Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών

Ανίχνευση κακόβουλων δυαδικών αρχείων με τη χρήση ευφυών τεχνικών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΙΟΣ Π. ΛΕΚΚΑΣ

Επιβλέπων : Γεώργιος Στάμου
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Γεώργιος Αλεξανδρίδης
Ε.ΔΙ.Π. Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών

Ανίχνευση κακόβουλων δυαδικών αρχείων με τη χρήση ευφυών τεχνικών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΙΟΣ Π. ΛΕΚΚΑΣ

Επιβλέπων : Γεώργιος Στάμου
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Γεώργιος Αλεξανδρίδης
Ε.ΔΙ.Π. Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Φεβρουαρίου 2019.

.....	.....	.....
Γεώργιος Στάμου Αναπληρωτής Καθηγητής Ε.Μ.Π.	Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

Δημήτριος Π. Λέκκας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι τεχνικές Μηχανικές Μάθησης διαδραματίζουν έναν ολοένα και μεγαλύτερο ρόλο στον τομέα της Ασφάλειας Υπολογιστών. Λόγω της δομής του προβλήματος, η μεγαλύτερη πρόκληση στην αναγνώριση κακόβουλου λογισμικού είναι η εξαγωγή ωφέλιμων και ισχυρών χαρακτηριστικών τα οποία θα είναι ικανά να εκπαιδεύσουν επαρκώς τους ταξινομητές για να βελτιώσουν την προβλεπτική τους ισχύ. Στην βιβλιογραφία υπάρχουν εργασίες που ακολουθούν κατά κύριο λόγο δυο βασικές προσεγγίσεις στην ανίχνευση και κατηγοριοποίηση κακόβουλου λογισμικού· στην πρώτη η έμφαση δίνεται στην εκπαίδευση βαθιών νευρωνικών δικτύων χωρίς χρήση εξειδικευμένης γνώσης του τομέα της Ασφάλειας των Υπολογιστών, ενώ στη δεύτερη εξάγονται χαρακτηριστικά με ιδιαίτερα έντονες υπολογιστικές απαιτήσεις.

Ωστόσο, η εκπαίδευση νευρωνικών δικτύων αποδεικνύεται ανεπαρκής στην αναγνώριση πολυμορφικού και μεταμορφικού λογισμικού, ενώ η εξαγωγή περίπλοκων χαρακτηριστικών καθιστά τα μοντέλα απαγορευτικά για εφαρμογές πραγματικού χρόνου. Στο πλαίσιο της εργασίας προτείνεται η εξαγωγή ενός συνόλου χαρακτηριστικών που είναι ικανά να αναδείξουν επαρκώς τις προθέσεις ενός εκτελέσιμου αρχείου, ενώ ταυτόχρονα η εξαγωγή τους γίνεται με αρκετά αποτελεσματικό τρόπο.

Παράλληλα, εξετάζονται διάφορα σύνολα χαρακτηριστικών πού έχουν προταθεί στη βιβλιογραφία και ενσωματώνονται πρωτότυπες προσθήκες που βελτιώνουν αισθητά την απόδοση των ταξινομητών. Αξιολογείται πρακτικά και σημασιολογικά το κέρδος πληροφορίας των συνόλων χαρακτηριστικών που εξάγονται και επιλέγονται τα τελικά χαρακτηριστικά μέσω ενός άπληστου αλγόριθμου πρόσθιας βηματικής επιλογής.

Στη συνέχεια αξιολογείται η επίδοση διάφορων μοντέλων επιβλεπόμενης μάθησης και δίνεται έμφαση σε εκείνα τα οποία εκπαιδεύονται μέσω κατασκευής δεντρικής δομής. Μετά από εκτενή ανάλυση της ολικής αρχιτεκτονικής μηχανικής μάθησης του προβλήματος προτείνεται το τελικό σύστημα, η αξιοπιστία του οποίου αποδεικνύεται με βάση κάποιες κρίσιμες μετρικές αξιολόγησης. Τέλος επιβεβαιώνεται ότι η εξαγωγή των προτεινόμενων χαρακτηριστικών είναι αρκετά αποδοτική ώστε να εφαρμοστεί σε κατηγοριοποίηση πραγματικού χρόνου.

Λέξεις κλειδιά

κακόβουλο λογισμικό, μηχανική μάθηση, ενισχυμένα δέντρα, δέντρα απόφασης, μηχανές διανυσμάτων υποστήριξης, τυχαία δάση, εκτελέσιμα αρχεία Windows, συσκοτισμένο κακόβουλο λογισμικό

Abstract

Machine learning techniques play a continuously increasing role in the Computer Security sector. Due to the structure of the problem, the major challenge is the extraction of useful and powerful features that would help adequately train the classifiers in order to enhance their predictive ability. In literature, most works follow two major approaches in the detection and categorization of malware; in the first of them, the emphasis is placed in training deep neural networks without exploiting any domain-specific knowledge of the Computer Security area, while the second involves the extraction of computationally intensive features.

Nevertheless, neural network training proves to be inadequate in the detection of polymorphic or metamorphic malware, whilst the extraction of complicated features renders the use of those models prohibitive for real-time applications. In the context of the current thesis, the extraction of a robust set of features that are able to properly indicate the intensions of an executable file, is proposed, while their extraction is achieved in a highly efficient manner, at the same time.

Furthermore, several feature sets proposed in the literature are examined and novel additions, which lead to a substantial increase in the classifiers' performance, are also incorporated. The information gain of the extracted feature sets is evaluated practically and semantically and the final features are selected through a greedy forward stepwise selection algorithm.

The performance of several supervised learning classifiers is subsequently evaluated and emphasis is placed on those that are trained through the construction of a tree structure. After an extensive analysis of our machine learning pipeline, the final system is proposed, the performance of which is proven on a set of critical evaluation metrics. Finally, it is demonstrated that the extraction of the proposed features is efficient enough to perform real-time malware classification and detection.

Key words

malware, machine learning, gradient boosting, decision trees, support vector machines, random forest, PE files, obfuscated malware

Ευχαριστίες

Με την εκπόνηση της παρούσης διπλωματικής εργασίας ολοκληρώνεται ο κύκλος σπουδών μου στην Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Το όμορφο αυτό ταξίδι γνώσης έφτασε στο τέλος του και θα ήθελα πριν αναφερθώ στο περιεχόμενο της εργασίας να απευθύνω τις ευχαριστίες μου σε όλους όσους συνέβαλλαν σε αυτό το επίτευγμα.

Αρχικά, θα ήθελα να ευχαριστήσω τόσο τον κ. Γεώργιο Στάμου, Αναπληρωτή Καθηγητή Ε.Μ.Π και επιβλέποντα της παρούσας εργασίας, που μου έδωσε την δυνατότητα να ασχοληθώ με ένα τόσο σύγχρονο και ελκυστικό θέμα, όσο και τους κ.κ. Ανδρέα-Γεώργιο Σταφυλοπάτη και Νικόλαο Παπασπύρου, Καθηγητές Ε.Μ.Π. που μου έκαναν την τιμή να είναι μέλη της εξεταστικής επιτροπής. Μέσω της εκπόνησης της εργασίας αυτής είχα την ευκαιρία να διευρύνω τους επιστημονικούς μου ορίζοντες και να γνωρίσω τις ιδιαιτερότητες και προκλήσεις της έρευνας. Επίσης, θέλω να ευχαριστήσω όλο το προσωπικό του Εργαστηρίου Ευφυών Συστημάτων, και ιδιαίτερα τον κ. Γεώργιο Αλεξανδρίδη, Ε.ΔΙ.Π Ε.Μ.Π., που μέχρι την ολοκλήρωση της εργασίας ήταν πάντα διαθέσιμος να μου παρέχει οποιαδήποτε βοήθεια κατέστη αναγκαία. Η στήριξη αλλά και καθοδήγηση του αποδείχθηκε καθοριστική για την επιτυχία της παρούσας εργασίας.

Τέλος, κατά τη διάρκεια της φοιτητικής μου ζωής ήμουν τυχερός να έχω την αμέριστη στήριξη τόσο από την οικογένεια μου όσο και από τον κοινωνικό μου περίγυρο. Αυτή η στήριξη έπαιξε καταλυτικό ρόλο στην πορεία μου και ευελπιστώ να δικαίωσα όσους πίστεψαν σε μένα και συνέβαλλαν στην επίτευξη των στόχων και ονείρων μου.

Δημήτριος Π. Λέκκας,

Αθήνα, 26η Φεβρουαρίου 2019

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

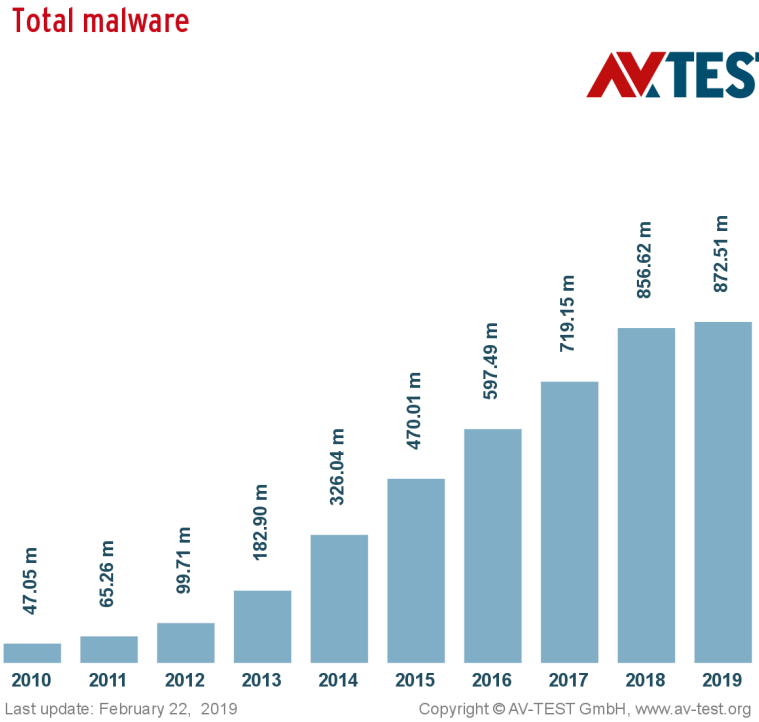
Κεφάλαιο 1

Εισαγωγή

Σύμφωνα με έρευνες και αναφορές της AV-TEST (Σχήμα 1.1), τα τελευταία χρόνια έχει παρατηρηθεί ραγδαία αύξηση στον αριθμό *κακόβουλων* (malware) αρχείων. Συγκεκριμένα, το 2018 ανιχνεύθηκαν 856 εκατομμύρια μολυσμένα αρχεία, αριθμός που αποτελεί τον μεγαλύτερο που έχει καταγραφεί μέχρι σήμερα [AVTE19]. Παράλληλα, έχει σημειωθεί αύξηση στον αριθμό των χρηστών που εμπιστεύονται και αποθηκεύουν μεγάλο μέρος των προσωπικών τους δεδομένων σε ηλεκτρονικές συσκευές. Το πρόβλημα όμως δεν περιορίζεται στους καθημερινούς χρήστες αλλά επεκτείνεται και σε επιχειρήσεις που φέρουν σημαντική ευθύνη για την ασφαλή διατήρηση των δεδομένων των χρηστών τους. Η εξάπλωση των υπολογιστών και του διαδικτύου στις ζωές μας έχει καταστήσει την εκτέλεση κακόβουλου κώδικα ικανή να επιφέρει σημαντικές οικονομικές, ψυχολογικές και ηθικές φθορές [Aike16].

Μια από τις σημαντικότερες προκλήσεις που αντιμετωπίζουν οι κατασκευαστές *αντικών λογισμικών* (antivirus) είναι ο τεράστιος όγκος δεδομένων και αρχείων που πρέπει να αναλύονται και να αξιολογούνται ως προς το περιεχόμενο και τις προθέσεις τους. Για παράδειγμα, προϊόντα της Microsoft που αφορούν την ανίχνευση κακόβουλου λογισμικού σε πραγματικό χρόνο είναι εγκατεστημένα σε πάνω από 160 εκατομμύρια υπολογιστές παγκοσμίως και εξετάζουν περίπου 700 εκατομμύρια υπολογιστές σε μηνιαία βάση. Το μεγάλο πλήθος παραγόμενων ανομοιογενών αρχείων καθιστά τη διαδικασία της ανάλυσης ιδιαίτερη δύσκολη και έντονη υπολογιστικά. Ένας βασικός λόγος της εμφάνισης μεγάλου πλήθους διαφορετικών αρχείων αποτελεί το γεγονός ότι οι συγγραφείς κακόβολου λογισμικού εισάγουν τεχνικές *πολυμορφισμού* (polymorphism) και *συσκότισης* (obfuscation) κώδικα προκειμένου να αποφύγουν την ανίχνευση. Συνεπώς, προγράμματα τα οποία ανήκουν στην ίδια οικογένεια κακόβολου λογισμικού τροποποιούνται συνεχώς ώστε να ερμηνεύονται ως διαφορετικά αρχεία.

Η βιομηχανία κακόβουλου λογισμικού είναι πλέον αρκετά οργανωμένη και συνεχώς αναπτύσσονται καινούριες τεχνικές προκειμένου το κακόβουλο λογισμικό να καταφέρνει να παρακάμπτει τις παραδοσιακές μεθόδους προστασίας. Συνεπώς, οι κατασκευαστές λογισμικού προστασίας από ιούς αναγκάζονται να επινοούν πρωτότυπους μηχανισμούς αντιμέτρων προκειμένου να ανταποκρίνονται στις ανάγκες των χρηστών για απρόσκοπτη και ασφαλή χρήση των μηχανημάτων τους. Ο μεγάλος όγκος κακόβουλων αρχείων μπορεί να είναι απαγορευτικός για τη χειροκίνητη αναγνώριση και κατηγοριοποίηση τους αλλά ταυτόχρονα είναι ιδιαίτερα ωφέλιμος για την εφαρμογή τεχνικών μηχανικής μάθησης. Συγκεκριμένα, η δυνατότητα συλλογής ενός μεγάλου συνόλου καταγεγραμμένων δειγμάτων επιτρέπει την εκτενή εκπαίδευση ευφυών ταξινομητών ικανών να κατηγοριοποιούν αρχεία με χαμηλή μεροληψία και υψηλά ποσοστά επιτυχίας.



Σχήμα 1.1: Πλήθος κακόβουλου λογισμικού των τελευταίων 10 ετών

Το πρόβλημα της *κατηγοριοποίησης κακόβουλου λογισμικού* (malware classification) παρουσιάζει ιδιαίτερο ενδιαφέρον καθώς είναι πιο δύσκολο και πολύπλοκο από την ανίχνευση και απαιτεί εξελιγμένες τεχνικές που θα αναδείξουν την συμπεριφορά των εκτελέσιμων αρχείων. Συνεπώς, μεγάλο μέρος της ερευνητικής κοινότητας μηχανικής μάθησης έχει στρέψει το ενδιαφέρον της προς την ταξινόμηση κακόβουλου λογισμικού χρησιμοποιώντας ένα ιδιαίτερα πληθωρικό σύνολο δεδομένων από την Microsoft [Past19]. Επιπλέον, η επίτευξη αποδοτικής ανάλυσης τεράστιου όγκου δεδομένων απαιτεί την ομαδοποίηση και ταξινόμηση των αρχείων σε επιμέρους κατηγορίες. Μια τέτοια ταξινόμηση θα μας επέτρεπε να αναλύσουμε τη συμπεριφορά κάθε κατηγορίας, μελετώντας μια πληθώρα δειγμάτων με σκοπό την ανάπτυξη εξειδικευμένων μηχανισμών αντιμετώπισης για κάθε οικογένεια.

1.1 Περιγραφή Προβλήματος

Στην παρούσα διπλωματική εργασία θα ασχοληθούμε με το πρόβλημα της κατηγοριοποίησης κακόβουλων εκτελέσιμων αρχείων του αρχέτυπου *Φορητού Εκτελέσιμου* (Portable Executable) που προορίζονται για εκτέλεση στο λειτουργικό σύστημα Windows. Συγκεκριμένα, στόχος είναι η κατασκευή ενός συστήματος που θα λαμβάνει ως είσοδο δυαδικά εκτελέσιμα αρχεία και θα αποφαίνεται σε ποιες οικογένειες κακόβουλου λογισμικού ανήκουν. Για την πρόβλεψη του συστήματος μας θα χρησιμοποιήσουμε ταξινομητές επιβλεπόμενης μηχανικής μάθησης που θα εκπαιδεύονται από σύνολα χαρακτηριστικών που θα προτείνουμε στα πλαίσια της εργασίας.

Γενικά, η ανάλυση κακόβουλου λογισμικού γίνεται συνήθως μέσω *στατικών* (static analysis) [Sami10, Ye07] και *δυναμικών* τεχνικών (dynamic analysis) [Will07]. Η στατική ανάλυση δεν απαιτεί την εκτέλεση του αρχείου για την κατηγοριοποίηση του αλλά εμπεριέχει μικρότερο μέγεθος πληροφορίας σε σχέση με τη δυναμική ανάλυση που φανερώνει τη συμπεριφορά του εκτελέσιμου ως προς διάφορους πόρους του συστήματος. Στην εργασία αυτή επικεντρωνόμαστε στη στατική ανάλυση και εμπνεόμαστε από την πλούσια βιβλιογραφία στον τομέα της στατικής ανάλυσης ώστε να εξάγουμε χαρακτηριστικά που θα εκπαιδεύουν επαρκώς μοντέλα μηχανικής μάθησης.

Οι αναλυτές κακόβουλου λογισμικού, μέσω της στατικής ανάλυσης, εξάγουν διάφορα χαρακτηριστικά από τη δομή και σημασιολογία του προγράμματος όπως *κωδικούς εντολών* (operation codes) [Sant13], *γραφήματα κλήσης συναρτήσεων* (function call graph) [Hu09], *υπογραφές συμβολοσειρών* (string signatures) [Grif09], *χαρακτηριστικά επικεφαλίδας φορητών εκτελέσιμων* (PE header) [Raff17], εξαρτήσεις μεταξύ *προγραμματιστικών διεπαφών* της εφαρμογής (Application Programming Interfaces) [Ye07] και *δυναμικές βιβλιοθήκες* (Dynamic Linked Libraries) [Naro15]. Τα παραπάνω χαρακτηριστικά έχουν αποδειχθεί ικανά να βοηθήσουν στην ανίχνευση κακόβουλου λογισμικού, ωστόσο φέρονται να είναι αδύναμα σε πλήθος τεχνικών που χρησιμοποιούνται από τους συγγραφείς κακόβουλων προγραμμάτων [Chri05].

Η συνολική λογική στην μοντελοποίηση της συμπεριφοράς ενός εκτελέσιμου αρχείου έγκειται στο γεγονός ότι μπορεί τα μεμονωμένα χαρακτηριστικά που θα προτείνουμε να εμπίπτουν σε κάποια τυχαία κατανομή αλλά η αθροιστική αξιοποίηση τους είναι ικανή να φανερώσει σημαντικά στοιχεία των προθέσεων του αρχείου. Αναλυτικότερα, θέλουμε να κατασκευάσουμε μοντέλα βασισμένα σε ευφυής τεχνικές τα οποία θα είναι ικανά να ταξινομήσουν κακόβουλα εκτελέσιμα αρχεία σε 9 διαφορετικές οικογένειες με υψηλά ποσοστά ακρίβειας και αυτοπεποίθησης. Συγκεκριμένα το σύστημα μας θα είναι σε θέση να αποφανθεί με πολύ *χαμηλή καθυστέρηση* (low-latency) σε ποία οικογένεια κακόβουλων αρχείων ανήκει το αρχείο εισόδου.

1.2 Προκλήσεις

Το πρόβλημα της κατηγοριοποίησης κακόβουλου λογισμικού χαρακτηρίζεται από κάποιες ιδιαίτερες προκλήσεις λόγω της συνεχώς εξελισσόμενης τεχνολογίας που χρησιμοποιείται από τους συγγραφείς κακόβουλου κώδικα. Μια από τις σημαντικότερες προκλήσεις είναι η εντατική προσπάθεια αποφυγής και παράκαμψης της ανίχνευσης του κακόβουλου λογισμικού από προγράμματα προστασίας. Οι δημιουργοί τέτοιων λογισμικών κατασκευάζουν πολυμορφικά αρχεία που αλλάζουν διαρκώς την εμφάνιση τους διατηρώντας ίδια την συμπεριφορά. Αυτό επιτυγχάνεται με διάφορες τεχνικές όπως μετονομασία καταχωρητών, κρυπτογράφηση φόρτου εργασίας εκτελέσιμου και προσθήκη έξτρα εντολών που δεν επηρεάζουν την τελική λειτουργία του προγράμματος [Wong06, Chri05]. Πολλές από αυτές τις τεχνικές θα αναλυθούν και θα επεξηγηθεί πως το σύστημα μας ανταποκρίνεται ικανοποιητικά στον εντοπισμό και την κατηγοριοποίηση τέτοιων δειγμάτων.

Η δομή και οι ιδιαιτερότητες του συνόλου δεδομένων που χρησιμοποιήσαμε για την εκπαίδευση των ταξινομητών εισήγαγε δύο ακόμα σημαντικές προκλήσεις. Αρχικά, για κάθε δείγμα (εκτελέσιμο αρχείο) παρέχεται το αρχείο συμβολικής γλώσσας και το δεκαεξαδικό αρχείο με περικομμένες όλες τις πληροφορίες και τα μεταδεδομένα που αφορούν την επικεφαλίδα του φορητού εκτελέσιμου. Η επικεφαλίδα των αρχείων χρησιμοποιείται ευρέως από αναλυτές κακόβουλου λογισμικού και έχει αποδειχθεί ότι αποτελεί ένα ιδιαίτερα χρήσιμο χαρακτηριστικό ενός εκτελέσιμου αρχείου ως προς την ερμηνεία της συμπεριφοράς του.

Επιπλέουν έχουν εφαρμοστεί μοντέλα μηχανικής μάθησης που χρησιμοποιούν αποκλειστικά τις πληροφορίες τις επικεφαλίδας και πετυχαίνουν υψηλά ποσοστά επιτυχημένης ανίχνευσης [Raff17, Wang09]. Οπότε, το πρόβλημα που έχουμε να αντιμετωπίσουμε είναι εν γένει πιο δύσκολο από την κλασσική κατηγοριοποίηση καθώς θα έχουμε στη διάθεση μας λιγότερες πληροφορίες για τα αρχεία σε σχέση με αυτές που θα είχαμε σε ένα πραγματικό σύστημα.

Επιπρόσθετα, η άνιση κατανομή των δειγμάτων ως προς τις επιμέρους οικογένειες κακόβουλου λογισμικού δυσκολεύει αισθητά την εκπαίδευση των ταξινομητών καθώς επηρεάζει την διαχωριστική τους ικανότητα ανά κλάση και συνεπώς την τελική προβλεπτική ισχύ του συστήματος. Το σύνολο δεδομένων που δημοσιεύτηκε από τη Microsoft [Past19], παρόλο που αποτελεί το πληρέστερο σύνολο δεδομένων σε ανίχνευση κακόβουλου λογισμικού, παρέχει ιδιαίτερα μικρό αριθμό δειγμάτων για κάποιες συγκεκριμένες κατηγορίες κακόβουλων εκτελέσιμων, με αποτέλεσμα να πρέπει να προσαρμόσουμε τα μοντέλα μας κατάλληλα. Η πρόκληση είναι η σωστή ταξινόμηση δειγμάτων που ανήκουν στις κλάσεις με αρκετά χαμηλή εκπροσώπηση από το σύνολο των δειγμάτων.

Τέλος, η φύση του προβλήματος που απαιτεί την ανάλυση τεράστιων αρχείων για την εξαγωγή χαρακτηριστικών καθιστά το πρόβλημα υπολογιστικά έντονο και ενέχει σημαντικούς συμβιβασμούς όσων αφορά την απόδοση και την ταχύτητα ταξινόμησης. Στην εργασία αυτή ακολουθούμε μια προσέγγιση που μας επιτρέπει να πραγματοποιούμε γρήγορες εξαγωγές χαρακτηριστικών και ταξινομήσεις ώστε το σύστημα μας να μπορεί να χρησιμοποιηθεί σε εφαρμογές πραγματικού χρόνου.

1.3 Δομή Εργασίας

Στην παρούσα διπλωματική εργασία αντιμετωπίζουμε την ανίχνευση και κατηγοριοποίηση κακόβουλου λογισμικού ως πρόβλημα εξαγωγής ουσιαστικών και ισχυρών χαρακτηριστικών ικανών να αναδείξουν επαρκώς την κακόβουλη συμπεριφορά κάθε οικογένειας. Στο Κεφάλαιο 2 εισάγουμε το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση και αιτιολόγηση των επιλογών μας στη συνέχεια της εργασίας. Συνοπτικά, επεξηγούνται οι βασικές αρχές λειτουργίας των ταξινομητών επιβλεπόμενης μάθησης που χρησιμοποιήσαμε, οι απαραίτητες γνώσεις που αφορούν το κομμάτι της Ασφάλειας των Υπολογιστών καθώς και οι τεχνικές προδιαγραφές των εκτελέσιμων αρχείων προς εξέταση. Το Κεφάλαιο 3 αποτελεί το σημαντικότερο τμήμα της διπλωματικής καθώς αναφέρει και επεξηγεί εκτενώς τα χαρακτηριστικά που εξάγαμε και χρησιμοποιήσαμε στο τελικό μας σύστημα. Συγκεκριμένα, περιγράφεται η πληροφορία που μας προσδίδει κάθε σύνολο χαρακτηριστικών και προτείνεται ένας αποδοτικός τρόπος επιλογής εκείνων που ενισχύουν καλύτερα την προβλεπτική ισχύ του τελικού ταξινομητή. Στο Κεφάλαιο 4 γίνεται μια σύντομη περιγραφή στο σύνολο δεδομένων που χρησιμοποιήθηκε, στις μετρικές αξιολόγησης των μοντέλων και ακολουθεί η σύγκριση των ταξινομητών και η παρουσίαση των τελικών αποτελεσμάτων. Τέλος, στο Κεφάλαιο 5 καταγράφονται τα βασικά συμπεράσματα της εργασίας καθώς και μελλοντικές ερευνητικές κατευθύνσεις όσων αφορά την κατηγοριοποίηση κακόβουλου λογισμικού.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο συγκεκριμένο κεφάλαιο στόχος είναι να καλύψουμε όλο το θεωρητικό υπόβαθρο πάνω στο οποίο θα βασιστούμε στα επόμενα κεφάλαια για να περιγράψουμε την υλοποίηση καθώς και τα κίνητρα πίσω από τις σχεδιαστικές μας επιλογές και αποφάσεις. Λόγω του αντικειμένου της διπλωματικής που βρίσκεται στην τομή δυο διαφορετικών γνωστικών αντικειμένων, θα αναφερθούμε αρχικά σε ότι πληροφορίες χρειάζεται να καλύψουμε σχετικά με τα χαρακτηριστικά και τις ιδιοτροπίες του κακόβουλου λογισμικού και στη συνέχεια θα επεκταθούμε στους αλγορίθμους μηχανικής μάθησης που χρησιμοποιήσαμε προκειμένου να πετύχουμε την σωστή κατηγοριοποίηση των αρχείων.

2.1 Κακόβουλο Λογισμικό

Το κακόβουλο λογισμικό αδιαμφισβήτητα αποτελεί μια από τις σημαντικότερες κατηγορίες απειλών των υπολογιστικών συστημάτων. Ένα πρόγραμμα θεωρείται κακόβουλο όταν εισάγεται σε ένα σύστημα, συνήθως κρυφά, με στόχο να παραβιαστεί η εμπιστευτικότητα, ακεραιότητα, ή διαθεσιμότητα των δεδομένων, των εφαρμογών, ή του λειτουργικού συστήματος του θύματος, ή να προκληθεί με οποιονδήποτε άλλο τρόπο ενόχληση ή αναστάτωση στο θύμα.

