίνεται το απαιτούμενο μέγεθος δείγματος για κάθε πολλαπλότητα ανά περίπτωση.

Πίνακας 7. Μέγεθος δείγματος ανά πολλαπλότητα

Πολλαπλότητα	Μέγεθος δείγματος για καταχωρητές	Μέγεθος δείγματος για μνήμη
1	4117	20118
2	20259	20478
3	20477	20478
4	20478	20478
5	20478	20478
6	20478	20478
7	20478	20478
8	20478	20478
9	20478	20478
10	20478	20478

5 Αποτελέσματα

Τα αποτελέσματα ταξινομούνται σε κατηγορίες, ανάλογα με το μήνυμα που επιστρέφει ο μικροελεγκτής STM32 σε κάθε περίπτωση σφάλματος (βλέπε τα μηνύματα του Πιν.5). Τα μηνύματα κατατάσσονται στις κατηγορίες που φαίνονται στον παρακάτω πίνακα.

Πίνακας 8. Κατηγορίες αποτελεσμάτων

Κατηγορία	Περιγραφή
Access	Επιτυχής εισαγωγή σφάλματος
No Access	Ανεπιτυχής εισαγωγή σφάλματος
Exceptions	Εξαίρεση στον κώδικα
Hangs	Διακοπή εκτέλεσης

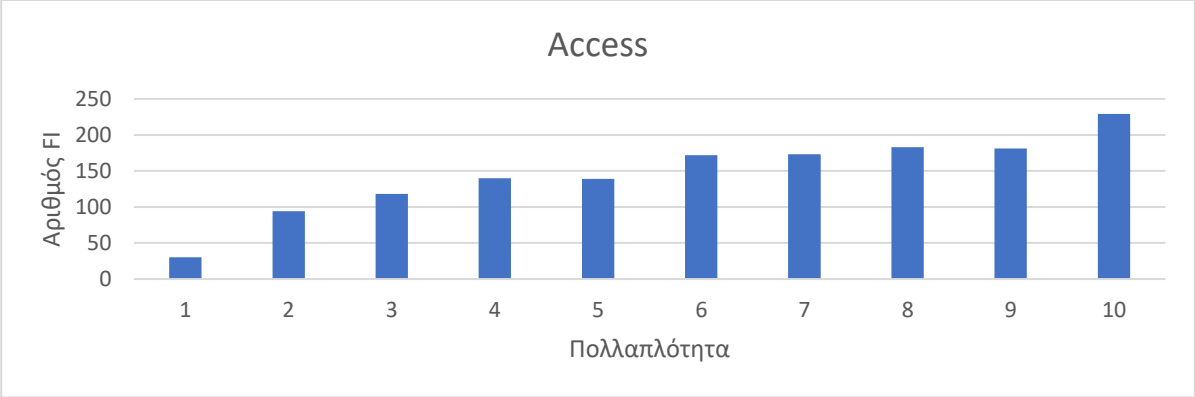
Η πρώτη κατηγορία (Access) σημαίνει ότι η εισαγωγή σφάλματος ήταν επιτυχής - δηλαδή επηρέασε την λειτουργία της εφαρμογής - και επιτεύχθηκε η αυθεντικοποίηση του χρήστη με λανθασμένο κωδικό (password). Η δεύτερη κατηγορία (No Access) σημαίνει ότι η εισαγωγή σφάλματος ήταν ανεπιτυχής - δηλαδή δεν επηρέασε την λειτουργία της εφαρμογής - και δεν επιτεύχθηκε η αυθεντικοποίηση του χρήστη με λανθασμένο κωδικό. Η επόμενη κατηγορία (Exceptions) αφορά στα μηνύματα ένα (1) έως και οκτώ (8) του Πίν.5, που έχουν να κάνουν με τις εξαιρέσεις κατά την εκτέλεση του κώδικα και σημαίνει ότι η εισαγωγή του σφάλματος οδήγησε τον επεξεργαστή σε εξαίρεση και ακύρωση της εκτέλεσης του υπόλοιπου κώδικα. Τέλος, η τελευταία κατηγορία (Hangs) σημαίνει ότι η εισαγωγή του σφάλματος διέκοψε την εκτέλεση του κώδικα στον επεξεργαστή, χωρίς την επιστροφή κάποιου μηνύματος εξαίρεσης.

Τα αποτελέσματα της εισαγωγής σφάλματος στους καταχωρητές, φαίνονται στον πίνακα που ακολουθεί (Πίν. 9). Παρατηρούμε ότι οι επιτυχείς εισαγωγές σφαλμάτων (κατηγορία Access) αυξάνονται, όσο αυξάνεται και η πολλαπλότητα των σφαλμάτων, δηλαδή ο αριθμός των bits, στα οποία γίνεται το bit-flip (Εικ. 10). Το ποσοστό επιτυχίας των σφαλμάτων αυξάνεται από 0,1% σε 1%, με την αύξηση της πολλαπλότητας.