Λόγω της εκθετικής αύξησης της πληροφορίας και των διαθέσιμων προγραμμάτων έχει αναπτυχθεί ένα ευρύ φάσμα απειλών που καθιστά αναγκαία την κατηγοριοποίηση τους προκειμένου να αναπτύσσονται τεχνικές αντιμετώπισης προσαρμοσμένες στις ιδιότητες της κάθε κατηγορίας. Συγκεκριμένα, τα κακόβουλα αρχεία ταξινομούνται με βάση τους τρόπους με τους οποίους διαδίδεται ή εξαπλώνεται το κακόβουλο λογισμικό, και κατόπιν τους διάφορους τύπους ενεργειών ή *βλαπτικών φορτίων* (payloads) που χρησιμοποιούνται από τη στιγμή που το κακόβουλο λογισμικό έχει φτάσει στο στόχο του. Στους μηχανισμούς εξάπλωσης περιλαμβάνονται εκείνοι που χρησιμοποιούνται από ιούς, *σκουλήκια* (worms) και *Δούρειους ίππους* (Trojans). Στα φορτία περιλαμβάνονται η *αλλοίωση του συστήματος* (system corruption), τα *ρομπότ* (bots), το *ηλεκτρονικό ψάρεμα* (phising), το *κατασκοπευτικό λογισμικό* (spyware) και τα *κιτ υπερχρήστη* (rootkits).

2.1.1 Εκτελέσιμα Αρχεία Windows (PE)

Το *Φορητό Εκτελέσιμο* (Portable Executable) αποτελεί μορφή εκτελέσιμου αρχείου και *αντικείμενου κώδικα* (object code) που χρησιμοποιείται σε εκδόσεις 32-bit και 64-bit του λειτουργικού συστήματος Windows. Η μορφή PE είναι ουσιαστικά μια δομή δεδομένων που ενθυλακώνει όλες τις πληροφορίες που απαιτούνται από τον *φορτωτή* (loader) του λειτουργικού συστήματος Windows προκειμένου να διαχειριστεί και να εκτελέσει τον εκτελέσιμο κώδικα που περιέχεται στο αρχείο. Η συγκεκριμένη μορφή αρχείου αποτελεί τροποποιη-

μένη έκδοση της μορφής Common Object File Format (COFF) που χρησιμοποιούνταν σε συστήματα Unix. Η Microsoft έκανε τη μετάβαση στη μορφή PE με την εισαγωγή του λειτουργικού συστήματος Windows NT 3.1. Όλες οι μεταγενέστερες εκδόσεις των Windows, συμπεριλαμβανομένων των Windows 95/98/ME, υποστηρίζουν αυτή τη δομή αρχείου. Κάποιες επεκτάσεις που έχουν προστεθεί απο τη Microsoft περιλαμβάνουν τη μορφή PE .NET, την έκδοση 64-bit PE32+ και μία ειδική προδιαγραφή για τα Windows CE.

Δομή Αρχέτυπου PE

Το αρχέτυπο PE αποτελείται από ένα σύνολο επικεφαλίδων και τμημάτων που ορίζουν στον *δυναμικό συνδέτη* (dynamic linker) πως να αντιστοιχήσει το αρχείο στης μνήμη. Μια εκτελέσιμη εικόνα αποτελείται από διάφορες περιοχές, καθεμία από τις οποίες έχει διαφορετικές απαιτήσεις για προστασία μνήμης. Αυτή η ανάγκη για διαφοροποίηση δικαιωμάτων ανά περιοχή επιβάλλει η αρχή κάθε τμήματος να είναι ευθυγραμμισμένη σε *όρια σελίδας* (page boundary). Για παράδειγμα, τυπικά το τμήμα .text (περιέχει τον κώδικα του προγράμματος) αντιστοιχίζεται με δικαιώματα *εκτέλεσης* (execute) και *ανάγνωσης* (readonly), ενώ το τμήμα .data (περιέχει τις οικουμενικές μεταβλητές) αντιστοιχίζεται με δικαιώματα *ανάγνωσης και εγγραφής* (readwrite). Όμως, προκειμένου να εξασφαλιστεί εξοικονόμηση χώρου, τα διαφορετικά τμήματα δεν είναι ευθυγραμμισμένα στο δίσκο. Ο δυναμικός συνδέτης είναι υπεύθυνος να αντιστοιχίσει κάθε τμήμα στη μνήμη ατομικά και να εκχωρήσει τα σωστά δικαιώματα χρήσης σε κάθε περιοχή σύμφωνα με τις υποδείξεις που βρίσκονται στην επικεφαλίδα.

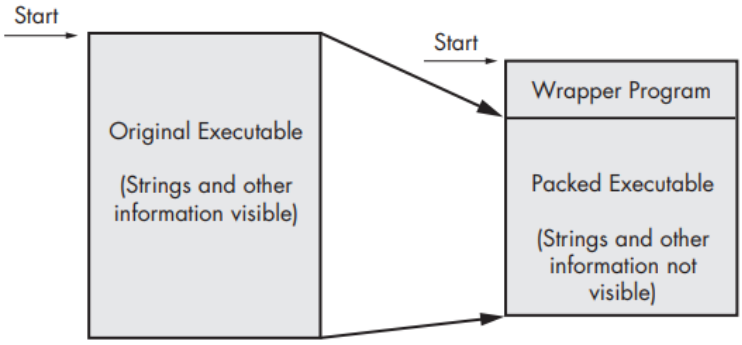
Πακεταρισμένο και Συσκοτισμένο Κακόβουλο Λογισμικό

Οι συγγραφείς κακόβουλου λογισμικού χρησιμοποιούν συνήθως τεχνικές *πακεταρίσματος* (packing) ή *συσκότισης κώδικα* (obfuscation) προκειμένου να δυσκολέψουν την ανάλυση και ανίχνευση του λογισμικού τους. Τα συσκοτισμένα προγράμματα αποσκοπούν στην συγκάλυψη της εκτέλεσης τους. Τα πακεταρισμένα προγράμματα αποτελούν υποσύνολο των συσκοτισμένων προγραμμάτων και ουσιαστικά είναι προγράμματα συμπιεσμένα ώστε να μην μπορούν να αναλυθούν. Οι δυο παραπάνω τεχνικές δυσκολεύουν σημαντικά την στατική ανάλυση ενός προγράμματος και αποτελούν μεγάλες προκλήσεις του τομέα της ανίχνευσης και ανάλυσης κακόβουλου λογισμικού.

Όταν εκτελείται ένα πακεταρισμένο πρόγραμμα, εκτελείται επίσης ένα δεύτερο μικρό πρόγραμμα που αποσκοπεί στην αποσυμπίεση και εκτέλεση του πακεταρισμένου αρχείου, όπως φαίνεται στο Σχήμα 2.1. Όταν ένα πακεταρισμένο πρόγραμμα αναλυθεί με στατικές μεθόδους μόνο το μικρό πρόγραμμα συμπίεσης-αποσυμπίεσης μπορεί να εξεταστεί περαιτέρω.

2.1.2 Τεχνικές Ανάλυσης Κακόβουλων Αρχείων

Τα εκτελέσιμα αρχεία προκειμένου να αναλυθούν για πιθανές απειλές θα πρέπει να περάσουν κάποια στάδια ανάλυσης τα οποία θα φανερώσουν τη συμπεριφορά και τις προθέσεις τους. Συνήθως, όταν πραγματοποιείται ανάλυση κακόβουλου λογισμικού οι ερευνητές έχουν στη διάθεση τους ένα εκτελέσιμο αρχείο το οποίο δεν είναι σε μορφή *αναγνώσιμη από τον άνθρωπο* (human-readable). Προκειμένου να εξάγουν νόημα από αυτό το αρχείο θα πρέπει οι αναλυτές να χρησιμοποιήσουν διάφορα εργαλεία και τεχνικές, καθένα από τα οποία θα φανερώνει μικρά κομμάτια πληροφορίας απαιτούμενα για την κατανόηση της συνολικής



Σχήμα 2.1: Το αρχείο αριστερά αναπαριστά το αυθεντικό εκτελέσιμο αρχείο μαζί με όλες τις συμβολοσειρές, τις εισαγωγές συναρτήσεων και άλλες πληροφορίες. Στα δεξιά φαίνεται το πακεταρισμένο εκτελέσιμο. Όλες οι πληροφορίες του αυθεντικού εκτελέσιμου είναι πλέον συμπιεσμένες και δεν μπορούν να διαβαστούν από εργαλεία στατικής ανάλυσης.

συμπεριφοράς του αρχείου. Στην ανάλυση κακόβουλου λογισμικού υπάρχουν δύο θεμελιώδεις προσεγγίσεις: στατική και δυναμική. Η *Στατική ανάλυση* (static analysis) αναφέρεται στην εξέταση του αρχείου χωρίς την εκτέλεση του. Η *Δυναμική Ανάλυση* (dynamic analysis) αναφέρεται στην εκτέλεση του αρχείου με σκοπό την εξαγωγή πληροφορίας για την συμπεριφορά του. Η συγκεκριμένη διπλωματική εργασία είναι βασισμένη σε στατική ανάλυση ωστόσο θα γίνει μία σύνοψη και των δύο τεχνικών για λόγους πληρότητας.

Στατική Ανάλυση

Η στατική ανάλυση είναι η διαδικασία ανάλυσης εκτελέσιμου δυαδικού κώδικα χωρίς στην πραγματικότητα να εκτελείται το αρχείο. Η στατική ανάλυση έχει το πλεονέκτημα ότι μπορεί να αποκαλύψει, πως θα συμπεριφερθεί ένα πρόγραμμα κάτω από ασυνήθιστες συνθήκες. Αυτό συμβαίνει γιατί μπορούμε να εξετάσουμε κάποια μέρη ενός προγράμματος που δεν θα είχαν εκτελεστεί στην κανονική ροή εκτέλεσης του προγράμματος. Το κακόβουλο λογισμικό μπορεί να αρχίσει την εκτέλεση του μετά από κάποιο χρονικό διάστημα ή όταν συμβεί κάποιο ειδικό γεγονός και ικανοποιηθεί μια *λογική συνθήκη* (logic bomb). Είναι σημαντικό να βρεθεί πως το κακόβουλο λογισμικό μπορεί να ξεφύγει από το εντοπισμό των αντιικών προγραμμάτων, καθώς επίσης πώς μπορεί ένα πρόγραμμα να παρακάμψει τείχη προστασίας και άλλες προστασίες ασφάλειας. Προκειμένου να επιτευχθεί στατική ανάλυση θα πρέπει κάποιος να έχει καλές γνώσεις της συμβολικής γλώσσας και του λειτουργικού συστήματος του στόχου.

Με την τεχνική *αντίστροφης μηχανικής* (reverse engineering), οι ερευνητές είναι σε θέση να μετατρέψουν την γλώσσα assembly σε γλώσσα υψηλότερου επιπέδου. Η αντίστροφη μηχανική είναι η μόνη λύση για την κατανόηση και την απόκτηση του πηγαίου κώδικα, από προγράμματα κλειστού κώδικα. Τα εργαλεία στατικής ανάλυσης περιλαμβάνουν αναλυτές προγράμματος, *αποσφαλματωτές* (debuggers) και *αποσυμβολομεταφραστές* (dissassemblers). Τα παραπάνω εργαλεία είναι σε θέση να ανιχνεύσουν αν το κακόβουλο λογισμικό χρησιμοποιεί κάποια από τις τεχνικές προστασίας του λογισμικού.

Δυναμική Ανάλυση

Η δυναμική ανάλυση είναι η διαδικασία ανάλυσης εκτελέσιμου που πραγματοποιείται κατά τη διάρκεια ή μετά την εκτέλεση του αρχείου. Συνήθως, αποτελεί το δεύτερο βήμα στην ανάλυση κακόβουλου λογισμικού και έπεται της στατικής ανάλυσης όταν ο αναλυτής έχει οδηγηθεί σε αδιέξοδο λόγω συσκότισης ή πακεταρίσματος του αρχείου. Το βασικό πλεονέκτημα της συγκεκριμένης μεθόδου είναι ότι ο αναλυτής μπορεί να παρατηρήσει την πραγματική συμπεριφορά του λογισμικού. Για παράδειγμα, μέσω δυναμικής ανάλυσης μπορούν να φανερωθούν ανοιχτές συνδέσεις που έχουν εκκινηθεί από το πρόγραμμα καθώς και αρχεία που δημιουργεί, τροποποιεί ή διαγράφει.

Η διαδικασία της δυναμικής ανάλυσης κακόβουλου λογισμικού πρέπει να πραγματοποιείται σε ένα απομονωμένο και ελεγχόμενο περιβάλλον (φυσικό ή εικονικό), μέσα στο οποίο, με την χρήση κατάλληλων εργαλείων, μπορούμε με ασφάλεια να εκτελέσουμε το κακόβουλο λογισμικό και να παρακολουθήσουμε την συμπεριφορά του.

Για τη δυναμική ανάλυση συνήθως προτιμάται η χρήση *εικονικών μηχανών* διότι διευκολύνει τη διαδικασία της ανάλυσης, καθώς έχουμε τη δυνατότητα να αποθηκεύουμε και να χρησιμοποιούμε διαφορετικές καταστάσεις της ίδιας εικονικής μηχανής με την αξιοποίηση του μηχανισμού των *στιγμιοτύπων* (snapshots). Στο εμπόριο υπάρχουν διάφορα ολοκληρωμένα προϊόντα που είναι υπεύθυνα για τη διεξαγωγή δυναμικής ανάλυσης και τα πιο δημοφιλή χρησιμοποιούν τεχνολογίες sandbox. Το *sandbox* είναι ένας μηχανισμός ασφάλειας που αποτελείται από εικονικά περιβάλλοντα πού προσομοιώνουν δικτυακές συσκευές με τρόπο τέτοιο ώστε να εξασφαλίζεται η κανονική λειτουργία του υπό εξέταση αρχείου.

Ωστόσο, η εξέλιξη στην ανάπτυξη κακόβουλου λογισμικού έχει επιφέρει την εμφάνιση λογισμικού πού είναι ικανό να αντιλαμβάνεται την εκτέλεση του σε εικονική μηχανή και να προσαρμόζει τη συμπεριφορά του ανάλογα. Η συγκεκριμένη τεχνική αποτελεί πρόκληση για τους αναλυτές οι οποίοι μπορεί να ξεγελαστούν από τον συγγραφέα του malware εάν δεν εκτελέσουν το αρχείο σε φυσικά μηχανήματα.

2.1.3 Τεχνικές Ανίχνευσης Κακόβουλων Αρχείων

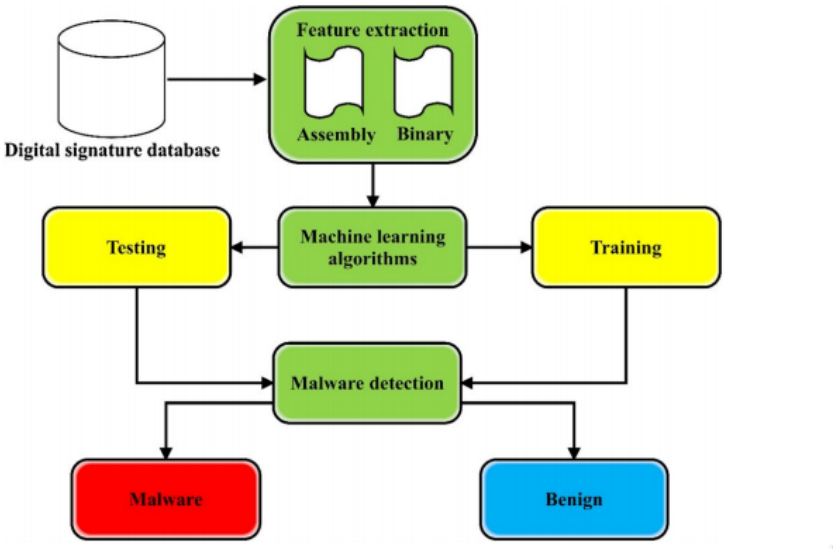
Μετά την ανάλυση ενός κακόβουλου εκτελέσιμου και την εξαγωγή χαρακτηριστικών πρέπει να εφαρμοστούν τεχνικές προκειμένου να ανιχνευθούν οι προθέσεις του αρχείου και να προστατευτεί το σύστημα από πιθανούς κινδύνους. Ουσιαστικά, αυτό το στάδιο αφορά την κατηγοριοποίηση ενός αρχείου στην οικογένεια που ανήκει. Οι διαφορετικές μέθοδοι για την παραπάνω ανίχνευση είναι μέσω υπογραφής, συμπεριφοράς και προδιαγραφών.

Ανίχνευση μέσω Υπογραφής

Η ανίχνευση *βασισμένη στην υπογραφή* (signature-based) διατηρεί συνήθως μια βάση δεδομένων από υπογραφές και ανιχνεύει ή κατηγοριοποιεί αρχεία συγκρίνοντας τα μοτίβα τους με την βάση. Οι συγκεκριμένες υπογραφές προκύπτουν ως σύνολο χαρακτηριστικών εξαγόμενων από τον κώδικα συμβολικής γλώσσας των αρχείων. Ουσιαστικά, κάθε οικογένεια κακόβουλου λογισμικού απαρτίζεται από ένα σύνολο χαρακτηριστικών που επιδεικνύει τις ιδιαιτερότητες και ιδιοτροπίες της κάθε οικογένειας. Στο Σχήμα 2.2 παρουσιάζεται η σχηματική αναπαράσταση της διαδικασίας ανίχνευσης με την συγκεκριμένη τεχνική.

Τα περισσότερα antivirus χρησιμοποιούν τη συγκεκριμένη τεχνική για ανίχνευση και κατηγοριοποίηση εκτελέσιμων αρχείων. Συγκεκριμένα, διατηρούν μια βιβλιοθήκη με γνωστές υπογραφές κώδικα και την ανανεώνουν συνεχώς. Αυτή η ιδιαιτερότητα της ανίχνευσης μέσω

υπογραφής έχει ως αποτέλεσμα να γίνεται γρήγορη και αποδοτική ανίχνευση με υψηλή ακρίβεια σε κακόβουλα λογισμικά που αντιστοιχούν σε κάποια υπογραφή της βάσης. Το βασικό μειονέκτημα είναι όμως ότι υστερεί στην ανίχνευση κακόβουλου λογισμικού που εμφανίζεται για πρώτη φορά (zero-day malware).



Σχήμα 2.2: Ανίχνευση Κακόβουλου Λογισμικού μέσω υπογραφής

Ανίχνευση μέσω Συμπεριφοράς

Η ανίχνευση *βασισμένη σε συμπεριφορά* (behavior-based) είναι επίσης γνωστή ως *βασισμένη σε ανωμαλίες* (anomaly-based) ή *ευριστική* (heuristic-based). Ο βασικός στόχος της συγκεκριμένης τεχνικής είναι να αναλύσει τη συμπεριφορά γνωστών και άγνωστων ιών. Οι παράμετροι συμπεριφοράς περιλαμβάνουν στοιχεία όπως οι διευθύνσεις αποστολέα και παραλήπτη του ιού, ο τύπος των συνημμένων αρχείων καθώς και διάφορα μετρήσιμα στατιστικά χαρακτηριστικά.

Η συγκεκριμένη τεχνική περιλαμβάνει δύο στάδια: εκπαίδευση και ανίχνευση. Κατά την εκπαίδευση παρατηρείται η συμπεριφορά του συστήματος σε κανονική λειτουργία (χωρίς malware) και χρησιμοποιούνται αλγόριθμοι μηχανικής μάθησης ώστε να μοντελοποιήσουν τη συμπεριφορά του συστήματος υπό κανονικές συνθήκες. Στο στάδιο της ανίχνευσης το προφίλ του συστήματος εκείνη τη χρονική στιγμή συγκρίνεται με το μοντελοποιημένο προφίλ πού παρατηρήθηκε υπό κανονικές συνθήκες. Σε περίπτωση που υπάρχουν διαφορές μεταξύ των δύο προφίλ τότε αυτές καταγράφονται και το αρχείο σηματοδοτείται ως υποψήφια απειλή για περαιτέρω εξέταση.

Η παραπάνω διαδικασία ακολουθεί τα εξής βήματα:

1. **Συλλογή Δεδομένων**: Συλλέγονται οι στατικές και δυναμικές πληροφορίες που εξάγονται.

2. **Ερμηνεία Δεδομένων**: Μετατρέπονται οι ακατέργαστες πληροφορίες που συλλέχθηκαν στο προηγούμενο βήμα σε *ενδιάμεσες αναπαραστάσεις* (intermediate representations).

3. **Ταίριασμα Αλγορίθμου**: Συγκρίνονται οι αναπαραστάσεις με την συμπεριφορά του συστήματος υπό κανονικές συνθήκες (χωρίς κακόβουλο λογισμικό).

Το σημαντικό πλεονέκτημα της συγκεκριμένους μεθόδου είναι ότι σε αντίθεση με την ανίχνευση μέσω υπογραφής, είναι ικανή να αναγνωρίσει zero-day malware. Αυτή η δυνατότητα προκύπτει από το γεγονός ότι οποιαδήποτε διαφοροποίηση από το προφίλ κανονικής λειτουργίας θεωρείται ύποπτη. Αντιθέτως, τα βασικά μειονεκτήματα της μεθόδου αφορούν το υψηλό ποσοστό *λανθασμένων προβλέψεων* (false alarms) και την πολυπλοκότητα ανεύρεσης των χαρακτηριστικών που πρέπει να χρησιμοποιηθούν κατά το στάδιο της εκπαίδευσης.

2.2 Μηχανική Μάθηση

Η *μάθηση* (learning) αποτελεί θεμελιώδη ιδιότητα της νοήμουσας συμπεριφοράς του ανθρώπου και παρά τις εκτενείς μελέτες επί χρόνια στον τομέα της Γνωστικής Ψυχολογίας, η έννοια της μάθησης δεν έχει γίνει πλήρως κατανοητή. Τα τελευταία χρόνια οι επιστήμονες του χώρου της *Τεχνητής Νοημοσύνης* (Artificial Intelligence) έχουν καταβάλλει σημαντικές προσπάθειες προκειμένου να δημιουργήσουν συστήματα ικανά να μαθαίνουν. Η έννοια της μάθησης στα υπολογιστικά συστήματα ονομάζεται *Μηχανική Μάθηση* (Machine Learning) και ορίζεται ως εξής:

"Το φαινόμενο κατά το οποίο ένα σύστημα βελτιώνει την απόδοση του κατά την εκτέλεση μιας συγκεκριμένης εργασίας, χωρίς να υπάρχει ανάγκη να προγραμματιστεί εκ νέου"

Βάσει του παραπάνω ορισμού, η Μηχανική Μάθηση έχει ως σκοπό την δημιουργία μηχανών ικανών να μαθαίνουν, να βελτιώνουν, δηλαδή, την απόδοσή τους σε κάποιους τομείς μέσω της αξιοποίησης προηγούμενης γνώσης και εμπειρίας. Ένας αναλυτικότερος και δημοφιλέστερος ορισμός της Μηχανικής Μάθησης δίνεται στο [Mitc97]:

"Ένα πρόγραμμα υπολογιστή λέμε ότι μαθαίνει από την εμπειρία E ως προς κάποια κλάση εργασιών T και μέτρο απόδοσης P, αν η απόδοσή του σε εργασίες από το T, όπως μετριέται από το P, βελτιώνεται μέσω της εμπειρίας E."

Ως κλάδος της τεχνητής νοημοσύνης, η μηχανική μάθηση ασχολείται κυρίως με τη μελέτη αλγορίθμων που βελτιώνουν τη συμπεριφορά τους σε κάποια εργασία που τους έχει ανατεθεί χρησιμοποιώντας την εμπειρία τους. Όσων αφορά τη σχεδίαση των συστημάτων μηχανικής μάθησης, για τα συστήματα που ανήκουν στη συμβολική τεχνητή νοημοσύνη, η δυνατότητα μάθησης προσδιορίζεται ως η ικανότητα άντλησης επιπρόσθετης γνώσης που επιφέρει μεταβολές στην υπάρχουσα καταχωρημένη γνώση είτε αλλάζοντας χαρακτηριστικά της είτε με αυξομείωση της. Στην περίπτωση των συστημάτων που ανήκουν στην μη-συμβολική τεχνητή νοημοσύνη (όπως τα νευρωνικά δίκτυα), ως μάθηση προσδιορίζεται η δυνατότητα που διαθέτουν τα συστήματα στο να μετασχηματίζουν την εσωτερική τους δομή, παρά στο να μεταβάλλουν κατάλληλα τη γνώση που έχει καταχωρηθεί μέσα σε αυτά κατά το σχεδιασμό τους. Παρόλο που απέχουμε αρκετά από την κατασκευή μηχανών που μαθαίνουν τόσο καλά όσο ο άνθρωπος, για συγκεκριμένες περιοχές μάθησης έχουν αναπτυχθεί αλγόριθμοι οι οποίοι έχουν επιτρέψει την εμφάνιση σύγχρονων εμπορικών εφαρμογών με σημαντική επιτυχία. Επιπρόσθετα, τα αποτελέσματα από διάφορες εφαρμογές της τεχνητής νοημοσύνης αρχίζουν ήδη να είναι ορατά και πολλές φορές επιφέρουν αξιοσημείωτα αποτελέσματα. Συνεπώς, όπως και σε πολλούς άλλους κλάδους έχει αρχίσει τα τελευταία χρόνια να γίνεται αισθητή η παρουσία της μηχανικής μάθησης και στον τομέα της *Ασφάλειας Υπολογιστών* (Computer Security).

2.2.1 Επιβλεπόμενη Μάθηση

Η *Επιβλεπόμενη Μάθηση* (Supervised Learning) αποτελεί μια ειδική κατηγορία μηχανικής μάθησης, στην οποία ο χαρακτηρισμός των δεδομένων βασίζεται σε κάποια δεδομένα εκπαίδευσης. Τα δεδομένα εκπαίδευσης απαρτίζονται από ένα σύνολο παραδειγμάτων τα οποία χρησιμοποιούνται για εκπαίδευση μοντέλων. Κάθε παράδειγμα του συνόλου δεδομένων αποτελείται από ένα διάνυσμα χαρακτηριστικών εισόδου και μια επιθυμητή τιμή εξόδου. Οι αλγόριθμοι και ταξινομητές αυτού του είδους μάθησης αναλύουν τα δεδομένα και παράγουν ένα μοντέλο ικανό να χαρακτηρίσει και να προβλέψει την τιμή εξόδου νέων παραδειγμάτων. Προκειμένου να καταστεί εφικτό το παραπάνω, ο αλγόριθμος πρέπει να έχει την ικανότητα να γενικεύει από τα δεδομένα εκπαίδευσης με ένα "λογικό" τρόπο χωρίς να οδηγείται σε *υπερεκπαίδευση* (overfitting). Η επιβλεπόμενη μάθηση χρησιμοποιείται σε προβλήματα *Ταξινόμησης* (Classification), *Παλινδρόμησης* (Regression) και *Διερμηνείας* (Interpretation)

Παρακάτω θα γίνει αναφορά σε κάποιες βασικές έννοιες της επιβλεπόμενης μάθησης:

Μοντέλα και Παράμετροι

Ένα *μοντέλο* (model) στην επιβλεπόμενη μάθηση αναφέρεται στην μαθηματική λογική από την οποία προκύπτει η πρόβλεψη $\hat{y}_i$ ως συνάρτηση του διανύσματος εισόδου $\mathbf{x_i}$. Ένα παράδειγμα είναι το *γραμμικό μοντέλο* (linear model), στο οποίο η πρόβλεψη προκύπτει ως $\hat{y}_i = \sum_j \theta_j x_{ij}$. Οι τιμές των προβλέψεων του μοντέλου μπορούν να λάβουν διαφορετικές ερμηνείες ανάλογα με το πρόβλημα (π.χ ταξινόμησης, παλινδρόμησης).

Οι *παράμετροι* (parameters) είναι το στοιχείο που είναι απροσδιόριστο στην αρχή και στόχος του μοντέλου είναι να επιλέξει τις καλύτερες δυνατές μέσω της διαδικασίας εκπαίδευσης από το σύνολο δεδομένων εισόδου. Το $\boldsymbol{\theta}$ αποτελεί το διάνυσμα παραμέτρων του μοντέλου.

Αντικειμενική Συνάρτηση

Η εκπαίδευση του μοντέλου εναπόκειται ουσιαστικά στον υπολογισμό του διανύσματος $\boldsymbol{\theta}$ που θα προσαρμοστεί όσο το δυνατόν καλύτερα στα δεδομένα εισόδου. Συνεπώς, πρέπει να οριστεί μια μετρική προκειμένου να αξιολογούμε την απόδοση του μοντέλου για κάθε διαφορετικό διάνυσμα $\boldsymbol{\theta}$ που επιλέγεται. Η αξιολόγηση του μοντέλου γίνεται μέσω της *αντικειμενικής συνάρτησης* (objective function).

Το βασικό χαρακτηριστικό των αντικειμενικών συναρτήσεων είναι ότι αποτελούνται από δυο όρους: τις *απώλειες λόγω εκπαίδευσης* (training loss) και τον *όρο κανονικοποίησης* (regularization term). Οπότε ορίζουμε την αντικειμενική συνάρτηση ως

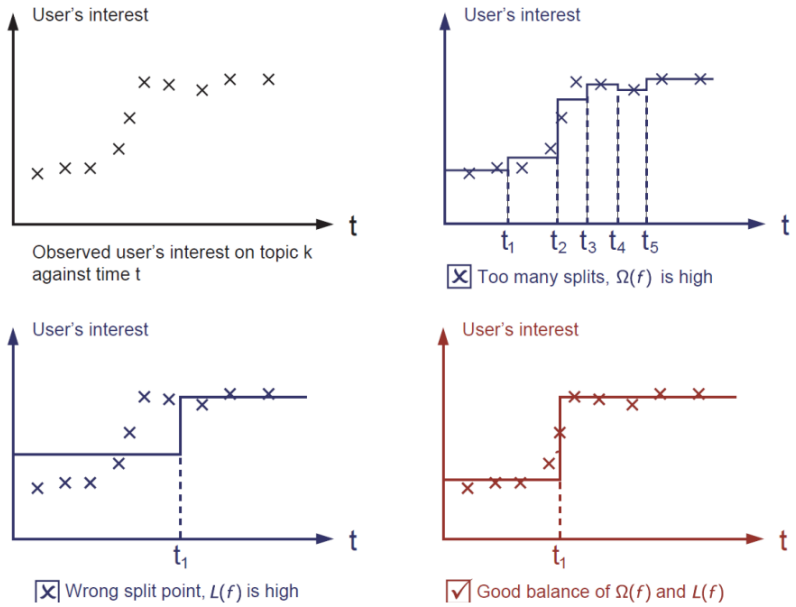
$$obj(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \Omega(\boldsymbol{\theta}) \tag{2.1}$$

όπου το L εκφράζει τη συνάρτηση απωλειών εκπαίδευσης και το Ω τον όρο κανονικοποίησης. Το σφάλμα εκπαίδευσης εκφράζει πόσο σωστές προβλέψεις παράγει το μοντέλο ως προς τα δεδομένα εκπαίδευσης. Μια συνήθης επιλογή για το L είναι το *μέσο τετραγωνικό σφάλμα* (mean squared error):

$$L(\boldsymbol{\theta}) = \sum_i (y_i - \hat{y}_i)^2 \tag{2.2}$$

Μια δεύτερη επιλογή για συνάρτηση σφάλματος εκπαίδευσης είναι η *λογιστική συνάρτηση* (logistic function), η οποία παρουσιάζεται παρακάτω

$$L(\boldsymbol{\theta}) = \sum_i y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i}) \tag{2.3}$$



Σχήμα 2.3: Παραδείγματα δυνατών προσαρμογών στα δεδομένα εισόδου

Ο όρος κανονικοποίησης ελέγχει την πολυπλοκότητα του μοντέλου και βοηθάει σημαντικά στην καταπολέμηση της υπερεκπαίδευσης. Προκειμένου να έχουμε ένα ικανοποιητικό αποτέλεσμα θα πρέπει να πετύχουμε μια καλή ισορροπία μεταξύ της ελαχιστοποίησης της συνάρτησης $\Omega(f)$ και της συνάρτησης $L(f)$ ώστε το μοντέλο μας να παράγει ακριβείς προβλέψεις αλλά να μην υπερπροσαρμόζεται στα δεδομένα εκπαίδευσης. Στο Σχήμα 2.3 παρουσιάζονται τέσσερα παραδείγματα συναρτήσεων προσαρμογής στην *βηματική συνάρτηση* (step function).

Παρατηρούμε ότι η καλύτερη επιλογή προσαρμογής είναι η κάτω δεξιά που είναι σημειωμένη με κόκκινο χρώμα. Ο γενικός κανόνας είναι ότι η προσαρμογή πρέπει να είναι απλή και ακριβής. Αναζητούμε ουσιαστικά έναν ικανοποιητικό συμβιβασμό μεταξύ της *διακύμανσης* (variance) του μοντέλου (εκφράζει την ακρίβεια των προβλέψεων) και της *μεροληψίας* (bias) (εκφράζει τη δυνατότητα γενίκευσης). Στη βιβλιογραφία ο παραπάνω συμβιβασμός αναφέρεται ως **bias-variance tradeoff**.

2.2.2 Μη-επιβλεπόμενη Μάθηση

Η *Μη-επιβλεπόμενη Μάθηση* (Unsupervised Learning) αποτελεί κατηγορία μηχανικής μάθησης, στόχος της οποίας είναι η ανακάλυψη πιθανής δομής που κρύβεται πίσω από *μη-επισημασμένα δεδομένα* (unlabeled data). Ουσιαστικά, οι αλγόριθμοι αυτού του είδους μάθησης κατασκευάζουν μοντέλα για κάποιο σύνολο εισόδων υπό μορφή παρατηρήσεων χωρίς να γνωρίζουν τις επιθυμητές εξόδους. Επειδή όμως δεν υπάρχει μηχανισμός ανταμοιβής ούτε σφάλμα λανθασμένης εξόδου είναι δύσκολο να αξιολογηθεί η απόδοση του μοντέλου. Η μη-επιβλεπόμενη μάθηση χρησιμοποιείται σε προβλήματα *Ανάλυσης Συσχετισμών* (Association Analysis) και *Ομαδοποίησης* (Clustering)

2.3 Ταξινομητές Μηχανικής Μάθησης

Οι *ταξινομητές* (classifiers) έχουν στόχο να αντιστοιχίσουν δείγματα στις κλάσεις που ανήκουν μέσω ενός συνόλου χαρακτηριστικών που περιγράφουν τα δείγματα. Συγκεκριμένα,

στην ταξινόμηση είναι γνωστό ένα σύνολο δειγμάτων που αποτελείται από *διανύσματα χαρακτηριστικών* (feature vectors) και *ετικέτες κλάσεων* (class labels). Στόχος είναι η εύρεση μιας συνάρτησης που δίνει σωστές απαντήσεις σε αυτά τα δείγματα και έχει μικρό σφάλμα στην πρόβλεψη των ετικετών κλάσης σε δεδομένα που δεν έχουν παρατηρηθεί στο παρελθόν.

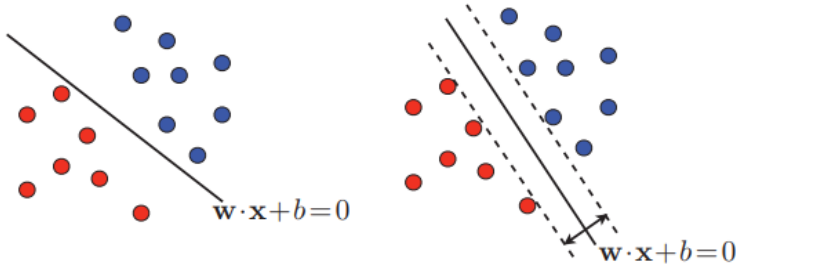
Έστω ότι έχουμε στη διάθεση μας l δείγματα εκπαίδευσης, $(y_i, \mathbf{x}_i)\ i = 1, \dots l$, που αποτελούνται από ετικέτες κλάσεων, $y_i \in \{1, -1\}$, και n-διάστατα διανύσματα χαρακτηριστικών, $\mathbf{x}_i \in \mathbf{R}^n$. Αναζητούμε συνάρτηση $f(\ \ ; \boldsymbol{\alpha}) : \mathbf{x} \mapsto y$ που αναπαριστά τον ταξινομητή $y = f(\mathbf{x}; \boldsymbol{\theta})$, όπου $\boldsymbol{\theta}$ είναι το σύνολο παραμέτρων του ταξινομητή.

2.3.1 Μηναχές Διανυσμάτων Υποστήριξης

Οι *μηχανές διανυσμάτων υποστήριξης* (support vector machines) [Bose92] επιλύουν το πρόβλημα της επιβλεπόμενης ταξινόμησης σε μεγάλο αριθμό διαστάσεων. Αρχικά, θα αναφερθούμε στον αλγόριθμο για γραμμικά διαχωρίσιμα πρότυπα και στην συνέχεια θα παρουσιάσουμε την γενικευμένη του εκδοχή που καλύπτει και τα μη-γραμμικά διαχωρίσιμα. Επιπλέον, θα γίνει αναφορά σε συναρτήσεις πυρήνα και στον τρόπο που μπορούμε να χρησιμοποιήσουμε τη συγκεκριμένη δυαδική μηχανή μάθησης για να ταξινομήσουμε δείγματα σε μεγαλύτερο πλήθος κλάσσεων.

Γραμμικά Διαχωρίσιμα Πρότυπα

Σε αυτή την περίπτωση κάνουμε την υπόθεση ότι τα δείγματα εκπαίδευσης μπορούν να χωριστούν γραμμικά και συνεπώς υπάρχει ένα *υπερεπίπεδο* (hyperplane) που χωρίζει τα δείγματα εκπαίδευσης σε δύο πληθυσμούς που ανήκουν σε διαφορετικές κλάσεις, όπως φαίνεται και στο αριστερό μέρος του Σχήματος 2.4. Δεδομένου όμως του άπειρου πλήθους από τέτοια πιθανά υπερεπίπεδα ο αλγόριθμος επιστρέφει εκείνο το οποίο μεγιστοποιεί το *περιθώριο διαχωρισμού* (margin separation), ή αλλιώς την απόσταση από τα κοντινότερα σημεία. Το συγκεκριμένο υπερεπίπεδο ονομάζεται *βέλτιστο υπερεπίπεδο* (maximum-margin hyperplane) και ένα παράδειγμα παρουσιάζεται στο δεξί μέρος του Σχήματος 2.4



Σχήμα 2.4: Δύο πιθανά διαχωριστικά υπερεπίπεδα. Δεξία παρουσιάζεται το βέλτιστο.

Προκειμένου να υπολογιστεί το βέλτιστο υπερεπίπεδο πρέπει να ορίσουμε αυστηρά το πρόβλημα βελτιστοποίησης που θέλουμε να λύσουμε. Το παραπάνω είναι ισοδύναμο με την εύρεση του μέγιστου περιθωρίου διαχωρισμού που εισήχθηκε από τους Vapnik και Chervonenkis [Vapn98].Το πρόβλημα αφορά την τοποθέτηση του υπερεπιπέδου με τέτοιο τρόπο ώστε:

- Τα δεδομένα της κλάσης -1 βρίσκονται όλα στην μια πλευρά του υπερεπιπέδου ενώ τα δεδομένα της κλάσης +1 βρίσκονται στην άλλη πλευρά του υπερεπιπέδου.

- Το υπερεπίπεδο τοποθετείται με τέτοιο τρόπο ώστε η απόσταση μεταξύ των διανυσμάτων που βρίσκονται πιο κοντά στο υπερεπίπεδο να είναι μέγιστη.

Το υπερεπίπεδο στο $\mathbb{R}^N$ ορίζεται από την παρακάτω Εξίσωση:

$$\mathbf{w}\mathbf{x} + b = 0 \tag{2.4}$$

όπου το **x** είναι σημείο του υπερεπιπέδου, το **w** είναι *n*-διάστατο διάνυσμα κάθετο στο υπερεπίπεδο, και το b είναι η απόσταση του κοντινότερου σημείου στο υπερεπίπεδο από την αρχή των αξόνων. Συνεπώς, ο ταξινομητής ορίζεται ως $f(\mathbf{x}; \mathbf{w}, b) = sgn(\mathbf{w} \cdot \mathbf{x} + b)$.

Οι παραπάνω απαιτήσεις οδηγούν σε πρόβλημα *τετραγωνικής βελτιστοποίησης* (quadratic optimization). Δοθέντος ενός δείγματος εκπαίδευσης αναζητούμε να βρεθούν οι βέλτιστες τιμές του διανύσματος βαρών **w** και της πόλωσης b ώστε να ικανοποιούν τους περιορισμούς

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{με } i = 1, \ldots, l. \tag{2.5}$$

και το διάνυσμα βαρών **w** να ελαχιστοποιεί τη συνάρτηση κόστους

$$\frac{1}{2}|\mathbf{w}|^2 \tag{2.6}$$

Το παραπάνω πρόβλημα βελτιστοποίησης με περιορισμούς λύνεται χρησιμοποιώντας τη μέθοδο των πολλαπλασιαστών Lagrange [Bert96].

Μη-γραμμικά Διαχωρίσιμα Πρότυπα

Σε αυτή την υποενότητα θα μελετήσουμε την περίπτωση που τα πρότυπα δεν δύναται να χωριστούν γραμμικά. Δοθέντος ενός τέτοιου δείγματος εκπαίδευσης, δεν είναι δυνατό να κατασκευάσουμε ένα υπερεπίπεδο διαχωρισμού χωρίς να έχουμε σφάλματα ταξινόμησης. Ο Vapnick και Cortes ελάφρυναν τον περιορισμό αυτόν εισάγοντας την έννοια του σφάλματος ταξινόμησης όπου το κόστος ελέγχεται από την παράμετρο C. Η παράμετρος C ελέγχει το συμβιβασμό μεταξύ της πολυπλοκότητας της μηχανής και του αριθμού των μη-διαχωρίσιμων σημείων. Παρά ταύτα, θέλουμε να βρούμε ένα βέλτιστο υπερεπίπεδο που θα ελαχιστοποιεί την πιθανότητα σφάλματος, υπολογισμένο επί του συνόλου του δείγματος εκπαίδευσης.

Το περιθώριο διαχωρισμού μεταξύ των κλάσεων λέγεται ότι είναι *ελαστικό* (soft) εάν υπάρχει ένα σημείο δεδομένων $(y_i, \mathbf{x}_i)$ το οποίο παραβιάζει την συνθήκη της Εξίσωσης 2.5. Η παραβίαση της παραπάνω συνθήκης δύναται να συμβεί στις παρακάτω περιπτώσεις:

- Το σημείο δεδομένων $(y_i, \mathbf{x}_i)$ βρίσκεται μέσα στην περιοχή διαχωρισμού, αλλά στην σωστή πλευρά την επιφάνειας απόφασης οπότε έχουμε σωστή ταξινόμηση.

- Το σημείο δεδομένων $(y_i, \mathbf{x}_i)$ βρίσκεται στην λάθος πλευρά της επιφάνειας απόφασης. Σε αυτή την περίπτωση έχουμε λάθος ταξινόμηση.

Προκειμένου να αντιμετωπίστουν τα μη-διαχωρίσιμα σημεία δεδομένων εισάγεται ένα σύνολο μη-αρνητικών βαθμωτών μεταβλητών $\{\xi\}_{i=1}^{l}$ στην εξίσωση ορισμού του διαχωριστικού υπερεπιπέδου (Εξίσωση 2.5):

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i = \quad \text{με } i = 1, \ldots, l. \tag{2.7}$$

Οι βαθμωτές μεταβλητές που εισήχθησαν ονομάζονται *μεταβλητές χαλάρωσης* (slack) και αποσκοπούν στην μέτρηση της απόκλισης ενός σημείου δεδομένων από την ιδανική συνθήκη διαχωρισιμότητας των προτύπων. Για $0 < \xi_i \leq 1$ το σημείο δεδομένων εμπίπτει στην

περιοχή διαχωρισμού, αλλά στην σωστή πλευρά της επιφάνειας απόφασης. Για $\xi_i > 1$, το σημείο εμπίπτει στη λάθος πλευρά του διαχωριστικού υπερεπιπέδου. Τα σημεία δεδομένων που ικανοποιούν την Εξίσωση 2.7 ακριβώς (ακόμα και αν $\xi_i > 0$) ονομάζονται *διανύσματα υποστήριξης* (support vectors).

Στόχος αποτελεί η εύρεση ενός υπερεπιπέδου διαχωρισμού όπου θα ελαχιστοποιείται το σφάλμα ταξινόμησης, ως μέσος όρος έπι του συνόλου του δείγματος εκπαίδευσης. Όμως, επειδή η παραπάνω ελαχιστοποίηση αποτελεί πρόβλημα μη-κυρτής βελτιστοποίσης που είναι *NP-πλήρες* (NP-complete) προσσεγίζουμε τη συνάρτησιακή εξίσωση σφάλματος ως εξής:

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{l}\xi_i \tag{2.8}$$

Με βάση τα παραπάνω, η διατύπωση του προβλήματος ταξινόμησης των μη-διαχωρίσιμων προτύπων αναφέρει ότι δοθέντος ενός δείγματος εκπαίδευσης αναζητούμε τις βέλτιστες τιμές του διανύσματος βαρών **w** και της πόλωσης b ώστε να ικανοποιούν τον περιορισμό της Εξίσωσης 2.7 και επιπλέον το διάνυσμα βαρών **w** και οι μεταβλητές χαλάρωσης ξ_i να ελαχιστοποιούν τη συνάρτηση κόστους (Εξίσωση 2.8). Τέλος, όπως και στη περίπτωση των γραμμικά διαχωρίσιμων προτύπων επιλύουμε το παραπάνω πρόβλημα βελτιστοποίησης με τη μέθοδο των πολλαπλασιαστών Lagrange.

Μέθοδοι Πυρήνων

Οι *συναρτήσεις πυρήνα* (kernel functions) αποτελούν απεικονίσεις των διανυσμάτων εισόδου $\mathbf{x}_i \in \mathbf{R}^n$ σε ένα νέο ευκλείδειο χώρο μεγαλύτερης διαστατικότητας H μέσω μιας μη-γραμμικής συνάρτησης $\mathbf{\Phi} : \mathbf{R}^n \mapsto H$. Μέσω της παραπάνω απεικόνισης μπορούμε να θέσουμε το ίδιο πρόβλημα ελαχιστοποίησης του περιθωρίου διαχωρισμού αντικαθιστώντας όλα τα γινόμενα $x_i \cdot x_j$ με τα αντίστοιχα γινόμενα $\mathbf{\Phi}^T(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$.

Μια συνάρτηση πυρήνα K ορίζεται ως $K(x_i, x_j) = \mathbf{\Phi}^T(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$, και όπως αναφέρθηκε και προηγουμένως οπουδήποτε προκύπτει το γινόμενο $x_i \cdot x_j$ αντικαθίσταται με το $K(x_i, x_j)$. Όμως, δεν είναι όλοι οι πυρήνες αποδεκτοί καθώς πρέπει να πληρούν συγκεκριμένες προϋποθέσεις όπως το θεώρημα Mercer [Burg98].

Ουσιαστικά με τις συναρτήσεις πυρήνα απεικονίζουμε τα δεδομένα σε ένα μεγαλύτερο χώρο διαστατικότητας όπου τα δείγματα εκπαίδευσης θα είναι πιο διάσπαρτα (άρα και περισσότερο διαχωρίσιμα) σε σχέση με τον αρχικό χώρο διαστάσεων και συνεπώς θα υπάρχει μεγαλύτερο περιθωρίου διαχωρισμού για το βέλτιστο υπερεπίπεδο.

Παρακάτω παρουσιάζονται συνοπτικά οι συναρτήσεις πυρήνα οι οποίες χρησιμοποιήθηκαν στην εργασία:

- **Γραμμικός** (Linear): $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j^T$

  Ο συγκεκριμένος πυρήνας αποτελεί καλή επιλογή μόνο για γραμμικά διαχωρίσιμα πρότυπα.

- **Πολυωνυμικός** (Polynomial): $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j^T + r)^d$

  Στον πυρήνα αυτό όσο αυξάνεται ο βαθμός του πολυωνύμου (d) τόσο πιο ευέλικτη γίνεται η συνάρτηση απόφασης.

- **Σιγμοειδής** (Sigmoid): $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \cdot \mathbf{x}_i\mathbf{x}_j^T + r)$

  Παρόλο που ο πυρήνας αυτός συναντάται κυρίως στα *νευρωνικά δίκτυα* (neural networks) επιφέρει συχνά αξιόλογα αποτελέσματα και στις Μηχανές Διανυσμάτων Υποστήριξης.

Ωστόσο, οι συνθήκες Mercer ικανοποιούνται μόνο για συγκεκριμένο εύρος τιμών των παραμέτρων α, r

- **Ακτινικής Βάσης** (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \cdot \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \gamma > 0$

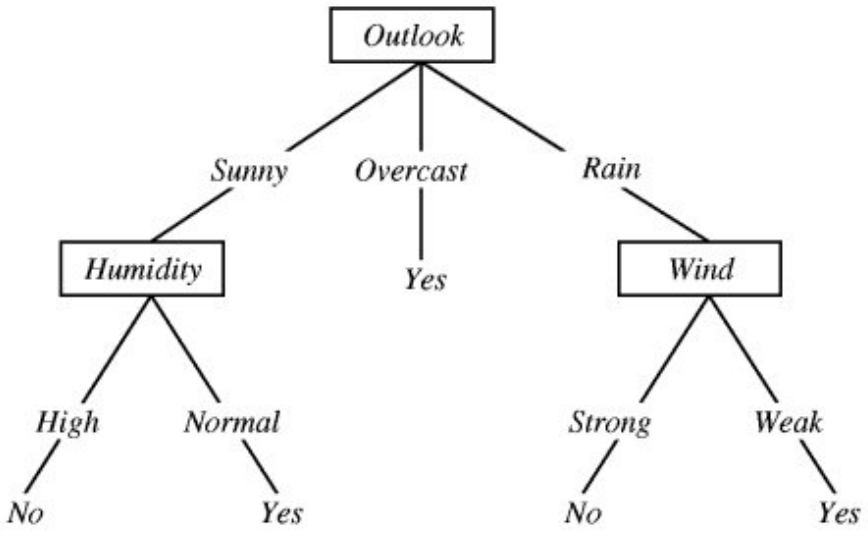
  Η θετική παράμετρος γ ελέγχει την ακτίνα της Γκαουσιανής συνάρτησης. Η επιλογή της κατάλληλης τιμής γίνεται μετά από δοκιμές για την εκάστοτε φύση του προβλήματος.

2.3.2 Δέντρα Αποφάσεων

Τα *Δένδρα Απόφασης- ΔΑ* (Decision Trees) αποτελούν έναν από τους πιο γνωστούς αλγόριθμους επιβλεπόμενης μάθησης και έχουν εφαρμοστεί επιτυχώς σε αρκετούς τομείς που απαιτείται ταξινόμηση. Η διαισθητικά κατανοητή ερμηνεία των ΔΑ τα περιγράφει ως ένα *σύνολο κανόνων ελέγχου* (if-then rules).

Ο αλγόριθμος ΔΑ δημιουργεί μια δενδροειδή μορφή όπου τα φύλλα είναι οι κλάσεις ταξινόμησης και σε κάθε κόμβο σκοπός είναι η επίτευξη περαιτέρω διαχωρισμού των μη κατηγοριοποιημένων δειγμάτων του συνόλου εκπαίδευσης. Ουσιαστικά, κάθε κόμβος του δέντρου αναπαριστά μια συνθήκη ελέγχου για κάποιο χαρακτηριστικό των δειγμάτων και κάθε κλάδος που εξέρχεται από τον κόμβο αντιστοιχεί σε μία από τις δυνατές τιμές του χαρακτηριστικού αυτού (ή σε ένα εύρος αυτών). Μετά τον σχηματισμό του δέντρου η ταξινόμηση ενός δείγματος προκύπτει μέσω απλής διάσχισης του δέντρου μέχρι να καταλήξει το συγκεκριμένο δείγμα σε κάποιο φύλλο. Η τιμή του φύλλου στην οποία θα καταλήξει είναι η κλάση που θα προβλέψει ο ταξινομητής για το δείγμα αυτό.

Προκειμένου να γίνει κατανοητή η φιλοσοφία του αλγορίθμου, στο Σχήμα 2.5 παρουσιάζουμε ένα παράδειγμα ΔΑ που είχε παρουσιαστεί στο [Quin96].



Σχήμα 2.5: Ένα παράδειγμα ΔΑ για το πρόβλημα της πρόβλεψης εάν θα είναι οι συνθήκες κατάλληλες για διεξαγωγή αγώνα τένις

Έστω ότι επιθυμούμε να ταξινομήσουμε το παρακάτω δείγμα:

$$(Outlook = Sunny, \quad Temperature = Hot, \quad Humidity = High, \quad Wind = Strong)$$

Συγκεκριμένα για το παραπάνω δείγμα η διάσχιση του δέντρου θα οδηγήσει στο φύλλο που βρίσκεται τέρμα αριστερά και συνεπώς ο ταξινομητής θα αποφανθεί ότι οι καιρικές συνθήκες δεν θα είναι κατάλληλες για αγώνα τένις.

Γενικά, τα ΔΑ αναπαριστούν μια διάζευξη συζεύξεων περιορισμών που αφορούν τα χαρακτηριστικά των δειγμάτων. Κάθε μονοπάτι του δέντρου που ξεκινάει από την ρίζα και καταλήγει σε φύλλο αντιστοιχεί σε μία σύζευξη *ελέγχων χαρακτηριστικών* (attribute tests). Για παράδειγμα το ΔΑ που παρουσιάζεται στο Σχήμα 2.5 αντιστοιχεί στην παρακάτω έκφραση:

$$(Outlook = Sunny \wedge Humidity = Normal) \bigvee$$
$$(Outlook = Overcast) \bigvee$$
$$(Outlook = Rain \wedge Wind = Weak)$$

Κατασκευή Δέντρου Απόφασης - Εκπαίδευση

Οι περισσότεροι αλγόριθμοι που έχουν δημιουργηθεί για την εκπαίδευση των ΔΑ αποτελούν παραλλαγές του βασικού αλγορίθμου που εφαρμόζει μια *άπληστη αναζήτηση* (greedy search) *από πάνω προς τα κάτω* (top-down) στο χώρο των δυνατών δέντρων απόφασης. Αυτή η προσέγγιση εφαρμόζεται από τον αλγόριθμο ID3 [Quin86] και C4.5 [Quin93].

Αρχικός στόχος της κατασκευής του ΔΑ είναι η εύρεση του χαρακτηριστικού που θα ελεγχθεί στη ρίζα του δέντρου. Όμως, προκειμένου να βρεθεί αυτό το χαρακτηριστικό πρέπει όλα τα χαρακτηριστικά να αξιολογηθούν μέσω ενός στατιστικού ελέγχου που θα αναδεικνύει πόσο ικανό είναι το καθένα να ταξινομήσει μόνο του τα δείγματα εκπαίδευσης. Το χαρακτηριστικό που θα αποδειχθεί ικανότερο είναι και αυτό που θα τοποθετηθεί στη ρίζα του δέντρου. Στη συνέχεια για κάθε δυνατή τιμή (ή εύρος τιμών) του χαρακτηριστικού της ρίζας θα δημιουργηθεί ένα αντίστοιχο κλαδί στο οποίο θα αντιστοιχήσουμε ταυτόχρονα όλα τα δείγματα εκπαίδευσης για τα οποία ισχύει η συγκεκριμένη συνθήκη ελέγχου. Έπειτα, το ικανότερο χαρακτηριστικό κάθε κόμβου θα επιλέγεται με την ίδια διαδικασία αλλά τα δείγματα εκπαίδευσης που θα λαμβάνονται υπόψη θα είναι μόνο αυτά που ανήκουν στο συγκεκριμένο κλάδο. Η παραπάνω διαδικασία είναι μια άπληστη αναζήτηση διότι ποτέ δεν γυρνάει πίσω να αναθεωρήσει κάποια απόφαση που έχει ήδη παρθεί.