Πίνακας 9. Συνολικά αποτελέσματα εισαγωγής σφάλματος

Category	Access		No Access		Exceptions		Hangs	
Fault Multiplicity	Successful fi in Registers	Successful fi in Memory	Unsuccessful fi in Registers	Unsuccessful fi in Memory	Exceptions in Registers	Exceptions in Memory	Hangs in Registers	Hangs in Memory
1	30	9	18956	18128	1202	1	812	2862
2	94	5	18016	18711	2188	2	702	2282
3	118	15	16897	18996	3115	4	870	1985
4	140	6	15831	17060	3611	8	1418	3926
5	139	13	15026	18032	4073	5	1762	2950
6	172	18	14490	17230	4650	15	1688	3737
7	173	22	13995	18727	4841	9	1991	2242
8	183	20	13611	18405	4984	13	2222	2562
9	181	23	12882	18753	5390	19	2547	2205
10	229	25	12656	18727	5113	12	3002	2236

Επίσης, αυξάνεται και ο αριθμός των σφαλμάτων, που προκαλούν εξαίρεση (στήλη Exceptions) ή διακοπή εκτέλεσης κώδικα (στήλη Hangs), καθώς αυξάνει ο αριθμός των καταχωρητών που επηρεάζονται (αύξηση πολλαπλότητας) (Εικ. 13 & 15).

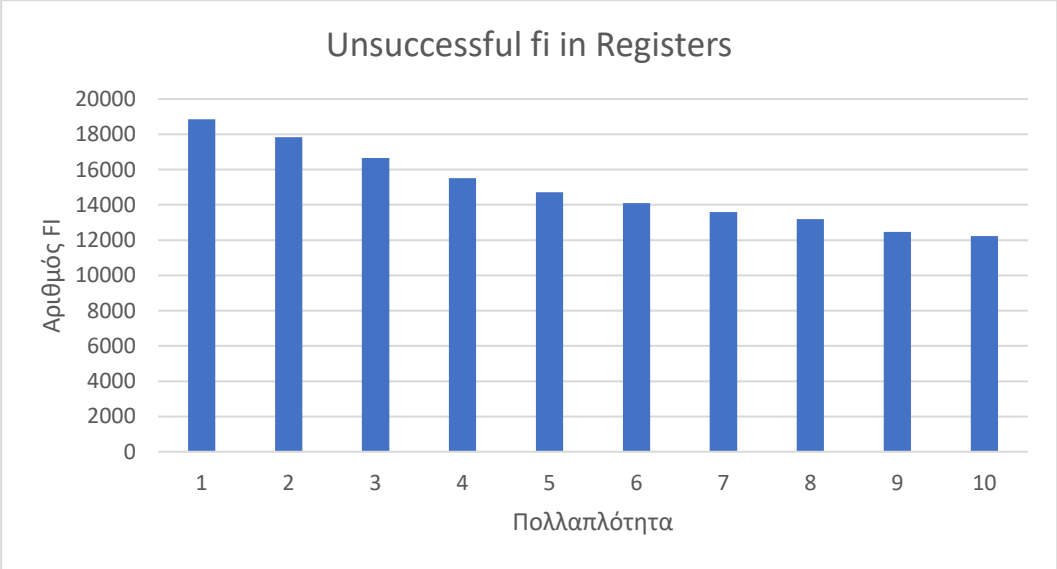


Εικόνα 10. Αριθμός επιτυχών fi στους καταχωρητές

Το ποσοστό των σφαλμάτων στους καταχωρητές, που οδηγούν σε αυθεντικοποίηση με λάθος password, αυξάνεται από 0,15% για πολλαπλότητα 1, σε 1% για πολλαπλότητα 10, όπως φαίνεται στην Εικ. 10.

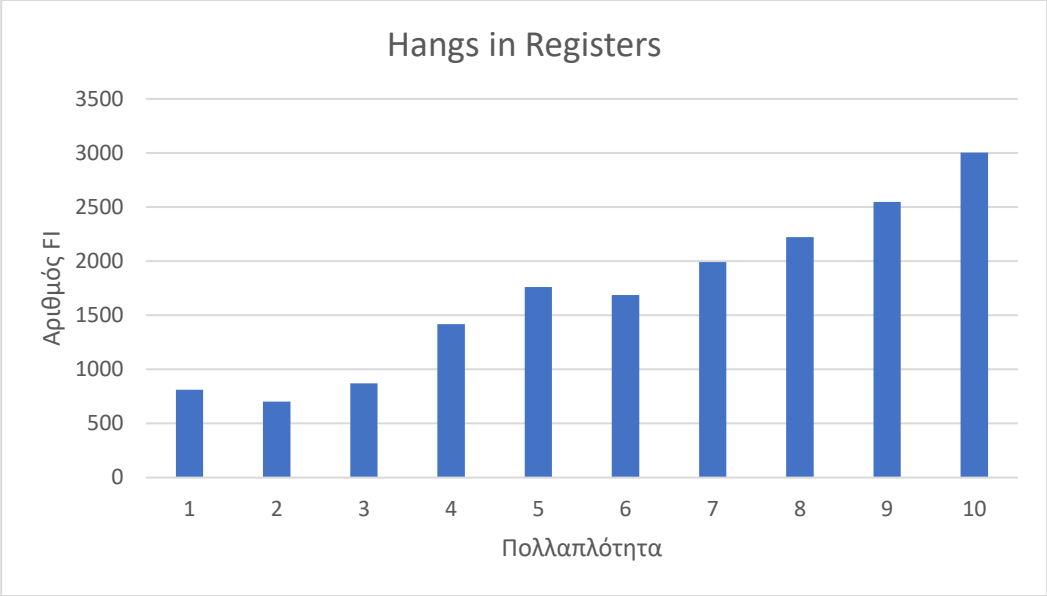
Τα αποτελέσματα είναι λογικά και αναμενόμενα, αφού καθώς αυξάνει η πολλαπλότητα των σφαλμάτων, αυξάνει και ο αριθμός των καταχωρητών που επηρεάζονται σε κάθε εισαγωγή σφάλματος, οπότε αυξάνεται και η πιθανότητα να επηρεαστεί το αποτέλεσμα της εκτέλεσης του κώδικα, από την εισαγωγή του σφάλματος.