Η κεντρική επιλογή του αλγορίθμου ID3 αφορά την επιλογή του χαρακτηριστικού ελέγχου σε κάθε κόμβο του δέντρου. Στόχος είναι η εύρεση του πιο χρήσιμου χαρακτηριστικού για την ταξινόμηση των δειγμάτων οπότε είναι αναγκαίο να οριστούν ποσοτικές μετρικές για την αξία του κάθε χαρακτηριστικού. Η στατιστική ιδιότητα που χρησιμοποιείται ως μετρική ονομάζεται *κέρδος πληροφορίας* (information gain) και μετράει την ικανότητα ενός χαρακτηριστικού να ταξινομήσει τα δείγματα εκπαίδευσης στις σωστές κλάσεις.

Κέρδος πληροφορίας

Προκειμένου να ορίσουμε επακριβώς το κέρδος της πληροφορίας θα ξεκινήσουμε ορίζοντας μια μετρική που χρησιμοποιείται συχνά στην επιστήμη της *θεωρίας πληροφορίας* (information theory), ονομάζεται *εντροπία* (entropy) και χαρακτηρίζει την ανομοιογένεια που υπάρχει σε μία αυθαίρετη συλλογή δειγμάτων. Δεδομένης μιας συλλογής S με δείγματα της μορφής $(y_i, \mathbf{x}_i)$ $i = 1, \ldots l$, και ετικέτες κλάσεων $y_i \in \{1, -1\}$, ορίζουμε ως p_+ και p_- τις πιθανότητες να ανήκει το δείγμα στην κλάση 1 και -1 αντίστοιχα. Η εντροπία για το παραπάνω σύνολο δεδομένων ορίζεται ως:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{2.9}$$

Η εντροπία στη θεωρία της πληροφορίας ερμηνεύεται επίσης ως ο ελάχιστος αριθμός από bits πληροφορίας που χρειάζονται για να κωδικοποιηθεί ένα τυχαίο δείγμα της συλλογής S. Η παραπάνω εξίσωση γενικεύεται από την δυαδική ταξινόμηση σε πολλαπλή ταξινόμηση πλήθους κλάσεων c ως εξής:

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{2.10}$$

Χρησιμοποιώντας την εντροπία ορίζεται η μετρική του κέρδους πληροφορίας που αναπαριστά τη μείωση της εντροπίας του συνόλου εκπαίδευσης S εάν επιλεγεί ως παράμετρος διαχωρισμού ένα συγκεκριμένο χαρακτηριστικό. Διαισθητικά, όσο μικρότερη η πληροφοριακή εντροπία (άρα και η ανομοιογένεια κλάσεων) τόσο μεγαλύτερες οι διαχωριστικές ικανότητες του χαρακτηριστικού. Συνεπώς, το κέρδος πληροφορίας ορίζεται ως:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2.11}$$

Το $Values(A)$ αναπαριστά το σύνολο όλων των δυνατών τιμών του χαρακτηριστικού A, και το S_v είναι το υποσύνολο του S για το οποίο το χαρακτηριστικό A έχει τιμή v (δηλαδή $S_v = \{s \in S | A(s) = v\}$). Το κέρδος πληροφορίας είναι το μέτρο που χρησιμοποιεί ο αλγόριθμος ID3 για να επιλέγει το καλύτερο χαρακτηριστικό διαχωρισμού σε κάθε βήμα.

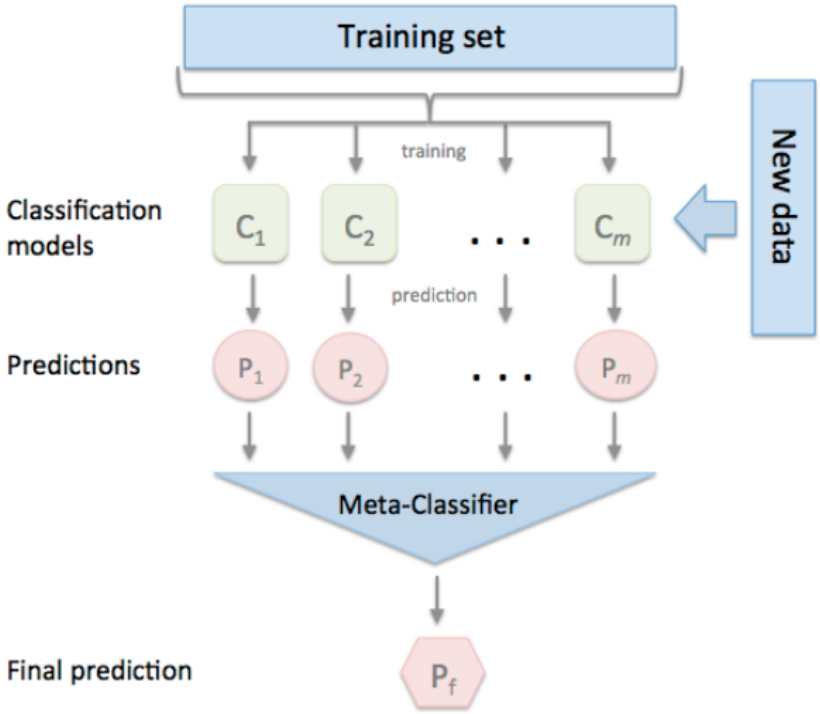
2.3.3 Τυχαία Δάση

Στην επιβλεπόμενη μάθηση οι επιδόσεις των ταξινομητών εξαρτώνται σημαντικά από την κατανομή των τιμών των δειγμάτων και σε κάποιες περιπτώσεις η εκπαίδευση μόνο ενός μοντέλου δεν είναι αρκετή για την επίτευξη ικανοποιητικών αποτελεσμάτων. Το παραπάνω πρόβλημα επιλύεται μέσω της χρήσης *συνδυαστικών ταξινομητών* (ensembled classifiers) που παρουσιάζονται σχηματικά στο Σχήμα 2.6.

Οι συνδυαστικοί ταξινομητές αποτελούν ένα σύνολο μεμονωμένων ταξινομητών που εκπαιδεύονται συνεργατικά και ανεξάρτητα μέσω ενός συνόλου δειγμάτων εισόδου. Κάθε ταξινομητής προβλέπει την κλάση ενός δείγματος και στη συνέχεια λαμβάνονται υπόψιν όλες τις προβλέψεις αποφασίζεται ποία θα είναι η πρόβλεψη του συνδυαστικού ταξινομητή.

Η λογική αυτών των μοντέλων στηρίζεται στο γεγονός ότι πολλοί ταξινομητές μπορούν να ελαχιστοποιήσουν σφάλματα που θα υπήρχαν εάν χρησιμοποιούνταν ο κάθε ταξινομητής μεμονωμένα. Ωστόσο, προκειμένου τα αποτελέσματα να είναι ικανοποιητικά θα πρέπει οι ταξινομητές που θα επιλεχθούν να είναι ακριβείς και ασυσχέτιστοι μεταξύ τους. Για να ελεγχθεί λοιπόν η συσχέτιση των δεδομένων υπάρχουν διάφορες τεχνικές και μέθοδοι στη βιβλιογραφία όπως ο συντελεστής συσχέτισης, το μέτρο διαφοράς, η εντροπία και η απόκλιση Kohavi-Wolpert [Rahm14].

Τα *τυχαία δάση* (random forests) είναι ένας συνδυαστικός ταξινομητής στον οποίο οι *ταξινομητές βάσης* (base classifiers) είναι δέντρα απόφασης. Το κάθε δέντρο απόφασης καταλήγει σε μία πρόβλεψη και έπειτα μέσω ψηφοφορίας αποφασίζεται η τελική πρόβλεψη του μοντέλου. Προκειμένου να κατασκευαστούν τα δέντρα απόφασης πού απαρτίζουν το συνδυαστικό ταξινομητή θα πρέπει να παραχθούν δείγματα εκπαίδευσης (συνήθως τυχαία). Μια δημοφιλής τεχνική παραγωγής δειγμάτων ονομάζεται bagging (bootstrap aggregating) κατά την οποία για την κατασκευή του δέντρου επιλέγονται τυχαία δείγματα (χωρίς αντικατάσταση) από τα δείγματα εκπαίδευσης του συνόλου δεδομένων [Brei96].



Σχήμα 2.6: Μοντέλο Συνδυαστικού Ταξινομητή

Το βασικό στοιχείο των τυχαίων δασών είναι ότι για το k-στό δέντρο παράγεται ένα διάνυσμα Θ_k ανεξάρτητο από όλα τα προηγούμενα τυχαία διανύσματα $\Theta_1, ..., \Theta_{k-1}$ αλλά με την ίδια κατανομή. Στη συνέχεια το k-στό δέντρο απόφασης εκπαιδεύεται μέσω των δειγμάτων εκπαίδευσης και του διανύσματος Θ_k. Το διάνυσμα Θ_k μπορεί να παραχθεί μέσω διαφόρων τεχνικών πού εμφανίζονται στη βιβλιογραφία (όπως bagging, random split selection) [Brei99, Diet98]. Μετά την εκπαίδευση και κατασκευή όλων των δέντρων απόφασης ακολουθεί η διαδικασία της ψηφοφορίας πού εκλέγει την τελική πρόβλεψη του ταξινομητή.

Ορισμός Τυχαίων Δασών

Ο αυστηρός μαθηματικός ορισμός των τυχαίων δασών έχει ως εξής:

Ένα τυχαίο δάσος είναι ένας ταξινομητής πού αποτελείται από μία συλλογή δενδρικών ταξινομητών $f(x, \Theta_k) \quad k = 1, ...$ *όπου τα* Θ_k *είναι διανύσματα ανεξάρτητα, τυχαία, και ακολουθούν την ίδια κατανομή. Το κάθε δέντρο δικαιούται μία ψήφο στην ψηφοφορία της τελικής απόφασης.*

Bagging (Bootstrap Aggregating)

Μια από τις πιο δημοφιλής τεχνικές συνδυαστικών ταξινομητών ονομάζεται bagging και αφορά την εκπαίδευση πολλών ταξινομητών με δείγματα που προκύπτουν από το αρχικό σύνολο δεδομένων μέσω της μεθόδου bootstrap. Η μέθοδος bootstrap βασίζεται στη λογική ότι τα δείγματα του συνόλου εκπαίδευσης μπορούν να επαναχρησιμοποιηθούν για περαιτέρω εκπαίδευση. Δεδομένου ενός συνόλου l δειγμάτων, η πιθανότητα επιλογής ενός δείγματος ισούται με $1 - (1 - \frac{1}{l})^l$. Το μοντέλο εκπαιδεύεται από τα boostrap δείγματα και αξιολογείται από τα δείγματα που δεν επιλέχθηκαν (out-of-bag data) ώστε να εκτιμηθεί η ακρίβεια

του μοντέλου. Έχει αποδειχθεί για την μέθοδο bagging ότι τα δέντρα απόφασης είναι ιδιαίτερα καλή επιλογή ως ταξινομητές βάσης διότι είναι ικανά να αναπαραστήσουν σύνθετες συσχετίσεις μεταξύ των χαρακτηριστικών των δεδομένων. Επίσης, μειώνουν σημαντικά τη *διακύμανση* (variance) πού αφορά τη μεταβλητότητα της πρόβλεψης του ταξινομητή για ένα συγκεκριμένο δείγμα ενώ στην περίπτωση πολλαπλών ταξινομητών εκφράζει το πόσο διαφέρουν μεταξύ τους οι προβλέψεις του κάθε ταξινομητή.

2.3.4 Διαβαθμιζόμενα Ενισχυμένα Δέντρα

Τα διαβαθμιζόμενα ενισχυμένα δέντρα, όπως προδίδει και το όνομα τους, αποτελούν έναν ταξινομητή επιβλεπόμενης μάθησης που χρησιμοποιεί συνδυασμούς δέντρων αποφάσεων όπως και τα τυχαία δάση. Ο συνδυασμός δέντρων αποφάσεων που χρησιμοποιείται αποτελείται από δέντρα ταξινόμησης και δέντρα παρεμβολής (CART). Οι τιμές που προβλέπει το κάθε δέντρο ξεχωριστά αθροίζονται και παράγουν την τελική πρόβλεψη. Το μοντέλο αυτό εκφράζεται μαθηματικά ως εξής:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{2.12}$$

όπου το K είναι ο αριθμός των δέντρων, f είναι μια συνάρτηση από των χώρο συναρτήσεων F, και F είναι το σύνολο όλων των δυνατών CART. Η αντικειμενική συνάρτηση προς ελαχιστοποίηση είναι η παρακάτω:

$$obj(\boldsymbol{\theta}) = \sum_{i}^{n} l(y_i, \hat{y_i}) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.13}$$

Με βάση τα παραπάνω παρατηρούμε ότι τα τυχαία δάση και τα ενισχυμένα δέντρα είναι ουσιαστικά δυο μοντέλα τα οποία χρησιμοποιούν συνδυασμό δέντρων για να καταλήξουν σε κάποια πρόβλεψη. Η διαφορά ανάμεσα στα δυο μοντέλα εναπόκειται αποκλειστικά στον τρόπο εκπαίδευσης.

Προσθετική Εκπαίδευση (Additive Training)

Προκειμένου να βελτιστοποιήσουμε το μοντέλο θα πρέπει να οριστούν οι παράμετροι του μοντέλου που αναπαρίστανται από το διάνυσμα $\boldsymbol{\theta}$. Αυτό που ουσιαστικά πρέπει να προσδιοριστεί είναι οι συναρτήσεις f_i, καθεμία από τις οποίες αντιπροσωπεύει τη δομή του i-στού δέντρου. Η εκμάθηση της δομής ενός δέντρου είναι αρκετά έντονο πρόβλημα υπολογιστικά και είναι αδύνατον να μάθουμε τη δομή όλων των δέντρων εξαρχής. Ο αλγόριθμος των ενισχυμένων δέντρων ακολουθεί μια προσθετική προσέγγιση στην οποία προσθέτουμε ένα δέντρο κάθε φορά. Η πρόβλεψη τιμής στο t συμβολίζεται ως $\hat{y_i}^{(t)}$. Οι προβλέψεις προκύπτουν με τον παρακάτω τρόπο:

$\hat{y_i}^{(0)} = 0$
$\hat{y_i}^{(1)} = f_1(x_i) = \hat{y_i}^{(0)} + f_1(x_i)$
$\hat{y_i}^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y_i}^{(1)} + f_2(x_i)$
...
$\hat{y_i}^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y_i}^{(t-1)} + f_t(x_i)$

Παρατηρούμε παραπάνω ότι σε κάθε βήμα προστίθεται η συνεισφορά ενός νέου δέντρου στην πρόβλεψη του ταξινομητή. Στην συγκεκριμένη μορφή εκπαίδευσης το δέντρο που προσθέτουμε κάθε φορά είναι εκείνο το οποίο ελαχιστοποιεί την αντικειμενική συνάρτηση:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$$

Ιδανικά θέλουμε να καταλήξουμε σε μία εξίσωση με έναν όρο 1ης τάξης και έναν όρο 2ης τάξης ώστε να μπορούμε να κάνουμε εύκολα βελτιστοποίηση. Προκειμένου να το πετύχουμε αυτό αναλύουμε την συνάρτηση κόστους σε σειρά Taylor μέχρι τη δεύτερη τάξη:

$$\sum_{i=1}^{n} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + +\Omega(f_t) + constant \tag{2.14}$$

όπου τα g_i και h_i ορίζονται ως

$$g_i = \partial_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)}) \tag{2.15}$$

$$h_i = \partial^2_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)}) \tag{2.16}$$

Το τελικό πρόβλημα βελτιστοποίησης προκύπτει εάν από την Εξίσωση (2.14) αφαιρέσουμε όλες τις σταθερές οπότε στο βήμα t στόχος είναι να βελτιστοποιήσουμε την παρακάτω έκφραση:

$$\sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{2.17}$$

Η παραπάνω μορφή στην οποία καταλήξαμε προσφέρει ιδιαίτερη ευελιξία διότι η αντικειμενική συνάρτηση εξαρτάται μόνο από τους όρους g_i και h_i. Συνεπώς, μπορούμε να ελαχιστοποιήσουμε οποιαδήποτε συνάρτηση κόστους χρησιμοποιώντας την ίδια συνάρτηση που θα δέχεται ως ορίσματα τα g_i και h_i.

Έπειτα λύνουμε το παραπάνω πρόβλημα και αφού βρούμε τη συνάρτηση $f_t(x)$ που ελαχιστοποιεί το κόστος τότε αντικαθιστούμε στην Εξίσωση 2.14 και λαμβάνουμε το κόστος της δομής του δέντρου f_t και ταυτόχρονα ένα μέτρο αξιολόγησης της επιλογής μας.

Εκμάθηση Δεντρικής Δομής

Παραπάνω ορίστηκε το μέτρο αξιολόγησης της δομής ενός δέντρου οπότε μένει να οριστεί ο τρόπος κατασκευής του δέντρου. Θεωρητικά θα μπορούσαμε να παράγουμε όλα τα δυνατά δέντρα και να τα αξιολογούμε προκειμένου να βρούμε το καλύτερο. Επειδή όμως αυτή η μέθοδος είναι πρακτικά αδύνατη υπολογιστικά, θα πρέπει να βελτιστοποιήσουμε τη δομή του δέντρου αυξητικά (ένα επίπεδο του δέντρου ανά φορά).

Το κάθε δείγμα αρχικά θεωρείται ως φύλλο. Στόχος είναι το κάθε φύλλο να χωρίζεται σε 2 φύλλα και η τιμή της μετρικής αξιολόγησης του φύλλου αυτού θα προκύπτει ως το άθροισμα της τιμής του καινούριου αριστερού φύλλου, της τιμής του καινούριου δεξιού φύλλου, της τιμής του τρέχοντος φύλλου και του όρου κανονικοποίησης στα καινούρια φύλλα. Εάν η

προκύπτουσα τιμή είναι μικρότερη από έναν όρο που καθορίζεται από τη συνάρτηση κανονικοποίησης τότε είναι καλύτερο να μην προστεθεί το συγκεκριμένο κλαδί στο δέντρο. Αυτή η τεχνική είναι αντίστοιχη της τεχνικής *κλαδέματος* (pruning) που χρησιμοποιείται στα μοντέλα που σχετίζονται με δέντρα απόφασης.

Τέλος, προκειμένου να βρούμε ποια φύλλα πρέπει να διαχωρίζουμε σε δύο θα πρέπει να αναζητούμε τον βέλτιστο διαχωρισμό. Ο βέλτιστος διαχωρισμός προκύπτει ταξινομώντας όλα τα δείγματα (φύλλα) στη σειρά και έπειτα μπορούμε αποδοτικά να υπολογίσουμε την τιμή αξιολόγησης όλων των δυνατών διαχωρισμών και να επιλέξουμε την μεγαλύτερη δυνατή.

Κεφάλαιο 3

Ανάλυση Υλοποίησης

Στο συγκεκριμένο Κεφάλαιο στόχος είναι να παρουσιαστεί ολόκληρη η πορεία σκέψης και υλοποίησης του τελικού συστήματος. Στην αρχή, θα αναλυθούν τα δεδομένα που είχαμε διαθέσιμα τόσο ως προς τη δομή τους όσο και ως προς την πληροφορία που μπορούμε να εξάγουμε από αυτά. Στη συνέχεια, θα βασιστούμε σε ένα ισχυρό θεωρητικό υπόβαθρο τεχνικών αντίστροφης μηχανικής ώστε να εξάγουμε σύνολα χαρακτηριστικών σημασιολογικά σωστά που διαθέτουν υψηλή ικανότητα διαχωρισιμότητας. Στο τέλος του Κεφαλαίου θα αναφερθούμε σε μεθόδους επιλογής συνόλων χαρακτηριστικών ώστε να αξιολογήσουμε τη δύναμη των προτεινόμενων χαρακτηριστικών και να αποφύγουμε εκείνα τα οποία είναι γενικά ως προς όλα τα εκτελέσιμα αρχεία.

3.1 Ανάλυση Δειγμάτων Εκπαίδευσης

Στην παρούσα ενότητα θα προβούμε σε ανάλυση των δειγμάτων εκπαίδευσης που είναι διαθέσιμα μέσω του συνόλου δεδομένων της Microsoft [Rone18]. Αρχικά, θα κάνουμε μια σύντομη αναφορά στο εργαλείο IDA Pro[1] που χρησιμοποιήθηκε για την παραγωγή των δειγμάτων εκπαίδευσης. Στη συνέχεια θα αναφερθούμε στη δομή των αρχείων που έχουμε στη διάθεση μας για να κατανοήσουμε το μέγεθος πληροφορίας που μπορούμε να εξάγουμε από αυτά.

3.1.1 IDA Pro

Ο IDA Pro (Interactive DisAssembler Professional) αποτελεί έναν από τους δημοφιλέστερους αποσυμβολομετραστές και χρησιμοποιείται από την πλειοψηφία αναλυτών κακόβουλου λογισμικού και γενικά αναλυτών *ευπαθειών* (vulnerabilities). Υποστηρίζει πολλά αρχέτυπα όπως Portable Executable (PE), Common Object File Format (COFF) και Executable and Linking Format (ELF).

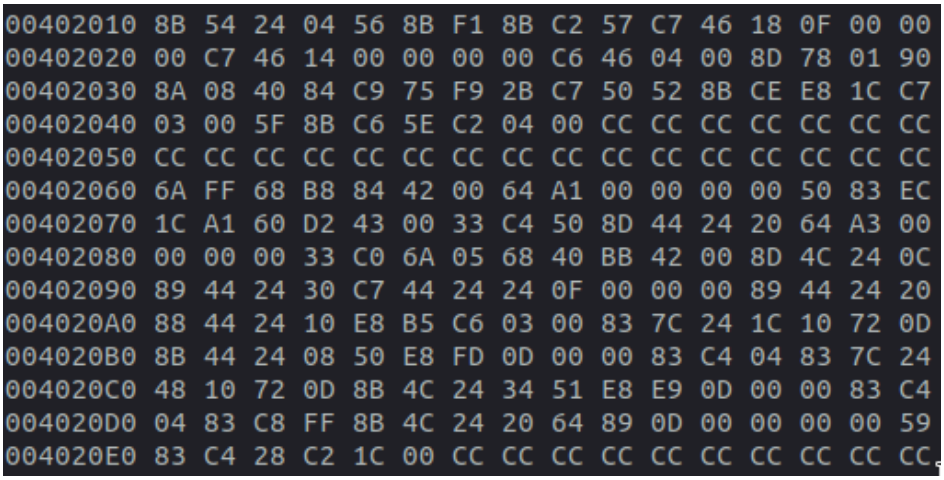
Στο σύνολο δεδομένων που χρησιμοποιήσαμε τα αρχεία από τα οποία θα εξάγουμε τα χαρακτηριστικά έχουν παραχθεί από το λογισμικό IDA Pro και περιέχουν πλήθος μεταδεδομένων που φανερώνουν κλήσεις συναρτήσεων, φόρτωση δυναμικών βιβλιοθηκών και παρουσία συμβολοσειρών. Συγκεκριμένα για κάθε δείγμα προς ταξινόμηση έχουμε στη διάθεση μας ένα δεκαεξαδικό αρχείο των bytes και ένα αρχείο assembly του προγράμματος.

3.1.2 Δομή δεκαεξαδικού αρχείου

Το αρχείο bytes είναι η ακατέργαστη δεκαεξαδική αναπαράσταση του περιεχομένου του κακόβολου εκτελέσιμου αρχείου. Μια απεικόνιση της δομής αυτού του αρχείου παρουσιά-

[1] https://www.hex-rays.com/products/ida/

ζεται στο Σχήμα 3.1



Σχήμα 3.1: Δομή δεκαεξαδικού αρχείου

Τα δεκαεξαδικά αρχεία που παράγονται από το IDA Pro είναι μια κλασσική δεκαεξαδική αναπαράσταση του περιεχομένου του προγράμματος. Κάθε γραμμή περιέχει δεκαεξαδικούς χαρακτήρες, οι οποίοι κωδικοποιούν πολλούς δυαδικούς αριθμούς. Οι δυαδικοί αριθμοί μπορούν να αναπαριστούν είτε δεδομένα είτε διευθύνσεις μνήμης.

Συγκεκριμένα, κάθε γραμμή αποτελείται από 20 bytes, όπου τα 4 πρώτα bytes αναπαριστούν την διεύθυνση στην οποία θα φορτωθεί το περιεχόμενο που αναπαρίσταται από τα υπόλοιπα 16 bytes της γραμμής. Η δεκαεξαδική αναπαράσταση περιέχει επίσης και τον χαρακτήρα “?” που υποδηλώνει ότι το συγκεκριμένο περιεχόμενο θα γίνει διαθέσιμο κατά την εκτέλεση του προγράμματος. Το IDA Pro είναι ένα εργαλείο στατικής ανάλυσης και συνεπώς δεν μπορεί να γνωρίζει εκ των προτέρων τη διεύθυνση που θα εισαχθεί στη συγκεκριμένη διεύθυνση μνήμης όταν εκτελεστεί το πρόγραμμα. Κατά κύριο λόγο αυτό συμβαίνει σε κλήσεις συναρτήσεων από δυναμικές βιβλιοθήκες.

3.1.3 Δομή αρχείου συμβολικής γλώσσας

Το αρχείο συμβολικής γλώσσας είναι το αρχείο που παράγεται από τον αποσυμβολομεταφραστή του IDA Pro και περιέχει όλο τον κώδικα συμβολικής γλώσσας του εκτελέσιμου μαζί με ένα πλήθος μεταδεδομένων. Η αποσυμβολομετάφραση παρουσιάζεται με γραμμικό τρόπο και οι διευθύνσεις που εμφανίζονται σε αυτήν είναι όλες *εικονικές διευθύνσεις* (virtual addresses). Συγκεκριμένα, κάθε γραμμή του αρχείου αυτού ακολουθεί την παρακάτω μορφή:

[Όνομα Τμήματος]:[Εικονική Διεύθυνση] {Ετικέτα Διεύθυνσης} {Εντολή} ; Σχόλια

Ένα παράδειγμα αρχείου συμβολικής γλώσσας παρουσιάζεται στο Σχήμα 3.2. Σημειώνεται ότι τα σχόλια στα αρχεία συμβολικής γλώσσας που εξήχθησαν από το IDA Pro περιέχουν ένα σημαντικό πλήθος μεταδεδομένων τα οποία εκμεταλλευτήκαμε σε μεγάλο βαθμό προκειμένου να εξάγουμε τα χαρακτηριστικά που θα λάβουν ως είσοδο οι ταξινομητές μας. Η δομή του αρχείου (μαζί με τα προσφερόμενα μεταδεδομένα) θα αναλυθεί περαιτέρω στην ενότητα της ανάλυσης των εξαγόμενων χαρακτηριστικών (Ενότητα 3.2)

```
.text:00401050 ; ======== S U B R O U T I N E ========================
.text:00401050
.text:00401050 ; Attributes: bp-based frame
.text:00401050
.text:00401050 foo     proc near      ; CODE XREF: demo_stackframe+2A↓p
.text:00401050
.text:00401050 arg_0   = dword ptr  8
.text:00401050 arg_4   = dword ptr  0Ch
.text:00401050
.text:00401050         push    ebp
.text:00401051         mov     ebp, esp
```

Σχήμα 3.2: Δομή αρχείου συμβολικής γλώσσας

3.2 Ανάλυση Εξαγόμενων Χαρακτηριστικών

Η μεγαλύτερη πρόκληση της παρούσας εργασίας ήταν η εξαγωγή χρήσιμων χαρακτηριστικών από τα αρχεία εισόδου. Στόχος είναι η εξαγωγή χαρακτηριστικών που έχουν έχουν κάποιο διαισθητικό νόημα ώστε να αυξήσουμε τις πιθανότητες γενίκευσης του προτεινόμενου μοντέλου. Δεδομένου ότι το πρόβλημα αποτελεί κομμάτι του τομέα Ασφάλειας Υπολογιστών εστιάσαμε σε χαρακτηριστικά τα οποία χρησιμοποιούνται συχνά από αναλυτές κακόβουλου λογισμικού κατά τη διαδικασία της στατικής ανάλυσης.

Η προσέγγιση μας στο πρόβλημα εμπεριέχει τη δημιουργία διαφορετικών συνόλων χαρακτηριστικών τα οποία στο τέλος θα αξιολογηθούν ατομικά για την εύρεση της σημαντικότητας τους αλλά και συλλογικά για να εξετάσουμε πως επηρεάζεται η απόδοση του συστήματος με συνδυασμό τους. Επιπλέον, διαίτερη έμφαση δόθηκε στην υπολογιστική απόδοση οπότε τα σύνολα χαρακτηριστικών θέλουμε να είναι μικρά σε πλήθος και να εξάγονται αρκετά γρήγορα ώστε το μοντέλο αυτό να μπορεί να εφαρμοστεί σε κατηγοριοποίηση *πραγματικού χρόνου* (real-time).

Η παρούσα ενότητα λοιπόν απαρτίζεται από υποενότητες όπου η καθεμία αναφέρεται σε ένα σύνολο χαρακτηριστικών και παρουσιάζει τη θεωρία και τα κίνητρα πίσω από την επιλογή του.

3.2.1 Καταχωρητές

Οι *καταχωρητές* (registers) είναι τοπικές περιοχές προσωρινής μνήμης μέσα στον επεξεργαστή, οι οποίες χρησιμοποιούνται για να διατηρούν μικρό πλήθος δεδομένων και να το επεξεργάζονται εκείνη τη στιγμή. Η κάθε αρχιτεκτονική διαθέτει καταχωρητές οι οποίοι συνήθως διαχωρίζονται μεταξύ τους ανάλογα με την λειτουργία την οποία επιτελούν και το είδος των δεδομένων που διατηρούν. Ο επεξεργαστής προκειμένου να εκτελέσει εντολές πρέπει να φέρει τα δεδομένα από τη μνήμη στους καταχωρητές και συνεπώς ο κώδικας assembly φανερώνει ποιοι καταχωρητές εμπλέκονται σε κάθε εντολή.

Δεδομένου, ότι το σύνολο δεδομένων αποτελείται από αρχεία που προορίζονται για την αρχιτεκτονική x86 θα κάνουμε μια σύντομη αναφορά στις ονομασίες και αντίστοιχες λειτουργίες των καταχωρητών της συγκεκριμένης αρχιτεκτονικής. Αρχικά οι καταχωρητές χωρίζονται σε 4 βασικές κατηγορίες:

- ***Καταχωρητές Γενικού Σκοπού*** (General Registers): Χρησιμοποιούνται για την εκτέλεση εντολών στον επεξεργαστή.
- ***Καταχωρητές Τμημάτων*** (Section Registers): Συντονίζουν τα τμήματα μνήμης.

- ***Σημαίες Κατάστασης*** (Status Flags): Χρησιμοποιούνται για λήψη αποφάσεων, κυρίως σε συνθήκες ελέγχου ροής.
- ***Δείκτες Εντολών*** (Instruction Pointers): Περιέχουν την διεύθυνση μνήμης της επόμενης προς εκτέλεση εντολής.

Για τις ανάγκες του συστήματος μας επιλέξαμε να εστιάσουμε μόνο στους καταχωρητές γενικού σκοπού της αρχιτεκτονικής x86. Συγκεκριμένα, οι καταχωρητές EAX, EBX, ECX, EDX αποθηκεύουν δεδομένα και διευθύνσεις μνήμης, και συνήθως χρησιμοποιούνται εναλλάξ στις διάφορες εντολές του προγράμματος. Ωστόσο, παρότι ονομάζονται γενικού σκοπού, η αρχιτεκτονική x86 χρησιμοποιεί κάποιους από αυτούς για συγκεκριμένες λειτουργίες. Για παράδειγμα, οι εντολές πολλαπλασιασμού και διαίρεσης χρησιμοποιούν πάντα τους καταχωρητές EAX και EDX, ενώ ο καταχωρητής EAX συνήθως περιέχει επίσης και την *τιμή επιστροφής* (return value) μιας συνάρτησης.

Η μνήμη των τοπικών μεταβλητών και συναρτήσεων αποθηκεύεται σε έναν ειδικό χώρο της μνήμης του προγράμματος που ονομάζεται *στοίβα* (stack), η οποία αποτελεί μια *δομή δεδομένων* (data structure) βασισμένη στη λογική *τελευταίο μέσα πρώτο έξω* (Last In First Out). Η αρχιτεκτονική x86 παρέχει ενσωματωμένη υποστήριξη για τη διαχείριση του μηχανισμού της στοίβας. Συγκεκριμένα, η παραπάνω υποστήριξη καθίσταται εφικτή μέσω των καταχωρητών ESP και EBP. Ο καταχωρητής ESP είναι ένας δείκτης στην αρχή της στοίβας, δηλαδή περιέχει τη διεύθυνση μνήμης στην οποία βρίσκεται το πρώτο στοιχείο της στοίβας. Η τιμή του συγκεκριμένου καταχωρητή αλλάζει όσο προστίθενται και αφαιρούνται δεδομένα από τη στοίβα. Αντίθετα, ο καταχωρητής EBP παραμένει σταθερός ανά συνάρτηση ώστε το πρόγραμμα να είναι σε θέση να γνωρίζει τη διεύθυνση που βρίσκονται οι τοπικές μεταβλητές και οι παράμετροι της συνάρτησης. Κάποιες συνήθεις εντολές που αφορούν την στοίβα είναι οι: *push, pop, call, leave, enter* και *ret*. Επίσης, πρέπει να σημειωθεί ότι συνήθως οι μεταγλωττιστές κάνουν αναφορά στις τοπικές μεταβλητές μέσω της τιμής του καταχωρητή EBP αλλά αυτό δεν αποτελεί κανόνα διότι είναι σχεδιαστική απόφαση του συγγραφέα του μεταγλωττιστή.

Οι *εντολές επανάληψης* (rep instructions) είναι ένα σύνολο εντολών που αποσκοπούν στη διαχείριση της *προσωρινής μνήμης δεδομένων* (data buffer). Για παράδειγμα, τέτοιες εντολές αποσκοπούν στον σχηματισμό, την αντιγραφή ή τη σύγκριση πινάκων από bytes ή λέξεις. Οι καταχωρητές EDI, ESI χρησιμοποιούνται για τις παραπάνω εντολές. Συγκεκριμένα, ο καταχωρητής ESI είναι δείκτης προέλευσης ενώ ο EDI είναι δείκτης προορισμού.

Συνοψίζοντας, οι καταχωρητές με τους οποίους θα ασχοληθούμε προκειμένου να εξάγουμε χρήσιμες πληροφορίες για το πρόβλημα μας παρουσιάζονται στον Πίνακα 3.1.

Καταχωρητές Γενικού Σκοπού x86							
eax	ebx	ecx	edx	esp	ebp	esi	edi

Πίνακας 3.1: Λίστα καταχωρητών που μελετήθηκαν και χρησιμοποιήθηκαν

Οι τιμές των καταχωρητών χρησιμοποιούνται κυρίως σε τεχνικές δυναμικής ανάλυσης [Ghia15, Ghia12] όπου παρατηρούνται οι κατανομές και οι εναλλαγές των τιμών τους και έπειτα συγκρίνονται με την αρχική τους κατάσταση. Επειδή η στατική ανάλυση δεν μας προσφέρει αυτή την πολυτέλεια θα περιοριστούμε στην καταγραφή της συχνότητας εμφάνισης του κάθε καταχωρητή και θα αγνοήσουμε τις τιμές που λαμβάνουν.

Γενικά, η κατανομή της συχνότητας εμφάνισης του κάθε καταχωρητή σε ένα πρόγραμμα οφείλεται κατά κύριο λόγο στον *μεταγλωττιστή* (compiler) ωστόσο πολλές φορές οι συγγραφείς κακόβουλου λογισμικού επινοούν τεχνικές που επηρεάζουν την παραπάνω κατανομή

προκειμένου να αποφύγουν την ανίχνευση από ευριστικές μεθόδους. Η *μετονομασία καταχωρητών* (register renaming) αποτελεί μια συνήθης τεχνική συσκότισης κώδικα και συνήθως επιτυγχάνεται μέσω εντολών που είναι περιττές και ταυτόχρονα δεν επηρεάζουν την λειτουργία του προγράμματος (π.χ *xchg, nop, mov ax 0, sub ax 0*). Συνεπώς, μια ασυνήθιστη κατανομή των καταχωρητών είναι ικανή να φανερώσει προσπάθειες συσκότισης κώδικα του κακόβουλου λογισμικού [Kaus]. Για παράδειγμα, έχει αποδειχθεί ότι η εντολή *mov* είναι *Turing-complete* [Dola] και έχει υλοποιηθεί μεταγλωττιστής που παράγει κώδικα μηχανής που αποτελείται μόνο από εντολές *mov*.

Το σύνολο χαρακτηριστικών που προέκυψαν από τους καταχωρητές αποτελείται από τη συχνότητα εμφάνισης των καταχωρητών γενικού σκοπού. Επιπλέον, συλλέξαμε το ποσοστό συμμετοχής του κάθε καταχωρητή επί του συνόλου εντολών. Αναλυτικότερα, το σύνολο χαρακτηριστικών που εξήχθησαν από τους καταχωρητές παρουσιάζεται στον Πίνακα 3.2.

Όνομα Χαρακτηριστικού	***Περιγραφή***
{eax,ebx,ecx,edx,esi,edi,ebp,esp}_freq	Συχνότητα εμφάνισης καταχωρητή
{eax,ebx,ecx,edx,esi,edi,ebp,esp}_crb	Συμμετοχή καταχωρητή επί συνόλου εντολών

Πίνακας 3.2: Σύνολο χαρακτηριστικών καταχωρητών “reg_featset”

3.2.2 Τμήματα

Όπως αναφέρθηκε στο Κεφάλαιο 2, το αρχέτυπο PE περιέχει μια επικεφαλίδα, η οποία ακολουθείται από μια σειρά από *τμήματα* (sections). Η επικεφαλίδα περιέχει *μεταδεδομένα* (metadata) που αφορούν το αρχείο. Μετά την επικεφαλίδα υπάρχουν τα πραγματικά τμήματα του αρχείου, καθένα από το οποία περιέχει χρήσιμες πληροφορίες. Τα πιο συνήθη και ενδιαφέροντα τμήματα είναι τα εξής:

- **.text**: Το τμήμα *.text* περιέχει τις εντολές που θα εκτελέσει ο επεξεργαστής. Όλα τα υπόλοιπα τμήματα περιέχουν δεδομένα και συμπληρωματικές πληροφορίες. Αυτό το τμήμα είναι το μόνο που έχει δικαιώματα εκτέλεσης και πρέπει να είναι και το μόνο που περιέχει εκτελέσιμο κώδικα.

- **.rdata**: Το τμήμα *.rdata* περιέχει κυρίως πληροφορίες για τα imports και τα exports. Τα imports είναι συναρτήσεις άλλων βιβλιοθηκών που χρησιμοποιούνται από το πρόγραμμα ενώ τα exports είναι συναρτήσεις του προγράμματος που πρόκειται να κληθούν από άλλα προγράμματα ή βιβλιοθήκες. Στο τμήμα *.data* μπορούν να αποθηκευτούν και άλλα δεδομένα μόνο για ανάγνωση που χρησιμοποιούνται από το πρόγραμμα. Κάποιες φορές το αρχείο θα περιέχει τα τμήματα *.idata* και *.edata* για τα imports και τα exports αντίστοιχα.

- **.data**: Το τμήμα *.data* περιέχει τα *καθολικά* (global) δεδομένα του προγράμματος, τα οποία είναι προσβάσιμα από οπουδήποτε. Τα *τοπικά* (local) δεδομένα δεν αποθηκεύονται σε κάποια τμήμα διότι διατηρούνται στην στοίβα του προγράμματος.

- **.rsrc**: Το τμήμα *.rsrc* περιέχει τους πόρους που χρησιμοποιούνται από το εκτελέσιμο όπως εικονίδια, εικόνες, μενού και *συμβολοσειρές* (strings). Οι συμβολοσειρές μπορούν να αποθηκευτούν είτε στο *.rsrc* τμήμα είτε στο κύριο πρόγραμμα αλλά συνήθως αποθηκεύονται στο *.rsrc* για υποστήριξη πολλαπλών γλωσσών.

Τμήμα	Περιγραφή περιεχομένου τμήματος
.bss	Μη-αρχικοποιημένα *στατικά* (statically-allocated) δεδομένα
.sbss	*"Μικρά δεδομένα"* (small data), τα οποία προσπελάζονται από εντολές μικρότερου μεγέθους που έχουν πρόσβαση σε περιορισμένο εύρος διευθύνσεων. Οι διευθύνσεις αυτές είναι σχετικές ως προς έναν καθολικό δείκτη που ονομάζεται GP και αυτός ο τρόπος διευθυνσιοδότησης αναφέρεται ως GP-relative addressing.
.cormeta	Μεταδεδομένα CLR (Common Language Runtime)
.data	Καθολικά δεδομένα
.drective	Πρόσθετες επιλογές που αφορούν τον συνδετή
.text	Περιέχει τον εκτελέσιμο κώδικα
.rdata	Δεδομένα μόνο για ανάγνωση προσβάσιμα από όλο το εκτελέσιμο
.idata	Πληροφορίες για τα imports συναρτήσεων
.edata	Πληροφορίες για τα exports συναρτήσεων
.pdata	Υπάρχει μόνο στα 64-bit εκτελέσιμα και διατηρεί πληροφορίες για χειρισμό εξαιρέσεων
.sdata	Αρχικοποιημένα «μικρά δεδομένα», τα οποία προσπελάζονται μέσω GP-relative addressing. Αντίστοιχο του τμήματος *.sbss* αλλά για αρχικοποιημένα δεδομένα
.xdata	Πληροφορίες σχετικά με τις εξαιρέσεις του προγράμματος
.sxdata	Καταχωρημένα δεδομένα διαχείρισης εξαιρέσεων
.rsrc	Πόροι που χρειάζονται από το εκτελέσιμο
.reloc	Πληροφορίες μετακίνησης αρχείων βιβλιοθήκης
.orpc	Δεδομένα που αφορούν *απομακρυσμένη κλήση διαδικασίας* (remote procedure call)
.tls	Δεδομένα που αφορούν τον *τοπικό χώρο αποθήκευσης νημάτων* (thread local storage). Ο τοπικός χώρος αποθήκευσης νημάτων είναι μια ειδική περιοχή της μνήμης όπου ένα αντικείμενο δεν είναι μεταβλητή που αποθηκεύεται στη στοίβα, όμως είναι τοπική για κάθε νήμα που εκτελεί τον κώδικα

Πίνακας 3.3: Λίστα τμημάτων αρχέτυπου PE

```
.data:00437000                          ; Section 3. (virtual address 00037000)
.data:00437000                          ; Virtual size              : 000EE68C ( 976524.)
.data:00437000                          ; Section size in file          : 000EDA00 ( 973312.)
.data:00437000                          ; Offset to raw data for section: 00035200
.data:00437000                          ; Flags C0000040: Data Readable Writable
.data:00437000                          ; Alignment     : default
.data:00437000                          ; ===========================================================================
.data:00437000
.data:00437000                          ; Segment type: Pure data
.data:00437000                          ; Segment permissions: Read/Write
```

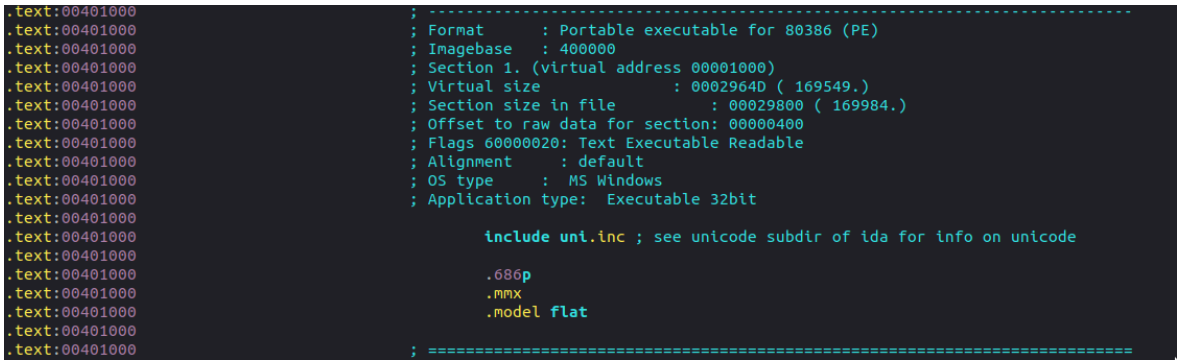
Σχήμα 3.3: Διαθέσιμα μεταδεδομένα του IDA Pro για το τμήμα .data

Στο σύστημα μας επιλέξαμε να συλλέξουμε χαρακτηριστικά που αφορούσαν κάποια τμήματα τα οποία είναι γνωστά και επιτελούν συγκεκριμένες λειτουργίες. Όλα τα τμήματα με τα οποία ασχοληθήκαμε παρουσιάζονται συνοπτικά στον Πίνακα 3.3. Να σημειωθεί ότι στα τμήματα αυτά συμπεριλαμβάνονται τόσο τα τμήματα που αναφέρονται στις προδιαγραφές της μορφής του αρχέτυπου PE, όσο και κάποια τμήματα που εισάγονται από τους μεταγλωττιστές για τυχόν βελτιστοποιήσεις.

Εξαγωγή πληροφορίας

Προκειμένου να εξάγουμε πληροφορίες από τα τμήματα που θέλουμε να μελετήσουμε πρέπει να αναλυθούν τα μεταδεδομένα που παρέχονται από τον αποσυμβολομεταφραστή του IDA Pro σχετικά με τα τμήματα του εκτελέσιμου αρχείου. Ενδεικτικά, στα Σχήματα 3.3-3.4 παρουσιάζονται κάποια μεταδεδομένα που έχουμε στη διάθεση μας από το αρχείο assembly στην αρχικοποίηση του κάθε τμήματος.

Παρακάτω αναφέρονται οι πληροφορίες των διαθέσιμων μεταδεδομένων με τις οποίες θα ασχοληθούμε:



Σχήμα 3.4: Διαθέσιμα μεταδεδομένα του IDA Pro για το τμήμα .text

- **Virtual size**: Είναι το εικονικό μέγεθος του τμήματος και αναφέρει πόσος χώρος δεσμεύεται για το τμήμα κατά τη διαδικασίας φόρτωσης του προγράμματος. Ουσιαστικά, αναπαριστά το μέγεθος του τμήματος όταν το πρόγραμμα θα φορτωθεί στη μνήμη.
- **Section size in file** : Είναι το πραγματικό μέγεθος του τμήματος στο δίσκο. Συνήθως είναι παρόμοιο με το εικονικό μέγεθος.

Για κάθε τμήμα συλλέξαμε το εικονικό και το πραγματικό του μέγεθος. Οι περισσότερες προσεγγίσεις καταμετρούν το ποσοστό του πλήθους γραμμών επί των συνολικός γραμμών του αρχείου assembly [Ahma16], ωστόσο επιλέξαμε να παραλείψουμε αυτό το χαρακτηριστικό διότι το πλήθος γραμμών είναι κάτι που εξαρτάται και από τον αποσυμβολομεταφραστή και δεν γενικεύεται. Παρακάτω, ακολουθεί αναφορά στα χαρακτηριστικά που εξήχθησαν:

1. **Πλήθος τμημάτων**: Ο αριθμός των διαφορετικών τμημάτων που εμφανίζονται στο εκτελέσιμο.
2. **Ποσοστό άγνωστων τμημάτων** Το ποσοστό των τμημάτων που τα ονόματα τους είναι άγνωστα (συνήθως οι ονομασίες των τμημάτων αυτών είναι συσκοτισμένες από τον συντάκτη του κακόβουλου λογισμικού). Άγνωστα τμήματα θεωρούμε όσα δεν παρουσιάζονται στον Πίνακα 3.5
3. **Ύπαρξη τμημάτων packing**: Μια τιμή αληθείας που αναπαριστά την ύπαρξη ή όχι τμημάτων που προκύπτουν από γνωστά λογισμικά packing. Συγκεκριμένα, μια λίστα (όχι εξαντλητική) τέτοιων τμημάτων παρουσιάζονται στον Πίνακα 3.4
4. **Ποσοστό εικονικού μεγέθους τμήματος**: Η συνεισφορά (ποσοστό) του συγκεκριμένου τμήματος στη συνολική εικονική μνήμη που δεσμεύτηκε από όλο το πρόγραμμα
5. **Ποσοστό εικονικού μεγέθους άγνωστων τμημάτων**: Το ποσοστό της εικονικής μνήμη που δεσμεύτηκε από άγνωστα τμήματα ως προς τη συνολική εικονική μνήμη που δεσμεύτηκε στο πρόγραμμα.
6. **Ομοιότητα εικονικού-φυσικού μεγέθους τμήματος**: Ένας δείκτης ομοιότητας που εκφράζει το πόσο κοντά είναι οι τιμές μεταξύ εικονικού και φυσικού μεγέθους κάπου τμήματος. Συγκεκριμένα, χρησιμοποιήσαμε το δείκτη:

$$VSRS\_similarity(section) = \frac{VirtualSize(section) - RawSize(section)}{RawSize(section)} \quad (3.1)$$

.aspack	.aspack1	.aspack3	.aspack6	.pack1	.unpack	.adata

Πίνακας 3.4: Ονόματα τμημάτων γνωστού λογισμικού packing

.data1	.pdata	.edata	.sbss	.reloc	DATA	.rsrc
.xdata	CODE	.data	.sdata	_RDATA	.tls	.idata
.orpc	.rdata	.text	.bss	.sxdata	BSS	.text1

Πίνακας 3.5: Έγκυρα τμήματα αρχέτυπου PE

Συσκοτισμένα Τμήματα

Κάποιες φορές τα ονόματα των τμημάτων συσκοτίζονται προκειμένου να δυσκολέψουν τον αναλυτή κακόβουλου λογισμικού [Lee10]. Προκειμένου, να διαχειριστούμε αυτή την περίπτωση συγκεντρώσαμε όλα τα ονόματα των έγκυρων τμημάτων. που ορίζονται στις προδιαγραφές του αρχέτυπου PE, και οποιοδήποτε τμήμα δεν ανήκει σε αυτά κατηγοριοποιείται ως άγνωστο τμήμα. Μέσω της καταμέτρησης των άγνωστων τμημάτων καθώς και της συμμετοχής τους επί του συνολικού εκτελέσιμου μπορούμε να μοντελοποιήσουμε καλύτερα τις προθέσεις του κακόβουλου λογισμικού και συνεπακόλουθα να προβλέψουμε την "οικογένεια" στην οποία ανήκει.

Πακετάρισμα (Packing)

Στην παρουσίαση των μεταδεδομένων που έχουμε στη διάθεση μας για κάθε τμήμα αναφέραμε ότι συνήθως το εικονικό μέγεθος και το πραγματικό μέγεθος του προγράμματος στο δίσκο είναι περίπου ίδια. Προφανώς, μπορεί να υπάρχουν μικρές διαφορές μεταξύ αυτών των μεγεθών λόγω των ευθυγραμμίσεων στην μνήμη και στον δίσκο. Ωστόσο, τα μεγέθη των τμημάτων αποτελούν ένα δυνατό χαρακτηριστικό ανίχνευσης πακεταρισμένων προγραμμάτων. Για παράδειγμα, εάν η τιμή του "Virtual Size" είναι πολύ μεγαλύτερη από την τιμή του "Section size in file", τότε το μέγεθος του τμήματος λαμβάνει πολύ περισσότερο χώρο στη μνήμη σε σχέση με το δίσκο. Αυτή η συμπεριφορά υποδηλώνει ότι το πρόγραμμα είναι συμπιεσμένο και μέσω του δείκτη ομοιότητας που ορίσαμε προηγουμένως μεταξύ των δυο παραπάνω μεγεθών, μπορούμε να εξάγουμε πληροφορία σχετικά με το πακετάρισμα του αρχείου. Η κάθε οικογένεια κακόβουλου λογισμικού επιτελεί συγκεκριμένες λειτουργίες και επιδέχεται συμπίεση ως ένα βαθμό. Ο δείκτης ομοιότητας λοιπόν εμπεριέχει και την πληροφορία του ποσοστού συμπίεσης που επιτεύχθηκε και εν τέλει αποτέλεσε ένα ιδιαίτερα ισχυρό χαρακτηριστικό στο πρόβλημα της κατηγοριοποίησης.

3.2.3 Κωδικοποιήσεις Εντολών

Κάθε *εντολή* (instruction) αντιστοιχεί σε μια κωδικοποίηση εντολής (operation code) και ενημερώνει την *κεντρική μονάδα επεξεργασίας* (KME - central processing unit) για την ενέργεια που το πρόγραμμα θέλει να πραγματοποιήσει.

Η κάθε αρχιτεκτονική διαθέτει ένα *σύνολο εντολών* (instruction set) το οποίο παρέχεται από τον κατασκευαστή της κάθε αρχιτεκτονικής και προσδιορίζει τις ενέργειες που υποστηρίζονται και διεκπεραιώνονται από τη συγκεκριμένη KME. Το γεγονός ότι η κάθε αρχιτεκτονική διαθέτει τις δικές τις κωδικοποιήσεις ενέχει τον περιορισμό ότι εάν εισάγουμε χαρακτηριστικά που αφορούν τα opcodes δεν θα μπορούμε να γενικεύσουμε τη λύση για πλήθος αρχιτεκτονικών. Ωστόσο, τα περισσότερα χαρακτηριστικά κατά την ανάλυση κακόβουλου

λογισμικό είναι άρρηκτα συνδεδεμένα με την αρχιτεκτονική για την οποία προορίζεται το εκάστοτε εκτελέσιμο.

Λαμβάνοντας υπόψη το γεγονός ότι τα opcodes αποτελούν τις εντολές που εκτελούνται στην ΚΜΕ μπορούμε να καταλάβουμε διαισθητικά την σημασία τους στην κατηγοριοποίηση κακόβουλου λογισμικού. Μέσα από αυτές μπορούμε να εντοπίσουμε *μοτίβα* (patterns) που προδίδουν την κακόβουλη φύση του λογισμικού. Συγκεκριμένα, στη βιβλιογραφία υπάρχουν πολλές δημοσιεύσεις που έχουν επισημάνει την σημασία των opcodes στην αναγνώριση κακόβουλου λογισμικού [Abou04, Schu01].

N-grams

Τα n-grams αποτελούν τα βασικά χαρακτηριστικά που εξάγονται σε ακολουθιακές προσεγγίσεις αναγνώρισης κακόβουλου λογισμικού. Τα αποτελέσματα από προηγούμενες ερευνητικές εργασίες πάνω στο θέμα έχουν δείξει ότι παρόλο που τα n-grams μικρού μήκους ($n = 1, n = 2$) μπορούν να εξαχθούν πολύ γρήγορα, η συμπεριφορά των εκτελέσιμων μπορεί να φανερωθεί καλύτερα κυρίως για μεγαλύτερες τιμές του n [Parv11]. Ωστόσο, με αύξηση του μήκους ενός n-gram, ο χώρος χαρακτηριστικών μεγαλώνει εκθετικά και ενέχει σημαντικές χωρικές και υπολογιστικές απαιτήσεις πόρων.

Λαμβάνοντας υπόψη τους υπολογιστικούς περιορισμούς επιλέξαμε να χρησιμοποιήσουμε n-grams μόνο μήκους $n = 1$ που εξάγονται γρήγορα και ουσιαστικά αποτυπώνουν για κάθε αρχείο τη συχνότητα εμφάνισης του κάθε opcode.

Προκειμένου να εξάγουμε τα opcodes διασχίσαμε το αρχείο και συγκεντρώσαμε αρχικά όλα τα δυνατά opcodes που εμφανίζονται έστω και σε ένα αρχείο. Έπειτα, μετρήσαμε το πλήθος εμφανίσεων του κάθε opcode επί του συνόλου των αρχείων εκπαίδευσης. Το συνολικό πλήθος opcode που εξήχθησαν ισούται με 2811, αριθμός ο οποίος αντιβαίνει με τους στόχους μας να ελαχιστοποιήσουμε το πλήθος των χαρακτηριστικών και των υπολογιστικών απαιτήσεων. Επομένως, στοχεύσαμε να μειώσουμε αισθητά το παραπάνω μέγεθος.