Το ποσοστό των σφαλμάτων στους καταχωρητές, που δεν επηρέασε τον μικροελεγκτή, μειώνεται από 89,7% για πολλαπλότητα 1, σε 58,2% για πολλαπλότητα 10, όπως φαίνεται στην Εικ. 11.



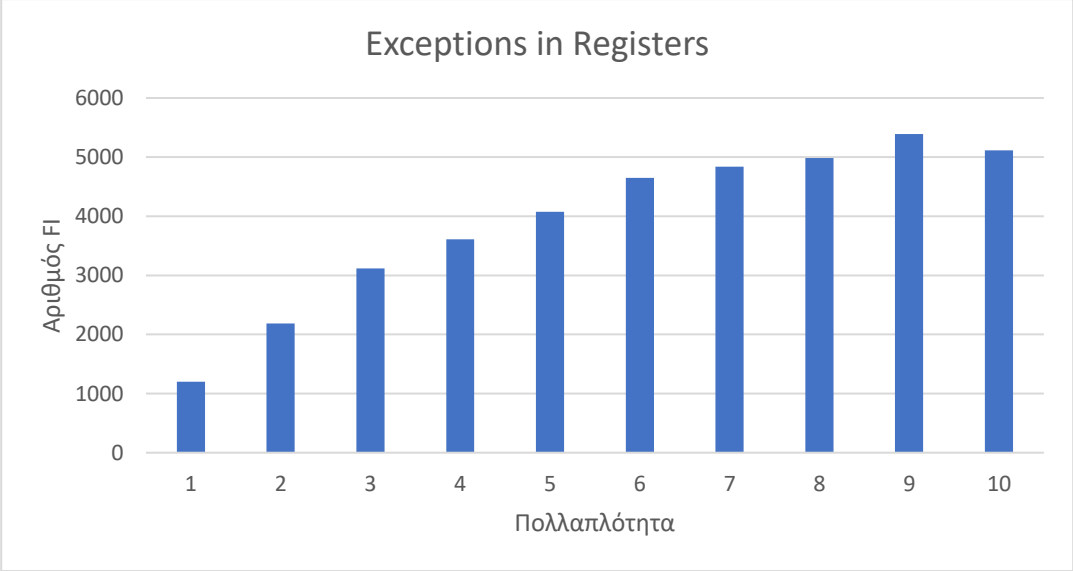
Εικόνα 11. Αριθμός ανεπιτυχών fi στους καταχωρητές

Το ποσοστό των σφαλμάτων στους καταχωρητές, που οδηγούν τον μικροελεγκτή σε διακοπή εκτέλεσης του κώδικα, αυξάνεται από σχεδόν 3,9% για πολλαπλότητα 1, σε 14,3% για πολλαπλότητα 10, όπως φαίνεται στην Εικ. 12.



Εικόνα 12. Αριθμός διακοπών κώδικα από fi στους καταχωρητές

Το ποσοστό των σφαλμάτων στους καταχωρητές που οδηγούν σε εξαίρεση και επαναφορά του μικροελεγκτή, αυξάνεται από 5,7% για πολλαπλότητα 1, σε 24,3% για πολλαπλότητα 10, όπως φαίνεται στην Εικ. 13.