Οι περισσότερες προσεγγίσεις, τόσο στο διαγωνισμό του Kaggle[2] όσο και στη βιβλιογραφία, επέλεγαν να προσμετρούν τα opcodes που εμφανίζονται τουλάχιστον k φορές σε κάθε αρχείο ή εκείνα τα οποία εμφανίζονται πάνω από l φορές σε τουλάχιστον ένα αρχείο. Επιπλέον, κάποιες από τις νικητήριες ομάδες επέλεξαν τα opcodes που απλώς βελτιστοποιούν τα τελικά τους αποτελέσματα. Τέτοιες προσεγγίσεις, όμως, ενέχουν τον κίνδυνο της δυσκολίας γενίκευσης. Η δική μας προσέγγιση βασίστηκε σε στατιστική ανάλυση που έχει γίνει σε μεγάλο δείγμα κακόβουλου λογισμικού και αναδεικνύει τα opcodes των οποίων η συχνότητα φανερώνει διαφοροποιήσεις μεταξύ διαφορετικών οικογενειών κακόβουλου λογισμικού [Bila06]. Συγκεκριμένα, στην παραπάνω ανάλυση παρατηρήθηκε ότι η συχνότητα των 14 πιο συχνών opcodes αποτελεί έναν μέτριο ταξινομητή σε αντίθεση με τα 14 πιο σπάνια τα οποία φανέρωναν περισσότερη πληροφορία. Εμείς εντάξαμε σε αυτό το σύνολο χαρακτηριστικών και τα πιο συχνά opcodes αλλά και τα πιο σπάνια. Ωστόσο, εκείνα τα οποία εμφανίζονταν πολύ αραιά αγνοήθηκαν διότι αλλοίωναν σημαντικά τα τελικά αποτελέσματα. Εν τέλει, συλλέξαμε 85 opcodes.

Σημειώνεται ότι τα opcodes σε περιπτώσεις συσκότισης κώδικα και πακεταρίσματος μπορεί να οδηγήσουν την διαδικασία κατηγοριοποίησης σε λάθος κατεύθυνση. Ωστόσο, αντιμετωπίζουμε το συγκεκριμένο πρόβλημα δημιουργώντας στη συνέχεια νέα σύνολα χαρακτηριστικών που στοχεύουν τον εντοπισμό τέτοιων περιπτώσεων ώστε το τελικό μας σύστημα να ταξινομεί σωστά ακόμα και αρχεία που αποσκοπούν στην δυσκολία ανίχνευσης τους.

[2] https://www.kaggle.com/c/malware-classification

Συνοψίζοντας, αφού αποφασίσαμε τα 85 opcodes που θα αποτελέσουν το σύνολο χαρακτηριστικών μας επιλέξαμε να σπάσουμε αυτό το σύνολο σε 2 σύνολα όπου στο ένα θα μετράμε τη συχνότητα εμφάνισης των παραπάνω opcode ενώ στο δεύτερο θα μετράμε τον TF-IDF δείκτη τους.

TF-IDF (Term Frequency - Inverse Document Frequency)

Το TF-IDF είναι μια στατιστική μετρική που αποσκοπεί στο να αναδεικνύει τη σημαντικότητα μιας λέξης ως προς ένα αρχείο δεδομένης μιας συλλογής αρχείων. Χρησιμοποιείται συχνά στον τομέα της *Ανάκτησης Πληροφορίας* (Information Retrieval) και σε προβλήματα *επεξεργασίας φυσικής γλώσσας* (natural language processing).

Η τιμή της συγκεκριμένης μετρικής απαρτίζεται από δυο βασικά μέρη:

- Term Frequency: Αναπαριστά τη συχνότητα εμφάνισης του συγκεκριμένου όρου σε κάποιο αρχείο. Στην προκειμένη περίπτωση ο όρος ταυτίζεται με κάποιο opcode. Η συχνότητα εμφάνισης του όρου υπολογίζεται από τον παρακάτω τύπο:

$$TF(x) = \frac{\text{Πλήθος εμφανίσεων του όρου x στο αρχείο}}{\text{Πλήθος εμφανίσεων όλων των όρων του αρχείου}} \tag{3.2}$$

- Inverse Term Frequency: Εκφράζει τη σημασία του όρου στο πρόβλημα της κατηγοριοποίησης ή αλλιώς τη διαχωριστική ικανότητα του όρου. Η αντίστροφη συχνότητα ενός όρου υπολογίζεται από τον παρακάτω τύπο:

$$IDF(x) = \ln(\frac{\text{Συνολικό πλήθος αρχείων}}{\text{Πλήθος αρχείων που περιέχουν τον όρο x}}) \tag{3.3}$$

Η μετρική TF-IDF προκύπτει εντέλει ως το γινόμενο των δυο παραπάνω όρων:

$$TF - IDF(x) = TF(x) \cdot IDF(x) \tag{3.4}$$

Συνεπώς, δημιουργήσαμε δυο σύνολα χαρακτηριστικών όπου το ένα μετράει το πλήθος εμφάνισης των 85 opcodes ενώ το δεύτερο υπολογίζει το TF-IDF των opcodes. Η τελική τους αξιολόγηση θα πραγματοποιηθεί στο στάδιο επιλογής χαρακτηριστικών. Σημειώνεται ότι τα n-grams είναι *εδραιωμένη μέθοδος* (state-of-the-art) στον τομέα ανίχνευσης κακόβολου λογισμικού και αποτελεί μια ασφαλή επιλογή για ικανοποιητικά αποτελέσματα.

3.2.4 Byte N-grams

Τα byte n-grams έχουν χρησιμοποιηθεί ως χαρακτηριστικά σε αρκετές εργασίες και αποτελούν έναν από τα πιο κοινά χαρακτηριστικά της στατικής ανάλυσης λογισμικού [Shab09]. Τα περισσότερα πειράματα από άλλες εργασίες εξάγουν τα n-grams για $n = 1$ έως $n = 8$ και γενικά έχει αναφερθεί ότι η αποτελεσματικότητα τους παρατηρείται για $n \geq 2$ [Shab09]. Λόγω της εκθετικής αύξησης του πλήθους των χαρακτηριστικών για $n > 1$ οι περισσότερες δημοσιεύσεις ακολουθούν την μέθοδο του Kolter και Maloof [Kolt04]. Η παραπάνω μέθοδος περιέχει αρχικά την επιλογή μιας τιμής του n και έπειτα χρησιμοποιείται ένα *σχέδιο κατάταξης χαρακτηριστικών* (feature ranking scheme) ώστε να επιλεγούν τα πρώτα k σε πλήθος n-grams ως χαρακτηριστικά για την εκμάθηση του μοντέλου.

Λόγω της απαίτησης του συστήματος μας για υψηλή ταχύτητα εξαγωγής χαρακτηριστικών επιλέξαμε να υπολογίσουμε μόνο τα n-grams μήκους $n = 1$ (συχνότητα εμφάνισης του

κάθε byte). Με αυτόν τον τρόπο θα επιχειρήσουμε να αμφισβητήσουμε τους ισχυρισμούς που θέλουν τα n-grams να είναι αποτελεσματικά μόνο για $n \geq 2$ [Shab09]. Συγκεκριμένα, θα αποδείξουμε ότι παρόλο που η συχνότητα του κάθε byte δεν μπορεί να θεωρηθεί ένα ισχυρό αυτόνομο χαρακτηριστικό κατηγοριοποίησης κακόβουλου λογισμικού, ο συνδυασμός τους με τα υπόλοιπα χαρακτηριστικά που έχουμε συλλέξει συνεισφέρει στην βελτίωση της ακρίβειας των μοντέλων μας. Δεδομένου ότι το πλήθος των δυνατών bytes ισούται με 256, ο αριθμός των χαρακτηριστικών που θα προστεθούν είναι μικρός και έτσι αποφεύγουμε την ανάγκη μηχανισμού ταξινόμησης χαρακτηριστικών που ενέχει υψηλό κίνδυνο το μοντέλο να μην γενικεύεται ικανοποιητικά σε νέα δεδομένα.

Όπως και με τα n-grams των opcodes, δημιουργήσαμε δυο διαφορετικά σύνολα χαρακτηριστικών. Το πρώτο θα περιέχει τη συχνότητα εμφάνισης του κάθε byte ενώ το δεύτερο θα περιέχει το TF-IDF του κάθε byte. Στο τελικό στάδιο της επιλογής των συνόλων χαρακτηριστικών θα αξιολογήσουμε ποιο από τα δυο ανταποκρίνεται καλύτερα στο πρόβλημα της κατηγοριοποίησης κακόβουλου λογισμικού.

3.2.5 Εντροπία

Η *εντροπία* (entropy), όπως αναφέρθηκε και στο Κεφάλαιο 2, αποτελεί θεμελιώδη έννοια της θεωρίας πληροφορίας και αναπαριστά την αβεβαιότητα που διακατέχει ένα σύστημα. Στην παρούσα εργασία επιλέξαμε να εξάγουμε χαρακτηριστικά από την εντροπία που παρατηρήθηκε στα δεκαεξαδικά αρχεία σε επίπεδο byte.

Η εντροπία αποτελεί ένα χαρακτηριστικό το οποίο έχει χρησιμοποιηθεί σε αρκετές ερευνητικές εργασίες που αφορούν την ανίχνευση και ταξινόμηση κακόβουλου λογισμικού [Cesa13, Perd08]. Επίσης, έχει αποδειχθεί ότι επιτυγχάνει πολύ καλά αποτελέσματα στην αναγνώριση συσκοτισμένων αρχείων [Bays13, Lyda07]. Τα θετικά αποτελέσματα οφείλονται στο γεγονός ότι υψηλή εντροπία παρατηρείται συνήθως σε κρυπτογραφημένο ή συμπιεσμένο περιεχόμενο. Οπότε, ένα αρχείο με ασυνήθιστα υψηλή εντροπία είναι κατά μεγάλη πιθανότητα πακεταρισμένο, χωρίς αυτό να σημαίνει απαραίτητα ότι είναι και κακόβουλο.

Εξαγωγή Χαρακτηριστικών Εντροπίας

Η προσέγγιση μας στην εξαγωγή εντροπίας βασίστηκε στην μέθοδο *κυλιόμενου παραθύρου* (sliding window) [Bays13]. Συγκεκριμένα, κάθε δεκαεξαδικό αρχείο αναπαρίσταται ως μια σειρά από παράθυρα των k bytes. Για τιμή του k επιλέξαμε $k = 10000$. Για κάθε παράθυρο j υπολογίσαμε την εντροπία του e_j μέσω του παρακάτω τύπου:

$$e_j = -\sum_{i=0}^{n} p_i \log_2(p_i) \tag{3.5}$$

όπου το p_i αναπαριστά τη συχνότητα εμφάνισης του byte με τιμή i στο παράθυρο j. Το n είναι ο συνολικός αριθμός συμβόλων που εμφανίζονται στο παράθυρο j Ο μέγιστος αριθμός συμβόλων στην περίπτωση μας ισούται με 256 οπότε ισχύει ότι $n \leq 256$. Επιπλέον, δεδομένου του παραπάνω μέγιστου αριθμού, η Εξίσωση (3.5) λαμβάνει τιμές μεταξύ 0 και 8.

Τα χαρακτηριστικά που εξήχθησαν παρουσιάζονται συνοπτικά για λόγους μελλοντικής αναφοράς στον Πίνακα 3.6. Χρησιμοποιήσαμε λοιπόν το δεκαεξαδικό αρχείο προκειμένου να εξάγουμε τα παρακάτω χαρακτηριστικά που αφορούν την εντροπία σε επίπεδο byte:

1. **Συνολική εντροπία**: Η εντροπία του συνολικού δεκαεξαδικού αρχείου θεωρώντας όλο το αρχείο ως ένα ενιαίο παράθυρο. Για αυτό το χαρακτηριστικό δεν χρησιμοποιήθηκε η μέθοδος του κυλιόμενου παραθύρου.

2. **Μέση εντροπία**: Η μέση τιμή της εντροπίας ανά παράθυρο. Έστω M ο συνολικός αριθμός των παραθύρων του αρχείου, τότε η μέση εντροπία δίνεται από τον παρακάτω τύπο:

$$Ent\_mean(hex\_file) = \mu = \frac{\sum_{i=0}^{M-1} e_j}{M} \tag{3.6}$$

3. **Απόκλιση εντροπίας**: Η απόκλιση της εντροπίας για τον πίνακα τιμών που απαρτίζεται από τις τιμές εντροπίας όλων των παραθύρων. Η απόκλιση αποτελεί ένα μέτρο της εξάπλωσης και κατανομής των τιμών και υπολογίζεται ως εξής:

$$Ent\_var(hex\_file) = \frac{\sum_{i=0}^{M-1} |x - \mu|^2}{M} \tag{3.7}$$

   Οι υψηλές τιμές απόκλισης επιδεικνύουν ότι υπάρχουν διακυμάνσεις από παράθυρο σε παράθυρο και αυτό εμπεριέχει χρήσιμη πληροφορία για την συμπεριφορά του εκτελέσιμου.

4. **{25,50,75,99}-Εκατοστημόριο**: Το εκατοστημόριο είναι ένα στατιστικό μέτρο το οποίο υπολογίζεται επί ενός συνόλου τιμών. Συγκεκριμένα, το p-στο εκατοστημόριο είναι η τιμή εκείνη που όταν όλες οι τιμές διαταχθούν σε αύξουσα σειρά, έχει από αριστερά της το $p\%$ των δεδομένων και από δεξιά το υπόλοιπο $(100 - p)\%$. Επιπλέον, το 50-στο εκατοστημόριο επειδή χωρίζει τις τιμές σε ίσα μέρη εκατέρωθεν είναι γνωστό ως διάμεσος στη στατιστική.

Όνομα Χαρακτηριστικού	***Περιγραφή***
Συνολική εντροπία	Τιμή εντροπίας επί του συνολικού δεκαεξαδικού αρχείου
Μέση εντροπία	Μέσος όρος εντροπίας παραθύρου των 10,000 bytes
Απόκλιση εντροπίας	Απόκλιση τιμής εντροπίας επί του πλήθους παραθύρων
{25,50,75,99}-Εκατοστημόριο εντροπίας	P-στο εκατοστημόριο επί του πλήθους παραθύρων

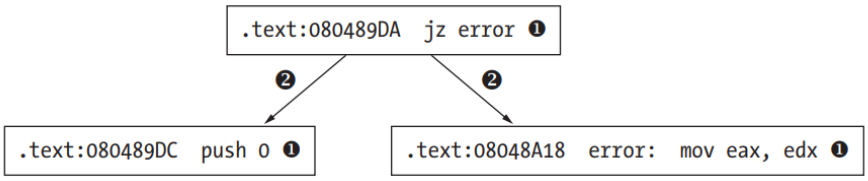
Πίνακας 3.6: Σύνολο χαρακτηριστικών εντροπίας “ent_featset”

3.2.6 Δια-παραπομπές

Οι *δια-παραπομπές* (cross-references) είναι αναφορές που γίνονται σε δεδομένα ή εντολές και αποτυπώνονται από τον αποσυμβολομεταφραστή του IDA Pro ως μεταδεδομένα στο αρχεία συμβολικής γλώσσας. Συγκεκριμένα, με τον όρο αναφορά εννοούμε ότι η διεύθυνση μνήμης κάποιας εντολής ή δεδομένου αποτελεί όρισμα κάποιας εντολής του εκτελέσιμου αρχείου.

Υπάρχουν δυο βασικές κατηγορίες δια-παραπομπών: δια-παραπομπές κώδικα και δια-παραπομπές δεδομένων. Για κάθε δια-παραπομπή ορίζεται μια *κατεύθυνση* (direction) που εκφράζει ότι μια διεύθυνση μνήμης αναφέρθηκε σε κάποια άλλη διεύθυνση μνήμης. Οι διευθύνσεις αυτές μπορούν να περιέχουν είτε δεδομένα είτε εντολές. Η δυνατότητα εξαγωγής

κατεύθυνσης ήταν αυτή που ώθησε τη κοινότητα ανάλυσης λογισμικού στην παραγωγή γραφημάτων. Για παράδειγμα, στο Σχήμα 3.5 παρουσιάζεται ένα γράφημα που παράχθηκε από τρεις εντολές συμβολικού κώδικα. Οι δια-παραπομπές στη στατική ανάλυση λογισμικού έχουν χρησιμοποιηθεί για την παραγωγή *γραφημάτων ελέγχου ροής* (control flow graphs) και *γραφημάτων κλήσεων συναρτήσεων* (function call graphs) [Kast02, Xu14]. Τα παραπάνω γραφήματα με τη σειρά τους έχει αποδειχτεί σε διάφορες εργασίες ότι είναι ιδιαίτερα χρήσιμα στην αναγνώριση κακόβουλου λογισμικού [Nair09].



Σχήμα 3.5: Γράφημα ροής εκτέλεσης προγράμματος

Όμως, δεδομένης της υψηλής πολυπλοκότητας κατασκευής των γραφημάτων αυτών επιλέξαμε να ακολουθήσουμε μια πρωτότυπη προσέγγιση που αφορά την άμεση εξαγωγή χαρακτηριστικών από τα μεταδεδομένα που αφορούν τις δια-παραπομπές και όχι την έμμεση εξαγωγή που χρησιμοποιείται συνήθως. Το πλήθος των αναφορών που γίνονται σε δεδομένα και εντολές, οι ενέργειες που πραγματοποιούνται στα δεδομένα που γίνεται η αναφορά, το ποσοστό αναγνώσεων-εγγραφών καθώς και άλλα στατιστικά μεγέθη στα οποία θα αναφερθούμε στη συνέχεια φανερώνουν στοιχεία της συμπεριφοράς του αρχείου και διευκολύνουν την κατηγοριοποίηση του.

Δια-παραπομπές Δεδομένων

Οι *δια-παραπομπές δεδομένων* (data cross-references) εκφράζουν τον τρόπο με τον οποίο προσπελάζονται τα δεδομένα ενός εκτελέσιμου αρχείου. Ο πρώτος τρόπος προσπέλασης αφορά τις *δια-παραπομπές ανάγνωσης* (read cross-reference) που υποδεικνύουν ότι το περιεχόμενο της διεύθυνσης μνήμης πρόκειται να διαβαστεί. Οι δια-παραπομπές ανάγνωσης προέρχονται μόνο από εντολές και μπορούν να αναφέρονται σε οποιαδήποτε διεύθυνση του προγράμματος. Η καθολική μεταβλητή *read_it* στο Σχήμα 3.6 αναφέρεται από τις διευθύνσεις μνήμης *_main+E* και *_main+25*, αναφέρεται από χαμηλότερες διευθύνσεις μνήμης λόγω του βέλους προς τα πάνω και κατηγοριοποιείται ως δια-παραπομπή ανάγνωσης λόγω του επιθέματος *”r”*.

Το δεύτερο είδος δια-παραπομπών δεδομένων είναι οι *δια-παραπομπές εγγραφής* (write cross-references) που αφορούν τις εγγραφές δεδομένων και μπορούν να προκύψουν μόνο από εντολές. Η καθολική μεταβλητή *write_it* στο Σχήμα 3.6 αναφέρεται από τις διευθύνσεις *_main+1B* και*_main+2E* και το επίθεμα *”w”* στο τέλος υποδηλώνει ότι η συγκεκριμένη μεταβλητή πρόκειται να εγγραφεί από τις εντολές που περιέχονται στις διευθύνσεις από τις οποίες αναφέρθηκε. Συγκεκριμένα, μια δια-παραπομπή εγγραφής που στοχεύει μια εντολή προγράμματος byte αποτελεί ένδειξη *αυτοτροποποιούμενου κώδικα* (self-modifying code) που παρατηρείται συνήθως σε ρουτίνες συσκότισης κώδικα.

Ο τρίτος τύπος αναφοράς δεδομένων ονομάζεται *δια-παραπομπή αντιστάθμισης* (offset cross-reference) και υποδεικνύει ότι έγινε αναφορά στη διεύθυνση μνήμης του δεδομένου και όχι στο περιεχόμενο της διεύθυνσης μνήμης του. Οι δια-παραπομπές αντιστάθμισης συνήθως είναι το αποτέλεσμα *πράξεων δεικτών* (pointer operations). Για παράδειγμα, οι εντολές που αφορούν την προσπέλαση πινάκων συνήθως υλοποιούνται ως πράξεις δεικτών που

προσθέτουν ένα offset στην αρχική διεύθυνση μνήμης του πίνακα. Συνήθως, οι συμβολοσειρές δηλώνονται σε πολλά προγράμματα ως καθολικοί πίνακες σε γλώσσες όπως C/C++, οπότε μπορούμε προσεγγιστικά να λάβουμε τις αναφορές που έγιναν σε συμβολοσειρές με αποδοτικό τρόπο σε σχέση με εξαγωγή συμβολοσειρών που αποτελεί χρονοβόρα διαδικασία. Σε αντίθεση με τις δια-παραπομπές εγγραφής και ανάγνωσης, που πηγάζουν μόνο από διευθύνσεις εντολών, οι δια-παραπομπές αντιστάθμισης μπορούν να προέλθουν ακόμα και από το τμήμα .data ως πίνακας δεικτών που καταλήγει στην παραγωγή μιας δια-παραπομπής αντιστάθμισης από κάθε τοποθεσία του πίνακα προς την τοποθεσία που "δείχνει" η κάθε θέση του πίνακα. Η διεύθυνση της καθολική μεταβλητής *ref_it* στο Σχήμα 3.6 αναφέρεται από τη διεύθυνση *_main+4* και το επίθεμα *"o"* δηλώνει ότι πρόκειται για δια-παραπομπή αντιστάθμισης.

```
.data:0040B720 read_it      dd ?                    ; DATA XREF: _main+E↑r
.data:0040B720                                      ; _main+25↑r
.data:0040B724 write_it     dd ?                    ; DATA XREF: _main+1B↑w
.data:0040B724                                      ; _main+2E↑w ...
.data:0040B728 ref_it       db    ? ;               ; DATA XREF: _main+4↑o
```

Σχήμα 3.6: Παράδειγμα δια-παραπομπών δεδομένων IDA Pro

Δια-παραπομπές Εντολών

Μια *δια-παραπομπή εντολής* (code cross-reference) υποδεικνύει ότι η εντολή μεταφέρει ή ενδέχεται να μεταφέρει την ροή του προγράμματος σε κάποια άλλη εντολή. Συγκεκριμένα, υπάρχουν τρεις βασικοί τρόποι ροής. Ο πρώτος τρόπος ονομάζεται *συνήθης ροή* (ordinary flow) και αφορά την διαδοχική μεταφορά ροής από την μία εντολή στην εντολή που βρίσκεται στην επόμενη θέση μνήμης. Δεδομένου ότι οι περισσότερες εντολές ακολουθούν αυτό το μοτίβο είναι δύσκολη η εξαγωγή χαρακτηριστικών από το πλήθος ή ποσοστό των συνήθης ροών και το IDA Pro δεν παρέχει κάποιου είδους μεταδεδομένα για αυτόν τον τύπο [Eagl08].

Οι εντολές που αφορούν τις κλήσεις συναρτήσεων (όπως η εντολή *call* της αρχιτεκτονικής x86) προκαλούν μια *ροή κλήσης* (call flow). Η ροή κλήσης υποδεικνύει ότι η ροή μεταφέρεται στην συνάρτηση που καλείται. Σε κάποιες περιπτώσεις μια κλήση συνάρτησης μπορεί να είναι συνήθης ροή εάν η διεύθυνση της συνάρτησης είναι η ακριβώς επόμενη εντολή. Οι ροές κλήσης αναπαρίστανται στο IDA Pro ως δια-παραπομπές κώδικα στην συνάρτηση για την οποία γίνεται η κλήση. Για παράδειγμα στο Σχήμα 3.7 φαίνεται ότι η συνάρτηση με όνομα *"callflow"* καλείται από τις διευθύνσεις *_main+20*, *_main:loc_401054* και η δια-παραπομπή είναι ροή κλήσης λόγω του επιθέματος *"p"*. Στο Σχήμα 3.8 φαίνεται η εντολή κλήσης που βρίσκεται στη διεύθυνση μνήμης με ετικέτα *loc_401054*.

```
.text:00401000 callflow     proc near               ; CODE XREF: _main+20↓p
.text:00401000                                      ; _main:loc_401054↓p
.text:00401000              push    ebp
.text:00401001              mov     ebp, esp
.text:00401003              pop     ebp
.text:00401004              retn
.text:00401004 callflow     endp
```

Σχήμα 3.7: Παράδειγμα δια-παραπομπής ροής κλήσης IDA Pro

Η *ροή άλματος* (jump flow) αφορά τις εντολές ελέγχου βρόγχων με ή χωρίς συνθήκη. Τα άλματα υπό συνθήκη που εν τέλει δεν ικανοποιείται η συνθήκη θεωρούνται δια-παραπομπές

συνήθης ροής διότι εν τέλει θα εκτελεστεί η επόμενη εντολή. Αντιθέτως, τα άλματα δίχως συνθήκη εκτελούνται πάντα επομένως δεν μπορούν θα θεωρηθούν συνήθης ροές. Όπως και οι κλήσεις ροής, έτσι και τα άλματα ροής αναπαρίστανται στο IDA Pro ως δια-παραπομπές κώδικα στην διεύθυνση που θα γίνει το άλμα. Για παράδειγμα, στο Σχήμα 3.8 η διεύθυνση με ετικέτα *loc_40104A* αναφέρεται στην εντολή άλματος *jnz short loc_40104A* της διεύθυνσης _main+2C και η δια-παραπομπή υποδηλώνεται ως ροή άλματος λόγω του επιθέματος *”j”*.

```
.text:00401010 _main          proc near
.text:00401010
.text:00401010 p              = dword ptr -4
.text:00401010
.text:00401010                push    ebp
.text:00401011                mov     ebp, esp
.text:00401013                push    ecx
.text:00401014                mov     [ebp+p], offset ref_it
.text:0040101B                mov     eax, [ebp+p]
.text:0040101E                mov     ecx, read_it
.text:00401024                mov     [eax], ecx
.text:00401026                mov     edx, [ebp+p]
.text:00401029                mov     eax, [edx]
.text:0040102B                mov     write_it, eax
.text:00401030                call    callflow
.text:00401035                cmp     read_it, 3
.text:0040103C                jnz     short loc_40104A
.text:0040103E                mov     write_it, 2
.text:00401048                jmp     short loc_401054
.text:0040104A ; ---------------------------------------------------------
.text:0040104A
.text:0040104A loc_40104A:                            ; CODE XREF: _main+2C↑j
.text:0040104A                mov     write_it, 1
.text:00401054
.text:00401054 loc_401054:                            ; CODE XREF: _main+38↑j
.text:00401054                call    callflow
.text:00401059                xor     eax, eax
```

Σχήμα 3.8: Παράδειγμα δια-παραπομπής άλματος ροής IDA Pro

Εξαγωγή Χαρακτηριστικών

Παρακάτω παρουσιάζονται τα χαρακτηριστικά που εξήχθησαν από τα μεταδεδομένα του IDA Pro που αφορούσαν τις δια-παραπομπές δεδομένων και εντολών.

1. **Πλήθος δια-παραπομπών κώδικα**: Το συνολικό πλήθος των αναφορών που έγιναν σε θέσεις μνήμης που περιείχαν εντολές.

2. **Ποσοστό ροών κλήσεων επί των δια-παραπομπών κώδικα**: Το συγκεκριμένο χαρακτηριστικό προκύπτει ως το σύνολο των εντολών στις οποίες έγινε αναφορά στη θέση μνήμη τους μέσω κλήσης συνάρτησης, προς το συνολικό αριθμό των δια-παραπομπών που έγιναν σε εντολές.

3. **Ποσοστό ροών αλμάτων επί των δια-παραπομπών κώδικα**: Το σύνολο των εντολών στις οποίες έγιναν αναφορές μέσω αλμάτων και όχι συναρτήσεων (εντολών *jmp*), προς το συνολικό αριθμό των δια-παραπομπών που έγιναν σε εντολές.

4. **Ποσοστό δια-παραπομπών εντολών από χαμηλότερες \ υψηλότερες διευθύνσεις**:

Το πλήθος των θέσεων μνήμης εντολών που αναφέρθηκαν από εντολές που βρίσκονται σε χαμηλότερες \υψηλότερες διευθύνσεις μνήμης προς το συνολικό πλήθος αναφορών σε θέσεις μνήμης εντολών.

5. **Πλήθος δια-παραπομπών δεδομένων**: Το συνολικό πλήθος των αναφορών που έγιναν σε θέσεις μνήμης που περιείχαν δεδομένα.

6. **Ποσοστό δεδομένων προς ανάγνωση**: Το πλήθος των δεδομένων που αναφέρθηκαν οι θέσεις μνήμης τους προκειμένου να αναγνωστούν προς το συνολικό πλήθος των δια-παραπομπών δεδομένων.

7. **Ποσοστό δεδομένων προς εγγραφή**: Το πλήθος των δεδομένων που αναφέρθηκαν οι θέσεις μνήμης τους προκειμένου να εγγραφούν προς το συνολικό πλήθος των δια-παραπομπών δεδομένων.

8. **Ποσοστό δια-παραπομπών δεδομένων από χαμηλότερες \υψηλότερες διευθύνσεις**: Το πλήθος των θέσεων μνήμης δεδομένων που αναφέρθηκαν από τοποθεσίες που βρίσκονται σε χαμηλότερες \υψηλότερες διευθύνσεις μνήμης προς το συνολικό πλήθος αναφορών σε θέσεις μνήμης δεδομένων.

3.3 Επιλογή Χαρακτηριστικών

Παρόλο που τα χαρακτηριστικά που προτάθηκαν στην προηγούμενη υποενότητα προέκυψαν έπειτα από εκτενή μελέτη και αιτιολογημένη χρήση, είναι πιθανόν ένα μέρος αυτών να μην διακρίνεται από υψηλή ικανότητα διαχωρισιμότητας. Για αυτό το λόγο είναι σημαντικό να αξιολογήσουμε τα σύνολα χαρακτηριστικών που επιλέξαμε και να συμπεριλάβουμε μόνο αυτά τα οποία βελτιώνουν την τελική επίδοση του ταξινομητή. Αρχικά, θα εφαρμόσουμε έναν αλγόριθμο πρόσθιας βηματικής επιλογής συνόλων χαρακτηριστικών ώστε να διατηρήσουμε μόνο τα σύνολα που παρουσιάζουν υψηλή απόδοση και στη συνέχεια θα αναλύσουμε το σύνολο δεδομένων σε κύριες συνιστώσες ώστε να επισπεύσουμε την εκπαίδευση του μοντέλου και να βελτιώσουμε την προβλεπτική ικανότητα του ταξινομητή.

3.3.1 Ανάλυση Κύριων Συνιστωσών

Η *ανάλυση κύριων συνιστωσών* (principal component analysis) είναι η απλούστερη και πλέον διαδεδομένη *πολυμεταβλητή ανάλυση* (multivariate analysis) που αποσκοπεί στην ανεύρεση από ένα πλήθος p μεταβλητών ορισμένων νέων ολιγάριθμων μεταβλητών οι οποίες έχουν την ιδιότητα να είναι γραμμικοί συνδυασμοί των αρχικών μεταβλητών και παράλληλα να μη συσχετίζονται μεταξύ τους. Το μεγάλο πλεονέκτημα της μεθόδου έγκειται στην ιδιαιτερότητα που διαθέτουν λόγω της ανάλυσης, να εξηγούν πολύ μεγάλο ποσοστό της ολικής μεταβλητότητας που αναπτύσσεται μεταξύ των p μεταβλητών, το οποίο τελικά κατανέμεται σε μερικές μόνο νέες μεταβλητές. Έτσι, το μεγαλύτερο μέρος της πληροφορίας που θα αντλούνταν από τις αρχικές p μεταβλητές συγκρατείται με τη δημιουργία των νέων μεταβλητών.

Η τεχνική των κύριων συνιστωσών έχει ως βάση, κατά τη διαδικασία υπολογισμού της, τη μήτρα των κατά *ζεύγη συσχετίσεων* (correlation matrix) των μεταβλητών. Κατά συνέπεια, προκειμένου η μέθοδος να θεωρηθεί επιτυχημένη, να παρέχει δηλαδή ουσιώδη πληροφόρηση, απαραίτητη προϋπόθεση είναι κάποιοι συντελεστές συσχέτισης των αρχικών μεταβλητών της μήτρας συσχετίσεων να φέρουν υψηλές τιμές θετικές ή αρνητικές. Εάν ισχύει η

προηγούμενη προϋπόθεση τότε ένα μεγάλο πλήθος χαρακτηριστικών μπορούν να αντιπροσωπευθούν επάξια από ένα αρκετά μικρότερο σύνολο κύριων συνιστωσών.

Επειδή λοιπόν η ανάλυση κύριων συνιστωσών ελαττώνει αισθητά τον αριθμό των χαρακτηριστικών τότε η εκπαίδευση του μοντέλου επιταχύνεται σημαντικά όπως και η πρόβλεψη του ταξινομητή. Ωστόσο, δεδομένου ότι η διαδικασία της ανάλυσης κύριων συνιστωσών είναι επίσης έντονη υπολογιστικά υπολογίσαμε ότι δεν παρατηρήθηκε αισθητή βελτίωση όσων αφορά την ταχύτητα εκπαίδευσης.

Ο λόγος που χρησιμοποιήσαμε την ανάλυση κύριων συνιστωσών ήταν ο πυρήνας λειτουργίας της μεθόδου που δημιουργεί συνιστώσες με τέτοιο τρόπο ώστε να μεγιστοποιείται η διακύμανση. Η διακύμανση όπως αναφέρθηκε και στο Κεφάλαιο 2 είναι άμεσα συνδεδεμένη με την εντροπία που εκφράζει ποσοτικά το πλήθος της πληροφορίας. Για πολλές κατανομές η εντροπία εξαρτάται από την διακύμανση της κατανομής γεγονός το οποίο μας ενδιαφέρει λόγω της χρησιμοποίησης δέντρων αποφάσεων ως ταξινομητές. Υπενθυμίζουμε ότι τα δέντρα απόφασης χρησιμοποιούν την εντροπία και το κέρδος πληροφορίας κατά την εκπαίδευση ώστε να σχηματίσουν την δομή τους. Συνεπώς, η ανάλυση κύριων συνιστωσών είναι κατάλληλη για εφαρμογή σε δέντρα αποφάσεων καθώς μετατρέπει το σύνολο χαρακτηριστικών με τέτοιο τρόπο ώστε να εξάγονται χαρακτηριστικά υψηλής διακύμανσης, που συνήθως είναι και αυτά που έχουν το μεγαλύτερο κέρδος πληροφορίας κατά την εκπαίδευση του δέντρου απόφασης. Αυτή η μορφή των χαρακτηριστικών βοηθάει την βελτίωση της προβλεπτικής ικανότητας του δέντρου απόφασης.

Επιπλέον, λόγω της αισθητής μείωσης του πλήθος των χαρακτηριστικών, η εκπαίδευση του μοντέλου επιταχύνεται σημαντικά όπως και η πρόβλεψη του ταξινομητή. Στα αποτελέσματα θα συγκριθεί η απόδοση των μοντέλων με και χωρίς την συγκεκριμένη τεχνική ώστε να αξιολογήσουμε εάν η κατανομή των χαρακτηριστικών του προβλήματος μας αναδεικνύει υψηλή συσχέτιση μεταξύ εντροπίας και διακύμανσης. Τέλος, κάποια χαρακτηριστικά που μπορεί να μην προσδίδουν κάποια πληροφορία διαχωρισιμότητας θα αφαιρεθούν με αποτέλεσμα να επιτυγχάνεται και μείωση θορύβου του συνόλου δεδομένων.

3.3.2 Πρόσθια Βηματική Επιλογή

Η *πρόσθια βηματική επιλογή* (forward stepwise selection) είναι η τεχνική που χρησιμοποιήσαμε προκειμένου να αξιολογήσουμε το κάθε σύνολο χαρακτηριστικών που εξάγαμε παραπάνω και να αποφασίσουμε εάν αξίζει να συμπεριληφθεί στο τελικώς εξαγόμενο σύνολο χαρακτηριστικών. Κατά τη διάρκεια της εργασίας μέχρι την τελική επιλογή των συνόλων χαρακτηριστικών που χρησιμοποιήθηκαν δοκιμάστηκαν διαφορά χαρακτηριστικά τα οποία όμως έπειτα από αξιολόγηση αποφασίσαμε να παραληφθούν από το τελικό σύνολο δεδομένων. Επιπλέον, μέσω αυτής της μεθόδου εξάγαμε συμπεράσματα και για την ικανότητα διαχωρισιμότητας του κάθε συνόλου χαρακτηριστικών.

Η μέθοδος που ακολουθήσαμε πρόκειται για έναν *άπληστο αλγόριθμο* (greedy algorithm) ο οποίος ξεκινάει αρχικά με το κενό σύνολο χαρακτηριστικών. Στη συνέχεια σε κάθε βήμα επιλέγει ποιο σύνολο χαρακτηριστικών από αυτά που αναλύθηκαν στην προηγούμενη ενότητα αξίζει περισσότερο να προστεθεί στο μοντέλο. Ο αλγόριθμος σταματάει όταν η προσθήκη οποιουδήποτε διαθέσιμου μη-επιλεγμένου ως τότε συνόλου χαρακτηριστικών δεν βελτιώνει την επίδοση του μοντέλου ή όταν όλα τα σύνολα χαρακτηριστικών προστεθούν. Η λειτουργία του αλγορίθμου παρουσιάζεται υπό την μορφή ψευδοκώδικα στον (Αλγόριθμο 1)

Η άπληστη επιλογή του αλγορίθμου βασίζεται στο γεγονός ότι κάθε φορά επιλέγει το σύνολο χαρακτηριστικών που θα μεγιστοποιήσει την απόδοση του μοντέλου στο τρέχον βήμα

Algorithm 1 Πρόσθια βηματική επιλογή συνόλων χαρακτηριστικών

```
procedure forward_stepwise_select(featsets)
    ▷ featsets: σύνολο που εμπεριέχει όλα τα σύνολα χαρακτηριστικών προς εξέταση
    best_featsets ← {}
    while (new_score > best_score)and(featsets.not_empty()) do
        new_featset = None
        for each x in featsets do
            best_featsets.append(x)
            temp ← featset_eval_score(best_featsets)
            best_featsets.pop(x)
            if temp ≥ new_score then
                new_score = temp
                new_featset = x
        if new_featset ≠ None then
            best_featsets.append(new_featset)
            featsets.remove(new_featset)
```

και δεν ασχολείται με την συνολική μεγιστοποίηση της επίδοσης. Επομένως, ο αλγόριθμος που έχουμε επιλέξει δεν εγγυάται την εύρεση της βέλτιστης επιλογής συνόλων χαρακτηριστικών. Όμως, η χρησιμοποίηση του αποτελεί έναν συμβιβασμό που χαρακτηρίζεται από ικανοποιητική πολυπλοκότητα $\mathcal{O}(n^2)$ ως προς το πλήθος των συνόλων χαρακτηριστικών προς εξέταση. Η εναλλακτική του αλγορίθμου που χρησιμοποιήσαμε είναι ένας brute force αλγόριθμος, όπου η επίδοση του μοντέλου θα υπολογιζόταν για κάθε δυνατό σύνολο συνόλων χαρακτηριστικών [Jame14]. Ο εξαντλητικός αλγόριθμος θα επέστρεφε εγγυημένα την βέλτιστη λύση ωστόσο θα χαρακτηριζόταν από πολυπλοκότητα ίση με τον αριθμό όλων των δυνατών υποσυνόλων των συνόλων χαρακτηριστικών, δηλαδή $\mathcal{O}(2^n)$.

Κεφάλαιο 4

Πειραματικά Αποτελέσματα

Στο παρόν Κεφάλαιο θα παρουσιαστούν τα αποτελέσματα που καταγράφηκαν από τους διάφορους ταξινομητές που χρησιμοποιήσαμε καθώς επίσης και η αξία των συνόλων χαρακτηριστικών που εξάγαμε. Έπειτα από σύντομη παρουσίαση του συνόλου δεδομένων, θα παρουσιαστούν οι χρόνοι εκπαίδευσης καθώς και ο μέσος χρόνος εξαγωγής χαρακτηριστικών και πρόβλεψης. Στο τέλος του Κεφαλαίου θα παρουσιαστεί το τελικό μοντέλο που προτείνουμε για κατηγοριοποίηση κακόβουλου λογισμικού σε πραγματικό χρόνο.

4.1 Σύνολο Δεδομένων

Σε αυτή την ενότητα θα μελετήσουμε το σύνολο δεδομένων που είχαμε στη διάθεση μας. Αρχικά, θα περιγράψουμε τη δομή του και την προέλευση του. Στη συνέχεια, θα αναλύσουμε όλες τις οικογένειες κακόβουλου λογισμικού που εμφανίζονται στο σύνολο δεδομένων και θα ολοκληρώσουμε την ενότητα με μια σύντομη στατιστική ανάλυση που θα αναδείξει την κατανομή του συνόλου καθώς και άλλα χρήσιμα χαρακτηριστικά του.

4.1.1 Περιγραφή Συνόλου Δεδομένων

Το Microsoft Malware Classification Challenge ανακοινώθηκε το 2015 και μαζί του δημοσιεύτηκε και το dataset μεγέθους περίπου 0.5 Terabytes, αποτελούμενο από ένα αρχείο συμβολικής γλώσσας και ένα δεκαεξαδικό αρχείο για κάθε δείγμα [Rone18]. Πέρα από τη χρήση του συγκεκριμένου dataset ως κριτήριο αξιολόγησης του διαγωνισμού, πλέον αποτελεί βασικό σημείο αναφοράς για την έρευνα που αφορά στην μοντελοποίηση συμπεριφοράς κακόβουλου λογισμικού. Επιπλέον, για το συγκεκριμένο dataset έχουν γίνει αναφορές σε πάνω από 50 ερευνητικές δημοσιεύσεις, γεγονός το οποίο επιδεικνύει το κύρος και την πληρότητα του. Η δημοτικότητα του μας δίνει επίσης τη δυνατότητα να συγκρίνουμε τα αποτελέσματα μας με πλήθος δημοσιεύσεων και να αξιολογήσουμε τα προτεινόμενα μας μοντέλα σε σχέση με τα ήδη δημοσιευμένα.

Το σύνολο δεδομένων περιέχει ένα σύνολο αρχείων κακόβουλου λογισμικού που αναπαριστούν μια μίξη 9 διαφορετικών οικογενειών. Κάθε κακόβουλο αρχείο χαρακτηρίζεται από ένα *αναγνωριστικό* (id), μία μοναδική *τιμή κατακερματισμού* (hash value) 20 χαρακτήρων και έναν ακέραιο που αναπαριστά καθεμία από τις εννέα οικογένειες κακόβουλου λογισμικού.

Για κάθε αρχείο, παρέχεται η δεκαεξαδική αναπαράσταση του περιεχομένου του εκτελέσιμου, χωρίς την επικεφαλίδα των αντίστοιχων αρχείων Windows (PE header). Επίσης, παρέχεται ένα αρχείο *συμβολικής γλώσσας* (assembly), μαζί με *μεταδεδομένα* (metadata) που αφορούν κλήσεις συναρτήσεων, συμβολοσειρές, κλπ. Το δεύτερο αρχείο παράχθηκε από το πρόγραμμα IDA Pro.

Η έλλειψη της επικεφαλίδας ουσιαστικά αποτελεί επιπλέον πρόκληση για την κατηγοριοποίηση ενός αρχείου και το συγκεκριμένο σύνολο δεδομένων αποσκοπεί στο να ωθήσει τους ερευνητές να επινοήσουν ακόμα πιο ανθεκτικές τεχνικές κατηγοριοποίησης που δεν περιορίζονται στην εξαγωγή δεδομένων από την επικεφαλίδα αλλά προσπελάζουν τα αρχεία για εξαγωγή σημασιολογικών χαρακτηριστικών.

4.1.2 Κατηγορίες Κακόβουλου Λογισμικού

Το σύνολο δεδομένων που χρησιμοποιήσαμε εμπεριέχει 9 διαφορετικές οικογένειες κακόβουλου λογισμικού. Οι επιμέρους κατηγορίες διαφέρουν ως προς τον τρόπο με τον οποίο διαδίδονται ή εξαπλώνονται, και τον τύπο ενεργειών ή βλαπτικών φορτίων που χρησιμοποιούν. Παρακάτω παρουσιάζονται συνοπτικά οι οικογένειες κακόβουλου λογισμικού με τις οποίες θα ασχοληθούμε στην παρούσα εργασία:

1. **Ramnit**: Αποσκοπεί στην κλοπή ευαίσθητων πληροφοριών, όπως τραπεζικών λογαριασμών και κωδικών εισόδου σε διάφορους ιστότοπους. Επιπλέον, μπορεί να δώσει απομακρυσμένη πρόσβαση στις λειτουργίες του υπολογιστή του θύματος. Τα κακόβουλα εκτελέσιμα αυτής της κατηγορίας είναι ικανά να μολύνουν εκτελέσιμα αρχεία EXE, αρχεία βιβλιοθηκών DDL και αρχεία διαδικτύου HTML. Ο σκοπός της συγκεκριμένης κατηγορίας έχει ως αποτέλεσμα να παρατηρείται συχνή χρήση προγραμματιστικών διεπαφών που αφορούν αιτήματα διαδικτύου HTTP.

2. **Lollipop**: Ανήκει στην κατηγορία κακόβουλου λογισμικού adware. Το adware αποσκοπεί στην εμφάνιση επιχορηγούμενων και παραπλανητικών διαφημίσεων. Όταν καταφέρει να διεισδύσει και να εγκατασταθεί στο σύστημα στόχο τότε αρχίζει να εμφανίζει προωθητικό περιεχόμενο σε μορφή αναδυόμενων διαφημίσεων ή διαφημίσεων κειμένου. Η οικογένεια Lollipop πέρα από τις βασικές λειτουργίες του adware, μπορεί να ανακατευθύνει τα αποτελέσματα της μηχανής αναζήτησης του χρήστη, να παρακολουθεί τις ενέργειες του και να στέλνει πληροφορίες του στον επιτιθέμενο.

3. **Kelihos_ver3**: Αποτελεί την τρίτη έκδοση της οικογένειας Kelihos και είναι ένα botnet *ομότιμων κόμβων* (peer-to-peer) με σκοπό την κλοπή ή εξόρυξη κρυπτονομισμάτων (cryptomining). Το botnet είναι ένα δίκτυο υπολογιστών που ελέγχεται απομακρυσμένα από τον επιτιθέμενο (botmaster) και πέρα από την πρόσβαση στα προσωπικά δεδομένα των χρηστών δίνει στον botmaster την δυνατότητα να εκτελέσει επιθέσεις χρησιμοποιώντας τα θύματα ως διακομιστές μεσολάβησης. Όταν το botnet αποτελείται από μεγάλο πλήθος υπολογιστών τότε ο δράστης αποκτά σημαντική υπολογιστική ισχύ, την οποία εκμεταλλεύεται η συγκεκριμένη οικογένεια κακόβουλου λογισμικού προκειμένου να αυξήσει τις πιθανότητες ο επιτιθέμενος να λύσει την *απόδειξη εργασίας* (proof-of-work) ώστε να εξορύξει κρυπτονομίσματα. Συγκεκριμένα, οι ιοί εξόρυξης κρυπτονομισμάτων ακολουθούν συγκεκριμένα μοτίβα και χαρακτηριστικά όπως η εντροπία που βοηθούν σημαντικά την ανίχνευση τους [Past19].

4. **Vundo**: Αυτή η κατηγορία κακόβουλου λογισμικού ανήκει στο ευρύτερο είδος των ιών δούρειων ίππων. Συνήθως κατεβάζει μολυσμένα αρχεία και εμφανίζει *αναδυόμενες διαφημίσεις* (pop-up advertisments). Ο τρόπος εξάπλωσης του είναι μέσω spam e-mail ή διαμοιρασμό αρχείων σε δίκτυο ομότιμων κόμβων. Οι διαφημίσεις που εμφανίζονται αποσκοπούν στο να παρακινήσουν τον χρήστη να επιλέξει κάποια από αυτές και να ανακατευθυνθεί σε κακόβουλους ιστότοπους που συνήθως παρακινούν τον

χρήστη να πληρώσει ένα αντίτιμο για να καθαρίσει τον υπολογιστή του από υποτιθέμενα μολυσμένα αρχεία. Κάποιες παραλλαγές αυτής της κατηγορίας επηρεάζουν και τις ρυθμίσεις ασφάλειας του υπολογιστή του θύματος προκειμένου να απενεργοποιήσουν τυχών αντιικά προγράμματα.

5. **Simda**: Ανήκει στην οικογένεια των ιών *κερκόπορτας* (backdoor) και λογισμικό αυτής της κατηγορίας είναι ικανό να υποκλέψει διαπιστευτήρια χρήστη, κωδικούς και πιστοποιητικά. Για την υποκλοπή των παραπάνω στοιχείων χρησιμοποιεί *καταγραφή πληκτρολόγησης* (keylogging) και ρουτίνες HTML injection. Κάποιες παραλλαγές αυτής της κατηγορίας εισάγουν *σενάρια* (scripts) σε ιστοσελίδες, επηρεάζουν την λειτουργία του συστήματος διαγράφοντας κρίσιμα *κλειδιά μητρώου* (registry keys) του λειτουργικού συστήματος των Windows και επιτρέπουν στον επιτηθέμενο να αποκτήσει απομακρυσμένη πρόσβαση.

6. **Tracur**: Τα κακόβουλα εκτελέσιμα αυτού του είδους είναι δούρειοι ίπποι οι οποίοι ανακατευθύνουν τις αναζητήσεις διαδικτύου του θύματος. Το κίνητρο των συντακτών τέτοιου λογισμικού είναι το χρηματικό όφελος μέσω παροχών αύξησης επισκεψιμότητας σε ιστότοπους. Ωστόσο, σε κάποιες παραλλαγές η ανακατεύθυνση οδηγεί τα θύματα σε κακόβουλους ιστότοπους που αποσκοπούν στη μόλυνση του υπολογιστή του θύματος.

7. **Kelihos_ver1**: Η πρώτη έκδοση της οικογένειας Kelihos που αναφέρθηκε προηγουμένως αποσκοπούσε κυρίως σε spamming. Το spamming είναι η αξιοποίηση συστημάτων μηνυμάτων για την αποστολή ανεπιθύμητων και αυτόκλητων μηνυμάτων κυρίως για σκοπούς διαφήμισης. Επιπλέον, κάποια δείγματα της πρώτης έκδοσης του Kelihos εκμεταλλεύονταν το μέγεθος του botnet για την εκτέλεση *κατανεμημένων επιθέσεων άρνησης υπηρεσίας* (distributed denial-of-service attacks) [Kerk14].

8. **Obfuscator.ACY**: Αυτή η κατηγορία εμπεριέχει κακόβουλα λογισμικά στα οποία έχει εφαρμοστεί η τεχνική της συσκότισης κώδικα προκειμένου να αποφευχθεί η ανίχνευση από αντιικά προγράμματα. Το είδος του κακόβουλου λογισμικού που βρίσκεται πίσω από τον συσκοτισμένο κώδικα μπορεί να ανήκει σε οποιαδήποτε κατηγορία. Στο Κεφάλαιο 2 της εργασίας έχουν αναλυθεί οι διάφοροι τρόποι που εφαρμόζονται ώστε να επιτευχθεί η συσκότιση. Επιπλέον, στην ανάλυση της υλοποίησης επισημάναμε ότι χαρακτηριστικά όπως η εντροπία, τα τμήματα πακεταρίσματος και ασυνήθιστη κατανομή σε χρήση καταχωρητών προδίδει ότι πιθανότητα έχουν εφαρμοστεί τεχνικές συσκότισης.

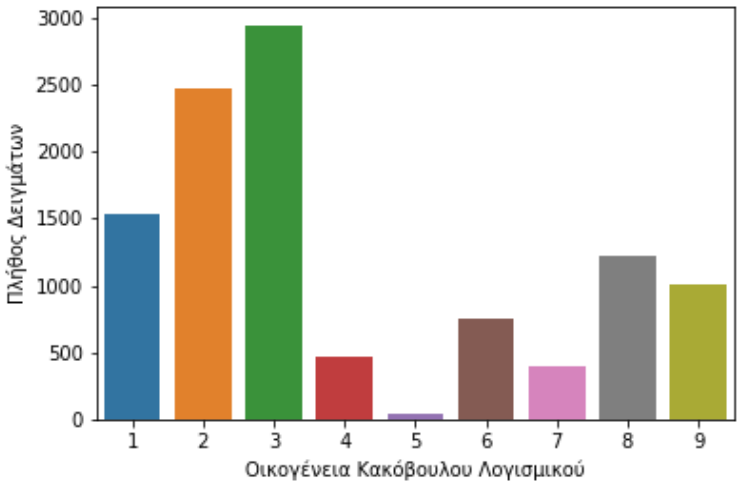
9. **Gatak**: Ανήκει στην ευρύτερη κατηγορία των δούρειων ίππων και βασική του λειτουργία είναι η συγκέντρωση πληροφοριών του θύματος με τη μετέπειτα αποστολή τους στον επιτιθέμενο. Ο τρόπος εξάπλωσης του είναι κυρίως μέσω τροποποίησης του ιού ώστε να παρουσιάζεται στη μορφή ενημερώσεων έγκυρων εφαρμογών. Μια ιδιαιτερότητα αυτής της οικογένειας κακόβουλου λογισμικού είναι η χρήση *στεγανογραφίας* (steganography) κατά το κατέβασμα κακόβουλων αρχείων προκειμένου να αποφευχθεί η ανίχνευση από διάφορους μηχανισμούς άμυνας.

4.1.3 Στατιστική Ανάλυση

Προκειμένου να εφαρμόσουμε μοντέλα μηχανικής μάθησης για την ανίχνευση κακόβουλου λογισμικού, πρέπει πρώτα να μελετήσουμε και να αναλύσουμε επαρκώς το σύνολο δεδομένων στο οποίο θα βασίσουμε την εκπαίδευση των μοντέλων μας. Οι στατιστικές ιδιότητες του συνόλου παίζουν σημαντικό ρόλο στην επιλογή του μοντέλου καθώς και στις υπερπαραμέτρους που θα επιλεγούν.

Κατανομή Κλάσεων

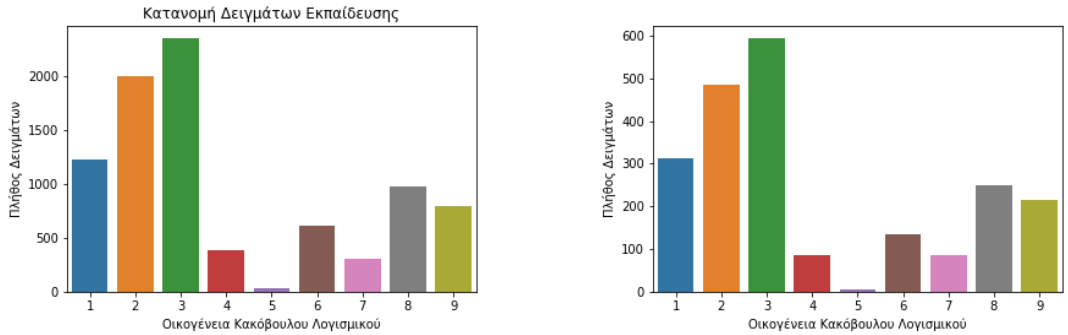
Ένα ιδιαίτερα σημαντικό χαρακτηριστικό ενός συνόλου δεδομένων επιβλεπόμενης μάθησης είναι η κατανομή των κλάσεων στα δείγματα. Στο Σχήμα 4.1 παρουσιάζεται η σχηματική απεικόνιση, σε μορφή ιστογράμματος, της κατανομής των οικογενειών κακόβουλου λογισμικού.



Σχήμα 4.1: Κατανομή Κλάσεων Συνόλου Δεδομένων

Με βάση την παραπάνω κατανομή παρατηρείται ότι υπάρχει μεγάλη ανισοκατανομή στις κλάσεις μεταξύ των δειγμάτων γεγονός το οποίο αποτελεί ακόμα μεγαλύτερη πρόκληση για την υψηλή απόδοση του μοντέλου. Επιπλέον, η οικογένεια κακόβουλου λογισμικού Simda έχει αρκετά μικρό αριθμό δειγμάτων, γεγονός το οποίο δυσκολεύει την επαρκή εκπαίδευση του μοντέλου για αναγνώριση της συγκεκριμένης κλάσης. Προκειμένου να αντιμετωπιστεί το συγκεκριμένο πρόβλημα θα θέτουμε υψηλή ποινή στο μοντέλο για προβλέψεις $\Psi\Theta_5$ και ΨA_5. Αυτή η στρατηγική ποινικοποίησης λανθασμένων αποφάσεων σε υποεκπροσωπούμενες κλάσεις εμφανίζεται συχνά στη βιβλιογραφία για αντιμετώπιση μεγάλων ανισοκατανομών στα σύνολα δεδομένων [Fern18].