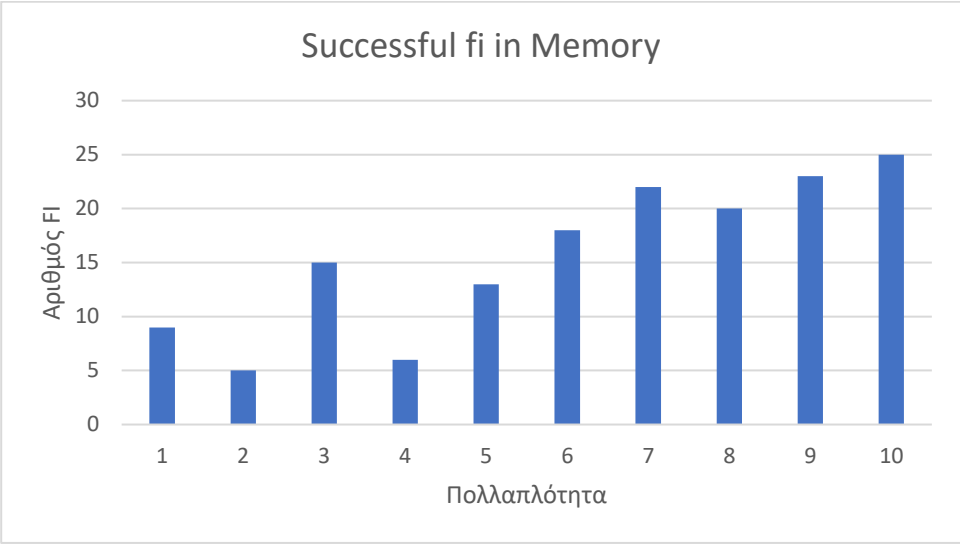


Εικόνα 13. Αριθμός εξαιρέσεων από fi στους καταχωρητές

Σε ότι αφορά στην εισαγωγή σφαλμάτων στη μνήμη, παρατηρούμε ότι υπήρξαν εισαγωγές που οδήγησαν σε αυθεντικοποίηση με λάθος password (κατηγορία Access), αλλά παρατηρήθηκε ότι το ποσοστό τους ήταν μικρότερο της τάξης του 0,1%, όπως φαίνεται στην Εικ. 14.

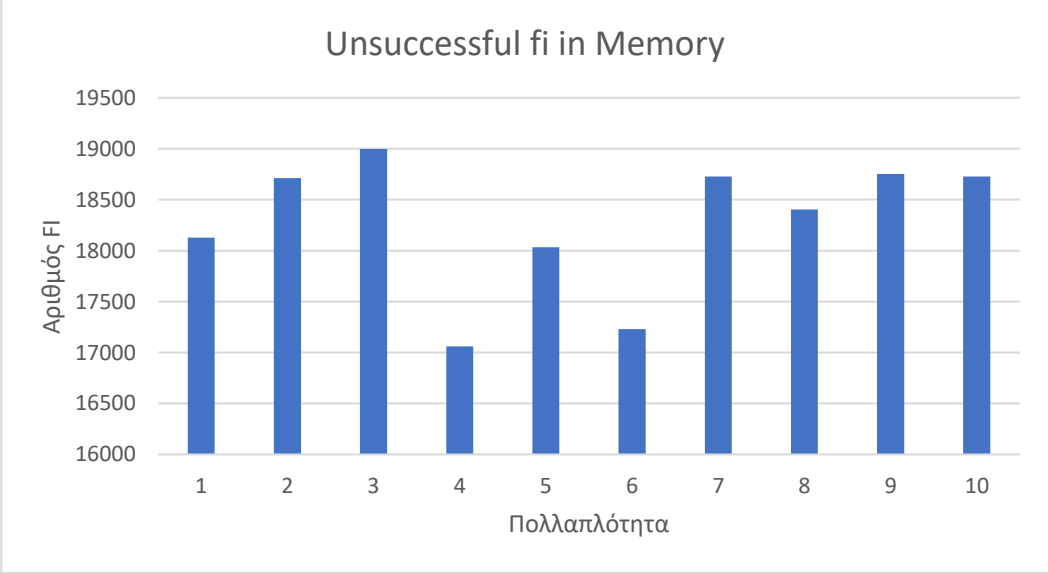
Επίσης, και εδώ παρατηρείτε αύξηση του αριθμού των σφαλμάτων, που καταλήγουν σε εξαίρεση ή διακοπή εκτέλεσης κώδικα (κατηγορίες Exceptions και Hangs αντίστοιχα) στο σύνολό τους, καθώς αυξάνει ο αριθμός των θέσεων μνήμης που επηρεάζονται.

Και εδώ τα αποτελέσματα είναι αναμενόμενα. Αν και η αυθεντικοποίηση με λανθασμένο password εμφανίζεται με πολύ μικρό ποσοστό, παρ' όλα αυτά φαίνεται ότι το αποτέλεσμα επηρεάζεται από την εισαγωγή σφαλμάτων, αλλά όχι στο βαθμό που επηρεάζεται από την εισαγωγή σφάλματος στους καταχωρητές.



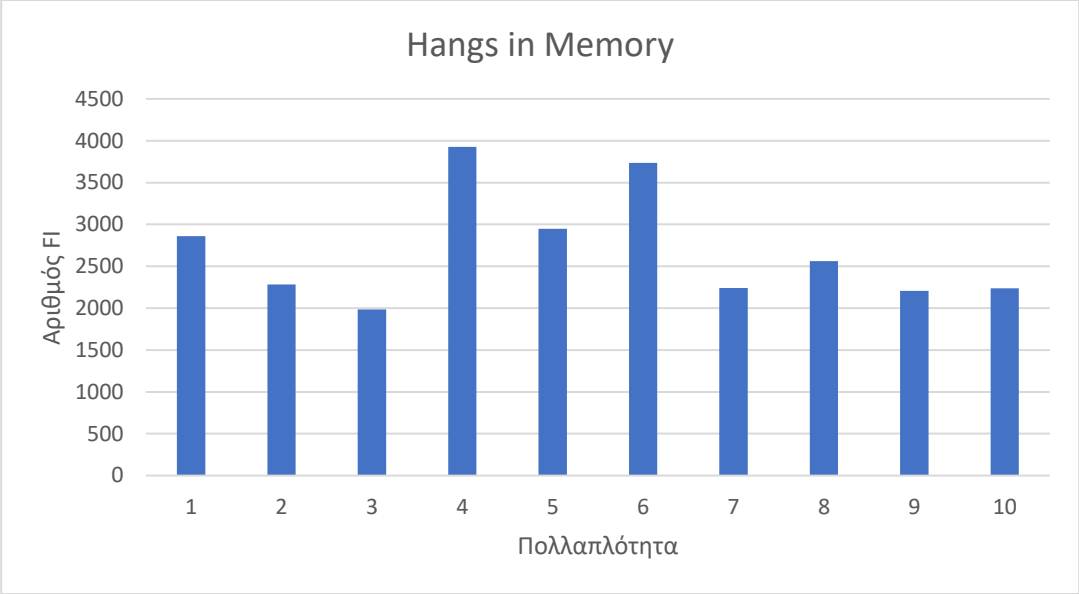
Εικόνα 14.Αριθμός επιτυχών fi στη μνήμη

Το ποσοστό των σφαλμάτων στη μνήμη, που δεν επηρέασε τον μικροελεγκτή, δεν παρουσιάζει κάποια ομαλότητα και κυμαίνεται από 81,2% για πολλαπλότητα 4 το ελάχιστο, έως 90,4% για πολλαπλότητα 3 το μέγιστο, όπως φαίνεται στην Εικ. 15.



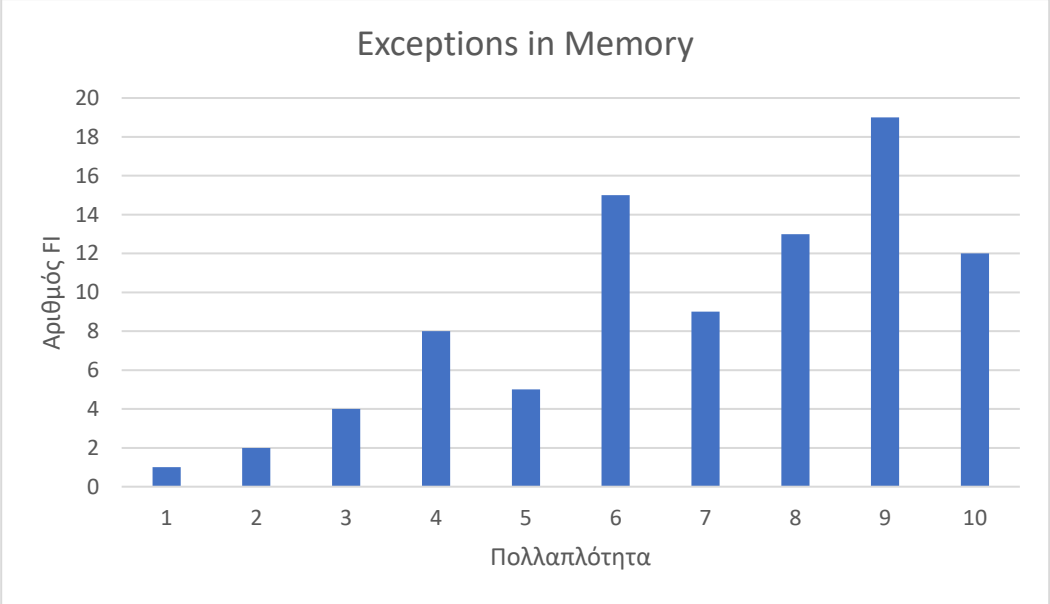
Εικόνα 15. Αριθμός ανεπιτυχών fi στη μνήμη

Το ποσοστό των σφαλμάτων στη μνήμη, που οδηγούν τον μικροελεγκτή σε διακοπή εκτέλεσης του κώδικα, δεν παρουσιάζει κάποια ομαλότητα και κυμαίνεται από 9,5% για πολλαπλότητα 3 το ελάχιστο, σε 18,7% για πολλαπλότητα 4 το μέγιστο, όπως φαίνεται στην Εικ. 16.



Εικόνα 16. Αριθμός διακοπών κώδικα από fi στη μνήμη

Το ποσοστό των σφαλμάτων στη μνήμη που οδηγούν σε εξαίρεση και επαναφορά του μικροελεγκτή, αυξάνεται από σχεδόν 0% για πολλαπλότητα 1, σε 0,005% για πολλαπλότητα 10, όπως φαίνεται στην Εικ. 17.



Εικόνα 17. Αριθμός εξαιρέσεων από fi στη μνήμη

Γενικά παρατηρείται ότι όσο αυξάνεται η πολλαπλότητα, δηλαδή ο αριθμός των bits στα οποία γίνεται το bitflip, τόσο αυξάνονται και οι επιτυχημένες προσπάθειες – επιθέσεις. Επίσης ότι η μνήμη φαίνεται να είναι πιο ανθεκτική στην εισαγωγή των σφαλμάτων, αφού με τον ίδιο αριθμό εισαγόμενων σφαλμάτων, εμφανίζονται λιγότερα επιτυχή αποτελέσματα.

6 Συμπεράσματα – Προτάσεις

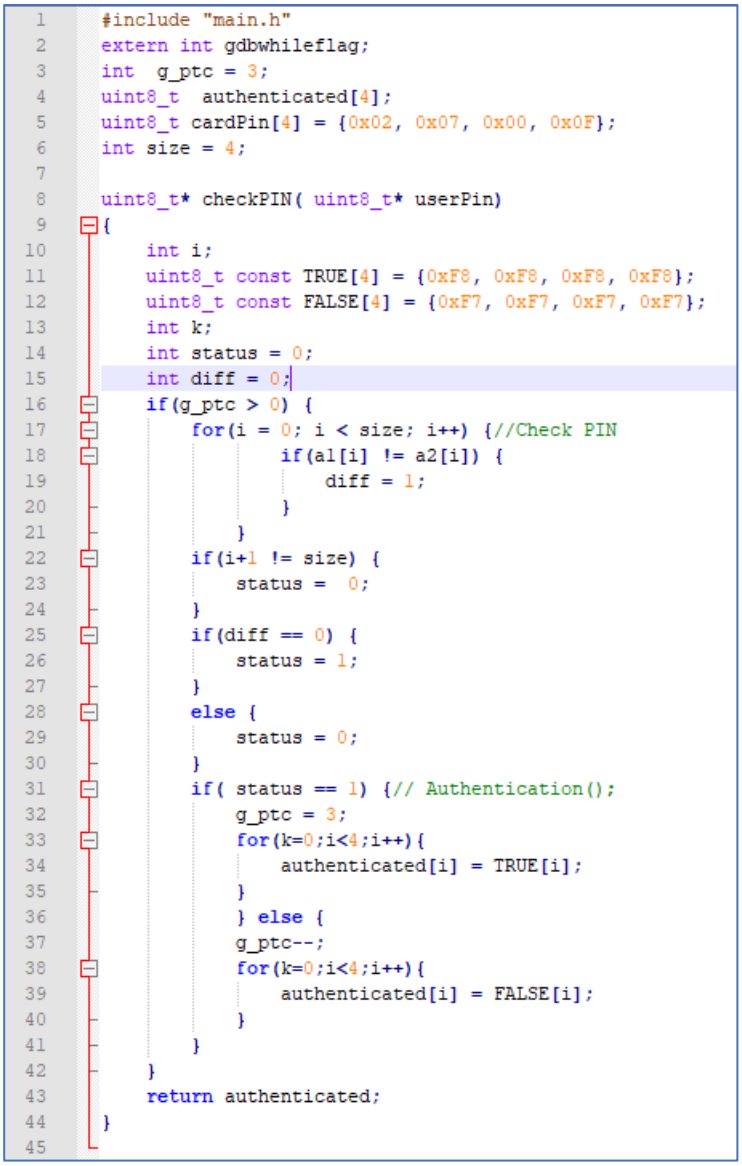
Όπως αποδείχτηκε κατά την αξιολόγηση της ασφάλειας της εφαρμογής αυθεντικοποίησης CHECKPIN, η πιθανότητα επίτευξης αυθεντικοποίησης χρήστη με λανθασμένο password είναι αρκετά μεγάλη. Παρατηρώντας καλύτερα των κώδικα, μπορούμε να βρούμε κι άλλα σημεία, τα οποία είναι εξίσου ευάλωτα σε τέτοιου είδους επιθέσεις, όπως πχ. η γραμμή είκοσι (20) του προηγούμενου κώδικα «if (diff == 0)». Έτσι, ο κώδικας καθίσταται ευαίσθητος σε τέτοιου είδους επιθέσεις.

Επίσης, εύκολα συμπεραίνουμε ότι, η εισαγωγή σφάλματος στους καταχωρητές οδήγησε σε μεγαλύτερα ποσοστά επιτυχιών, απ' ότι η εισαγωγή σφαλμάτων στη μνήμη του μικροελεγκτή. Με άλλα λόγια η μνήμη του μικροελεγκτή παρουσιάζει περισσότερη ανοχή σε σφάλματα.

Παρ' όλα αυτά και σύμφωνα με το [4], υπάρχουν διάφορες τεχνικές και αντίμετρα για τη βελτίωση της ασφάλειας του κώδικα. Προσθέτοντας, ως αντίμετρο, την καταμέτρηση των προσπαθειών αυθεντικοποίησης, καθώς και την συγχώνευση των δύο συναρτήσεων σε μία, επιτυγχάνουμε καλύτερη ασφάλεια και ανοχή στα σφάλματα, όπως φαίνεται παρακάτω.

Προσθέτοντας, λοιπόν μία σταθερά (g_ptc) που μειώνεται σε κάθε λάθος προσπάθεια, όπως φαίνεται στην Εικ. 18 και δίνοντας αρχική τιμή ίση με τρία (3), δίνουμε την δυνατότητα τριών προσπαθειών εισαγωγής password. Εάν ξεπεραστεί ο αριθμός των προσπαθειών, τότε το λογισμικό δεν ελέγχει καθόλου το password και επιστρέφει μήνυμα μη αυθεντικοποίησης. Στην περίπτωση που το παραπάνω αντίμετρο συνδυαστεί και με την εφαρμογή ακόμα αυστηρότερων αντιμέτρων, όπως ο αποκλεισμός του χρήστη για ορισμένο χρόνο, αυξάνεται κατά πολύ η ασφάλεια και η ανοχή στα σφάλματα.

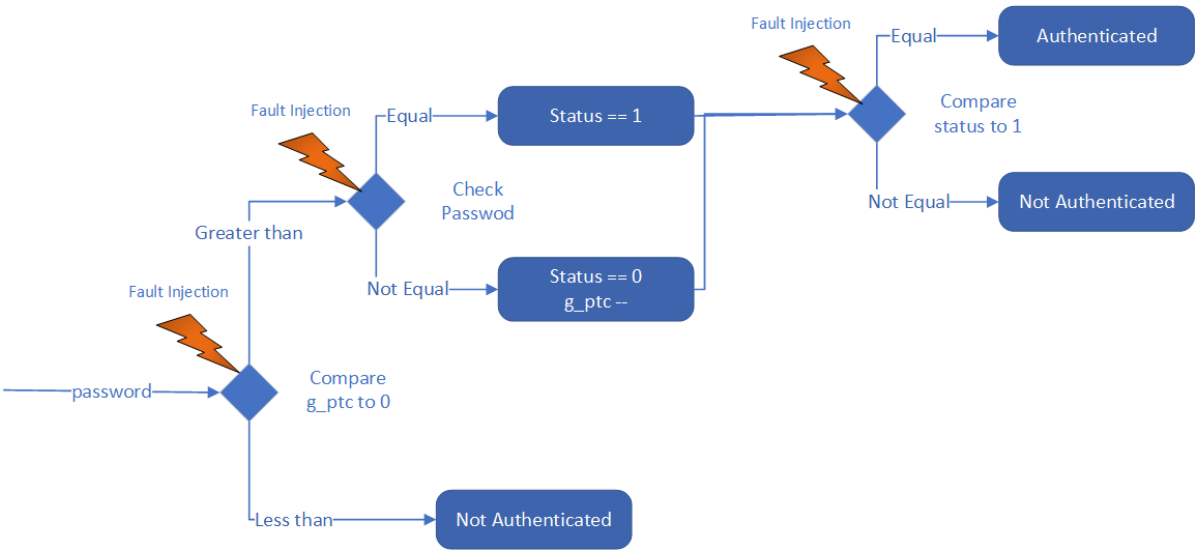
Συγχωνεύοντας τις δύο συναρτήσεις σε μία μειώνεται η πιθανότητα εισαγωγής σφάλματος, διότι θα πρέπει να εισαχθούν σφάλματα σε περισσότερους από δύο κόμβους αποφάσεων, όπως φαίνεται στην Εικ. 19.



```c
#include "main.h"
extern int gdbwhileflag;
int  g_ptc = 3;
uint8_t  authenticated[4];
uint8_t cardPin[4] = {0x02, 0x07, 0x00, 0x0F};
int size = 4;

uint8_t* checkPIN( uint8_t* userPin)
{
    int i;
    uint8_t const TRUE[4] = {0xF8, 0xF8, 0xF8, 0xF8};
    uint8_t const FALSE[4] = {0xF7, 0xF7, 0xF7, 0xF7};
    int k;
    int status = 0;
    int diff = 0;
    if(g_ptc > 0) {
        for(i = 0; i < size; i++) {//Check PIN
                if(a1[i] != a2[i]) {
                    diff = 1;
                }
            }
        if(i+1 != size) {
            status =  0;
        }
        if(diff == 0) {
            status = 1;
        }
        else {
            status = 0;
        }
        if( status == 1) {// Authentication();
            g_ptc = 3;
            for(k=0;i<4;i++){
                authenticated[i] = TRUE[i];
            }
            } else {
            g_ptc--;
            for(k=0;i<4;i++){
                authenticated[i] = FALSE[i];
            }
        }
    }
    return authenticated;
}
```

Εικόνα 18. CheckPIN.c με αντίμετρα



Εικόνα 19. Ροή εκτέλεσης κώδικα μετά τα αντίμετρα

7 Βιβλιογραφικές Πηγές

[1] Mohammad Eslami, Behnam Ghavami, Mohsen Raji, Ali Mahani. '*A survey on fault injection methods of digital integrated circuits*'. *Integration*, Volume 71, Pages 154-163, ISSN 0167-9260 (2020). Available at: https://doi.org/10.1016/j.vlsi.2019.11.006.

[2] Yuce B., Schaumont P. & WittemanM. "*Fault Attacks on Secure Embedded Software: Threats, Design, and Evaluation*". J Hardw Syst Secur 2, Pages 111–130 (2018). Available at: https://doi.org/10.1007/s41635-018-0038-1

[3] Papadimitriou A., Nomikos K., Psarakis M.,Aerabi E., Hely D. "*You can detect but you cannot hide: Fault Assisted Side Channel Analysis on Protected Software-based Block Ciphers*," 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Frascati, Italy, 2020, pp. 1-6, doi: 10.1109/DFT50435.2020.9250870.

[4] L. Dureuil, G. Petiot, M.-L. Potet, T.-H. Le, A. Crohen, Ph. de Choudens. *"FISSC: a Fault Injection and Simulation Secure Collection"*. In International Conference on Computer Safety, Reliability and Security (SAFECOMP), 2016. pdf et bibtex

[5] Ziade, Haissam & Ayoubi, Rafic & Velazco, R.. (2004). "*A Survey on Fault Injection Techniques"*. Int. Arab J. Inf. Technol.. 1. 171-186.

[6] Benso, Alfredo & Di Carlo, Stefano. (2011). "*The Art of Fault Injection*. Control Engineering and Applied Informatics". 13. 9-18.

[7] J. K. R. Sastry, J. S. Bhanu and K. SubbaRao, "*Attacking embedded systems through fault injection*," 2011 2nd National Conference on Emerging Trends and Applications in Computer Science, Shillong, 2011, pp. 1-5, doi: 10.1109/NCETACS.2011.5751374.

[8] P. Berthomé, K. Heydemann, X. Kauffmann-Tourkestansky and J. -. Lalande, *"High Level Model of Control Flow Attacks for Smart Card Functional Security,"* 2012 Seventh International Conference on Availability, Reliability and Security, Prague, Czech Republic, 2012, pp. 224-229, doi: 10.1109/ARES.2012.79.

[9] Hélène Le Bouder, Thierno Barry, Damien Couroussé, Jean-Louis Lanet, Ronan Lashermes. *"A TemplateAttack Against VERIFY PIN Algorithms".* SECRYPT 2016, Jul 2016, Lsbonne, Portugal. pp.231- 238, 10.5220/0005955102310238. hal-01383143Arlat

[10] Panda, Sourav; Liu, Yuanzhen; Hancke, Gerhard P.; Qureshi, Umair M. 2020. *"Behavioral Acoustic Emanations: Attack and Verification of PIN Entry Using Keypress Sounds"* Sensors 20, no. 11: 3015. https://doi.org/10.3390/s20113015.

[11] S. J. Murdoch, S. Drimer, R. Anderson and M. Bond, *"Chip and PIN is Broken,"* 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 433-446, doi: 10.1109/SP.2010.33.

[12] A. Höller, A. Krieg, T. Rauter, J. Iber and C. Kreiner, *"QEMU-Based Fault Injection for a System-Level Analysis of Software Countermeasures Against Fault Attacks,"* 2015 Euromicro Conference on Digital System Design, Madeira, Portugal, 2015, pp. 530-533, doi: 10.1109/DSD.2015.79.

[13] R. Leveugle, A. Calvez, P. Maistri and P. Vanhauwaert, *"Statistical fault injection: Quantified error and confidence",* 2009 Design, Automation & Test in Europe Conference & Exhibition, Nice, 2009, pp. 502-506, doi: 10.1109/DATE.2009.5090716.

[14] Reference manual for STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs

[15] Cortex-M3 Technical Reference Manual (TRM) Revision: r1p1

[16] STM32F10xxx/20xxx/21xxx/L1xxxx Cortex®-M3 programming manual

[17] Datasheet for Medium-density performance line ARM®-based 32-bit MCU with 64or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces

[18] Berthier, M., Bringer, J., Chabanne, H., Le, T.H., Riviere, L., Servant, V.: *"Idea: Embedded fault injection simulator on smartcard"*. In: ESSoS. LNCS, vol. 8364, pp. 222{229. Springer (2014)

[19] J. Machemie, C. Mazin, J. Lanet and J. Cartigny, *"SmartCM a smart card fault injection simulator,"* 2011 IEEE International Workshop on Information Forensics and Security, Iguacu Falls, Brazil, 2011, pp. 1-6, doi: 10.1109/WIFS.2011.6123124.

[20] Sere, A., Lanet, J.L., Iguchi-Cartigny, J.: *"Evaluation of countermeasures against fault attacks on smart cards"*. International Journal of Security and Its Applications 5(2) (2011)

[21] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson and E. Encrenaz, *"Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller,"* 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, 2013, pp. 77-88, doi: 10.1109/FDTC.2013.9.

[22] User manual for FINAS (Fault Injection by NASoufis)

[23] C. H. Kim and J. Quisquater, *"Faults, Injection Methods, and Fault Attacks,"* in IEEE Design & Test of Computers, vol. 24, no. 6, pp. 544-545, Nov.-Dec. 2007, doi: 10.1109/MDT.2007.186.