Επιπλέον, πέρα από την αρχική κατανομή του συνόλου δεδομένων, όπως αναφέραμε και προηγουμένως ο διαχωρισμός του αρχικού συνόλου σε δείγματα εκπαίδευσης και δείγματα επικύρωσης είναι σημαντικός παράγοντας ώστε το μοντέλο να εκπαιδευτεί σωστά. Έτσι λοιπόν, επιλέξαμε τα δείγματα εκπαίδευσης και επικύρωσης να ακολουθούν την ίδια κατανομή με το αρχικό σύνολο δεδομένων. Αυτός ο τρόπος διαχωρισμού των δεδομένων ονομάζεται διαστρωματοποίηση. Τα δείγματα εκπαίδευσης αποτέλεσαν το 80% του αρχικού συνόλου ενώ τα δείγματα επικύρωσης το υπόλοιπο 20%. Παρακάτω παρουσιάζονται οι κατανομές των δειγμάτων εκπαίδευσης και επικύρωσης που είναι σε πλήρη αντιστοιχία με το αρχικό σύνολο δεδομένων.



(a) Εκπαίδευσης **(b)** Επικύρωσης

Σχήμα 4.2: Κατανομές δειγμάτων

4.2 Εξαγωγή Χαρακτηριστικών

Η εξαγωγή χαρακτηριστικών όπως έχουμε ήδη αναφέρει αποτελεί ίσως το σημαντικότερο κομμάτι της εργασίας και ο απώτερος σκοπός μας είναι η αποδοτική εξαγωγή τους ώστε το σύστημα μας να μπορεί να εφαρμοστεί σε κατηγοριοποίηση πραγματικού χρόνου. Συνεπώς, πρέπει να είμαστε ιδιαίτερα προσεκτικοί να επιλέξουμε χαρακτηριστικά που δεν θα αυξήσουν σημαντικά τις υπολογιστικές απαιτήσεις του συστήματος. Στην Ενότητα 3.2 παρουσιάσαμε όλα τα σύνολα χαρακτηριστικών που αποφασίσαμε να συμπεριλάβουμε στο σύστημα μας. Σε αυτή την Ενότητα θα αποδείξουμε ότι η επιλογή τους είναι σύμφωνη με τις απαιτήσεις απόδοσης του συστήματος μας.

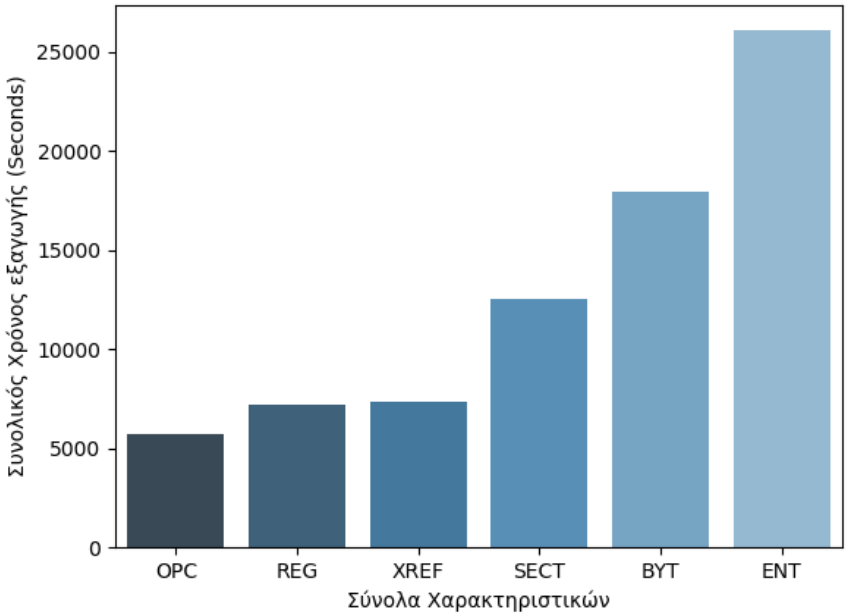
Λόγω του ότι αυτή η Ενότητα αφορά την παρουσίαση χρόνων εκτέλεσης είναι σημαντικό να αναφερθούν πληροφορίες για τα μηχανήματα στα οποία πραγματοποιήθηκε η εξαγωγή των συνόλων χαρακτηριστικών. Για τα πειράματα μας χρησιμοποιήθηκε φορητός υπολογιστής με τετραπύρηνο επεξεργαστή της Intel i7-3740QM και 8 GB μνήμη RAM DDR3. Τονίζεται ότι δεν εστιάσαμε στο να μειώσουμε τον χρόνο όσο το δυνατόν περισσότερο αλλά στο να αποδείξουμε ότι ο χρόνος είναι αρκετά μικρός για τις απαιτήσεις μας. Εκμεταλλευτήκαμε μόνο τον έναν επεξεργαστή καθώς δεν παραλληλοποιήσαμε τη διαδικασία και εξάγαμε ένα μέρος των χαρακτηριστικών με βιβλιοθήκες ταιριάσματος προτύπων της Python που είναι σημαντικά πιο αργή από τις εντολές επεξεργασίας κειμένου του λειτουργικού συστήματος Unix.

Προκειμένου να διευκολύνουμε την παρουσίαση των γραφημάτων και των αποτελεσμάτων παραθέτουμε στον Πίνακα 4.1 συμβολικά ονόματα σε καθένα από τα σύνολα με τα οποία ασχοληθήκαμε.

Συμβολικό Όνομα	Σύνολο Χαρακτηριστικών
OPC	Συχνότητα εμφάνισης επιλεγμένων εντολών (3.2.3)
REG	Συχνότητα εμφάνισης καταχωρητών (3.2.1)
XREF	Μεταδεδομένα Δια-παραπομπών (3.2.6)
SECT	Στοιχεία τμημάτων (3.2.2)
BYT	Συχνότητα εμφάνισης byte (3.2.4)
ENT	Εντροπία δεκαεξαδικού αρχείου (3.2.5)

Πίνακας 4.1: Συμβολισμοί συνόλων χαρακτηριστικών

Στον Πίνακα 4.2 παρουσιάζονται για κάθε ομάδα χαρακτηριστικών της Ενότητας 3.2 ο μέσος χρόνος εξαγωγής των χαρακτηριστικών του συνόλου, η απόκλιση του χρόνου εξαγωγής δείγματος, ο συνολικός χρόνος που απαιτήθηκε για να εξαχθούν τα χαρακτηριστικά από όλα τα δείγματα του συνόλου δεδομένων και το 99-στο εκατοστημόριο. Σημειώνεται ότι ο μέσος χρόνος εξαγωγής χαρακτηριστικών για το κάθε αρχείο δεν προκύπτει ως το άθροισμα των τιμών του Πίνακα 4.2 καθώς στην καταγραφή των αποτελεσμάτων αυτών διασχίζαμε τα αρχεία για κάθε σύνολο χαρακτηριστικών ξεχωριστά. Επιπλέον, στο Σχήμα 4.3 δίνεται μια γραφική απεικόνιση του χρόνου εξαγωγής των χαρακτηριστικών της κάθε ομάδας για ολόκληρο το σύνολο δεδομένων.



Σχήμα 4.3: Συνολικός χρόνος εξαγωγής ομάδων χαρακτηριστικών

Η εντροπία αποτέλεσε ένα από τα πιο απαιτητικά χαρακτηριστικά διότι η τιμή παραθύρου που επιλέξαμε απαιτούσε πολλούς υπολογισμούς. Παρόλο που συνολικά η εντροπία δεν βελτίωνε αισθητά την απόδοση του ταξινομητή, έπαιξε σημαντικό ρόλο στην αναγνώριση κρυπτογραφημένων και συσκοτισμένων αρχείων.

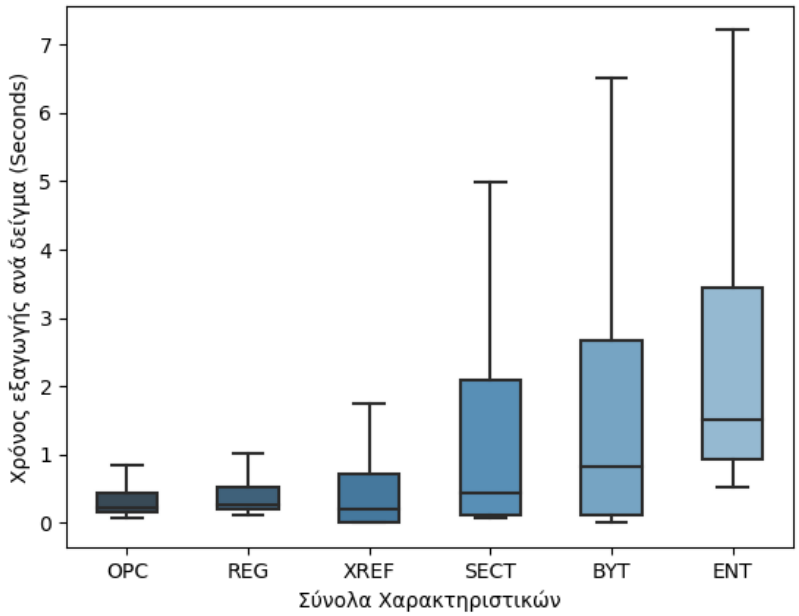
Ο χρόνος εξαγωγής των χαρακτηριστικών αυξάνει γραμμικά με το μέγεθος του αρχείου και συγκεκριμένα αυξάνει γραμμικά με το μέγεθος των γραμμών. Το μεγαλύτερο μέρος του χρόνου εξαγωγής οφείλεται στη διάσχιση του αρχείου και συνεπώς όσο μεγαλύτερο είναι

Σύνολο	Μέσος Όρος	Απόκλιση	99-Εκατοστημόριο	Συνολικός Χρόνος
OPC	0.53	0.76	4.72	5719
REG	0.67	1.51	5.99	7237
XREF	0.68	1.05	4.31	7361
SECT	1.16	1.74	4.20	12577
BYT	1.65	4.01	9.02	17940
ENT	2.40	4.03	9.84	26096

Πίνακας 4.2: Στατιστικά στοιχεία χρόνου εξαγωγής χαρακτηριστικών

τόσο προστίθεται επιπλέον φόρτος στη διαδικασία κατηγοριοποίησης του.

Στο Σχήμα 4.4 παρουσιάζεται μια πληρέστερη στατιστική παρουσίαση των χρόνων εξαγωγής χαρακτηριστικών μέσω ενός θηκογράμματος. Το *θηκόγραμμα* (boxplot) είναι γνωστό και ως το διάγραμμα των πέντε αριθμών. Πρόκειται για ένα ορθογώνιο με δύο *κεραίες* (whiskers) το οποίο κατασκευάζεται ως εξής: η κάτω βάση του ορθογωνίου βρίσκεται στην ελάχιστη παρατήρηση, η πάνω βάση βρίσκεται στη μέγιστη παρατήρηση, η διάμεσος αναπαριστάνεται με ένα οριζόντιο ευθύγραμμο τμήμα μέσα στο ορθογώνιο και το ύψος του ορθογωνίου αναπαριστά το εύρος των παρατηρήσεων που ανήκουν στο 50% εκατέρωθεν του μέσου. Γενικά, το θηκόγραμμα έχει χρησιμοποιηθεί ευρέως στην βιβλιογραφία για παρουσίαση χρόνων εκτέλεσης καθώς διευκολύνει τη διερευνητική ανάλυση δεδομένων.



Σχήμα 4.4: Θηκόγραμμα χρόνου εξαγωγής ομάδων χαρακτηριστικών ανά δείγμα

4.3 Μετρικές Αξιολόγησης

Στο πρόβλημα της ταξινόμησης επιβλεπόμενης μάθησης όπως αναφέραμε και στην θεωρία στόχος του ταξινομητή είναι η πρόβλεψη της κλάσης που ανήκει ένα δείγμα με βάση ένα σύνολο χαρακτηριστικών που δίνεται ως είσοδος. Για να αξιολογήσουμε την απόδοση του ταξινομητή πρέπει συνεπώς να αξιολογήσουμε τις προβλέψεις του. Αρχικά θα ορίσουμε κάποιες θεμελιώδεις μονάδες πρόβλεψης στις οποίες βασίζονται σχεδόν όλες οι μετρικές που θα χρησιμοποιήσουμε. Συγκεκριμένα, σε κάθε πρόβλεψη υπάρχουν τα παρακάτω ενδεχόμενα:

- **Αληθώς Θετικό ($\mathbf{A\Theta}_i$)**: Μία πρόβλεψη χαρακτηρίζεται αληθώς θετική για την κατηγορία i όταν ο ταξινομητής προβλέπει ότι το δείγμα ανήκει στην κατηγορία i και η πρόβλεψη του είναι σωστή.

- **Αληθώς Αρνητικό ($\mathbf{AA}_i$)**: Μία πρόβλεψη χαρακτηρίζεται αληθώς αρνητική για την κατηγορία i όταν ο ταξινομητής έχει προβλέψει ότι το δείγμα ανήκει σε μια κατηγορία διαφορετική της κλάσης i και η πρόβλεψη είναι σωστή.

- **Ψευδώς Θετικό ($\mathbf{\Psi\Theta}_i$)**: Μια πρόβλεψη είναι ψευδώς θετική για την κατηγορία i όταν ο ταξινομητής προβλέπει ότι το δείγμα ανήκει στην κλάση i αλλά το δείγμα ανήκει σε άλλη κλάση.

- **Ψευδώς Αρνητικό ($\mathbf{\Psi A}_i$)**: Μια πρόβλεψη είναι ψευδώς αρνητική για την κατηγορία i όταν το δείγμα ανήκει στην κλάση i και ο ταξινομητής προβλέπει διαφορετική κλάση.

Οι παραπάνω όροι προβλέψεων προέρχονται από την δυαδική ταξινόμηση και επεκτείνονται στην πολλαπλή ταξινόμηση μέσω της τεχνικής *ένας εναντίον όλων* (one-vs-all) όπου για κάθε κλάση θεωρούμε δυαδική ταξινόμηση (κλάση εναντίων όλων των υπόλοιπων κλάσεων).

4.3.1 Ακρίβεια

Η *ακρίβεια* (accuracy) ενός ταξινομητή είναι η απλούστερη μετρική που χρησιμοποιείται για αξιολόγηση των μοντέλων. Προκύπτει ως το ποσοστό των σωστών προβλέψεων του ταξινομητή επί του συνόλου των προβλέψεων που έγιναν. Συγκεκριμένα δίνεται από τον παρακάτω τύπο:

$$\textit{Ακρίβεια} = \frac{\textit{Αριθμός σωστών προβλέψεων}}{\textit{Αριθμός συνολικών προβλέψεων}} \tag{4.1}$$

Στην ταξινόμηση πολλαπλών κλάσεων με βάση τις τιμές των ΑΘ, ΨΘ, ΨΑ και ΑΨ που ορίσαμε προηγουμένως η ακρίβεια υπολογίζεται ως εξής:

$$\textit{Ακρίβεια} = \frac{\sum_{i=1}^{9}(A\Theta_i + A\Psi_i)}{\sum_{i=1}^{9}(A\Psi_i + A\Theta_i + \Psi A_i + \Psi\Theta_i)} \tag{4.2}$$

Ωστόσο, η απλότητα αυτής της μετρικής περιέχει και κάποιες "παγίδες" στην αξιολόγηση του μοντέλου καθώς εμπίπτει στο παράδοξο της ακρίβειας: ταξινομητές με χαμηλότερη ακρίβεια μπορούν να έχουν καλύτερη προβλεπτική ισχύ σε σχέση με ταξινομητές πολύ υψηλής ακρίβειας. Αυτό το παράδοξο έχει αιτιολογηθεί σε διάφορες εργασίες και παρατηρείται κυρίως σε περιπτώσεις όπου υπάρχει μεγάλη ανομοιομορφία στην κατανομή των δειγμάτων των κλάσεων [Vago08, Brad97]. Η απλοποιημένη εξήγηση του παράδοξου είναι ότι ακόμα και ένας ταξινομητής που προβλέπει πάντα μια συγκεκριμένη κλάση θα έχει πολύ υψηλή ακρίβεια εάν το μεγαλύτερο ποσοστό των δειγμάτων ανήκουν στην κλάση αυτή. Παρόλα αυτά όμως θα έχει πολύ χαμηλή προβλεπτική ισχύ σε ένα σύνολο δειγμάτων με διαφορετική κατανομή των κλάσεων. Σε περιπτώσεις ανομοιόμορφης κατανομής η ανάκληση και η μετρική F_1 αξιολογούν πιο ουσιαστικά την προβλεπτική ικανότητα των μοντέλων.

4.3.2 Ανάκληση

Η *ανάκληση* (recall) εκφράζει το ποσοστό των θετικών δειγμάτων που προβλέφθηκαν σωστά από τον ταξινομητή. Στην περίπτωση μας που έχουμε πολλαπλή ταξινόμηση λαμβάνουμε τον *συγκεντρωτικό μέσο όρο* (micro-average) που συναθροίζει τη συμβολή όλων των κλάσεων για να υπολογίσει την μέση ανάκληση. Η εξίσωση υπολογισμού παρουσιάζεται παρακάτω:

$$\textit{Ανάκληση} = \frac{\sum_{i=1}^{9} A\Theta_i}{\sum_{i=1}^{9} (A\Theta_i + \Psi A_i)} \tag{4.3}$$

4.3.3 Πιστότητα

Η *πιστότητα* (precision) εκφράζει το ποσοστό των σωστών θετικών προβλέψεων του ταξινομητή. Λόγω της πολλαπλής ταξινόμησης και της έλλειψης δυαδικότητας πρόβλεψης (θετική - αρνητική) λαμβάνουμε ως μέτρο τον συγκεντρωτικό μέσο όρο, όπως και στη ανάκληση, ώστε να ληφθεί υπόψη η συμβολή όλων των κλάσεων στην τελική ανάκληση. Παρακάτω παρουσιάζεται η εξίσωση υπολογισμού του συγκεντρωτικού μέσου όρου της πιστότητας:

$$\textit{Πιστότητα} = \frac{\sum_{i=1}^{9} A\Theta_i}{\sum_{i=1}^{9} (A\Theta_i + \Psi\Theta_i)} \tag{4.4}$$

4.3.4 Μετρική F_1

Η *μετρική* F_1 (F_1-score) αποτελεί ένα μέτρο το οποίο συνδυάζει ακρίβεια και ανάκληση. Συγκεκριμένα, είναι ο αρμονικός μέσος της ακρίβειας και της ανάκλησης και μαθηματικά εκφράζεται από τον παρακάτω τύπο:

$$\textit{Μετρική } F_1 = 2 \cdot \frac{\textit{Πιστότητα} \cdot \textit{Ανάκληση}}{\textit{Πιστότητα} + \textit{Ανάκληση}} \tag{4.5}$$

4.3.5 Διασταυρωμένη επικύρωση k-μερών

Η *διασταυρωμένη επικύρωση* (cross validation) είναι καθιερωμένη μέθοδος στον τομέα της εφαρμοσμένης στατιστικής και εποπτευόμενης μηχανικής μάθησης τόσο για την επιλογή του τελικού μοντέλου ενός συστήματος όσο και για την ορθή αξιολόγηση του [Koha95]. Λόγω του τρόπου λειτουργίας της επιβλεπόμενης μάθησης το μοντέλο που εκπαιδεύεται προσπαθεί να προσαρμοστεί όσο το δυνατόν καλύτερα στα δεδομένα από τα οποία εκπαιδεύεται. Επομένως, καλά αποτελέσματα μετρικών σε ένα σύνολο δεδομένων αξιολόγησης μπορεί να είναι προϊόν υπερπροσαρμογής είτε "τυχερής" επιλογής του συνόλου. Η τεχνική της διασταυρωμένης επικύρωσης αντιμετωπίζει το παραπάνω πρόβλημα προσπαθώντας να αποτιμήσει την επίδοση ενός συστήματος με τρόπο τέτοιο ώστε η απόδοση να μην εξαρτάται από τον διαχωρισμό συνόλου εκπαίδευσης και αξιολόγησης [Refa09].

Στην διασταυρωμένη επικύρωση k-μερών το σύνολο των δεδομένων εκπαίδευσης χωρίζεται αρχικά σε k ίσα μέρη τα οποία ονομάζονται *πλαίσια* (folds). Στη συνέχεια, γίνονται k επαναλήψεις εκπαίδευσης και αξιολόγησης έτσι ώστε σε κάθε επανάληψη ένα διαφορετικό πλαίσιο να χρησιμοποιείται για την αξιολόγηση και τα υπόλοιπα $k-1$ να χρησιμοποιούνται για την εκπαίδευση. Πριν τον διαχωρισμό του συνόλου δεδομένων εκπαίδευσης ακολουθείται μια διαδικασία *διαστρωματοποίησης* (stratification) ώστε τα δεδομένα να αναδιοργανωθούν με σκοπό το κάθε πλαίσιο να αποτελεί δίκαιη απεικόνιση του συνόλου δεδομένων. Η δίκαιη απεικόνιση του συνόλου δεδομένων από κάθε πλαίσιο συνήθως εξασφαλίζεται με διατήρηση της αρχικής κατανομής των κλάσεων σε καθένα από τα πλαίσια.

Στην συγκεκριμένη εργασία η επιλογή των μοντέλων καθώς και η επιλογή των υπερπαραμέτρων τους βασίστηκε στα αποτελέσματα των παραπάνω μετρικών που λήφθηκαν για διασταυρωμένη επικύρωση για $k = 5$. Τέλος, το τελικό μοντέλο που προτείνουμε θα αξιολογηθεί για λόγους πληρότητας και για διασταυρωμένη επικύρωση $k = 10$.

4.4 Αποτελέσματα

Το σημαντικότερο κομμάτι ενός συστήματος μηχανικής μάθησης είναι η αξιολόγηση της προβλεπτικής του ικανότητας. Στο κεφάλαιο αυτό θα αξιολογήσουμε τα μοντέλα που χρησιμοποιήσαμε με βάση τις μετρικές αξιολόγησης που ορίστηκαν στην Ενότητα 4.3. Επιπλέον, θα μελετήσουμε την επιρροή που είχε η προ-επεξεργασία των δεδομένων στην απόδοση του συστήματος και θα προτείνουμε ένα τελικό πλήρες μοντέλο το οποίο θα συγκρίνουμε με αντίστοιχα μοντέλα αιχμής της βιβλιογραφίας.

Η βασική πρόκληση στα μοντέλα που χρησιμοποιήθηκαν ήταν η εύρεση των υπερπαραμέτρων με τις οποίες θα εκπαιδευτούν. Στα περισσότερα μοντέλα μηχανικής μάθησης οι βέλτιστες υπερπαράμετροι βρίσκονται μέσω αναζήτησης πλέγματος από ένα εύρος τιμών για κάθε δυνατή υπερπαράμετρο. Στην εργασία ακολουθήσαμε κι εμείς την μέθοδο της αναζήτησης πλέγματος επιλέγοντας να βελτιστοποιήσουμε τις υπερπαραμέτρους του μοντέλου ως προς τη μετρική της ακρίβειας.

4.4.1 Μηχανές Διανυσμάτων Υποστήριξης

Στις μηχανές διανυσμάτων υποστήριξης λόγω της ιδιαιτερότητας της αντικειμενικής συνάρτησης που χρησιμοποιούν προ-επεξεργαστήκαμε τα δεδομένα με χρήση *τυπικής κλιμάκωσης* (standard scaling). Ειδικότερα, πολλά στοιχεία που χρησιμοποιούνται στην αντικειμενική συνάρτηση υποθέτουν ότι τα δεδομένα είναι κεντραρισμένα γύρω από το 0 και ότι

έχουν διακύμανση ίδιας τάξης μεγέθους [Gran03]. Επειδή, στο σύνολο δεδομένων μας υπάρχουν χαρακτηριστικά με μεγάλες διαφορές ως προς την διακύμανση, τα χαρακτηριστικά με μεγάλες τιμές κυριαρχούν στην αντικειμενική συνάρτηση μειώνοντας την επίδοση του ταξινομητή.

Στον Πίνακα 4.3 παρουσιάζονται τα δέκα καλύτερα σύνολα υπερπαραμέτρων που προέκυψαν μετά από εκτενή χρήση της αναζήτησης πλέγματος. Η υπεπαράμετρος γ επιλέχθηκε μικρή ώστε να μειωθεί όσο το δυνατόν περισσότερο η μεροληψία του ταξινομητή.

c	kernel	gamma	Ακρίβεια
0.1	linear	0.001	0.9762
0.1	linear	0.01	0.9761
1	linear	0.001	0.9741
1	linear	0.01	0.9743
1	rbf	0.01	0.9495
1	poly	0.01	0.9607
1	linear	10	0.9748
10	poly	0.1	0.9593
10	linear	1	0.9734
10	poly	10	0.9582

Πίνακας 4.3: Αναζήτηση πλέγματος υπεραπαραμέτρων για ΜΔΥ

Το καλύτερο μοντέλο μετά την αναζήτηση πλέγματος στο σύνολο υπερπαραμέτρων παρουσιάζεται για λόγους πληρότητας και μελλοντικής αναφοράς στον Πίνακα 4.4.

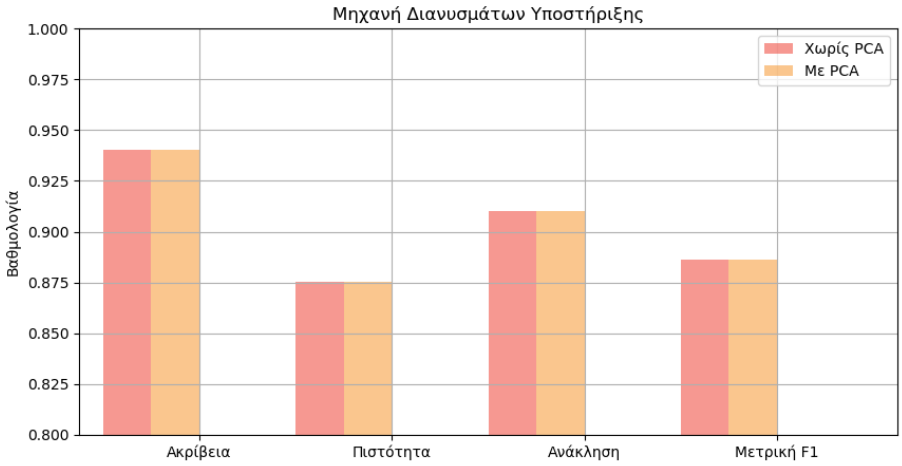
c	kernel	gamma	tol	coef0
0.1	linear	0.001	0.001	0.0

Πίνακας 4.4: Βέλτιστες υπερπαράμετροι ΜΔΥ

Για το καλύτερο μοντέλο των ΜΔΥ του Πίνακα 4.4 θα παρουσιάσουμε όλες τις μετρικές αξιολόγησης ώστε να σχηματίσουμε μια πληρέστερη εικόνα για την επίδοση του ταξινομητή. Επιπλέον, εξετάζουμε την επιρροή της ανάλυσης κύριων συνιστωσών στις μετρικές αξιολόγησης συγκρίνοντας τα αποτελέσματα του ταξινομητή με ή χωρίς χρήση της παραπάνω τεχνικής. Στο Σχήμα 4.5 απεικονίζεται η σύγκριση του ταξινομητή ΜΔΥ με ή χωρίς χρήση της παραπάνω τεχνικής και φαίνεται ότι η ανάλυση κύριων συνιστωσών δεν επιφέρει κάποια βελτίωση. Επιπλέον, στον Πίνακα 4.5 παρουσιάζονται οι ακριβείς τιμές των μετρικών αξιολόγησης. Λόγω της αύξησης του χρόνου πρόβλεψης που προσθέτει η ανάλυση κύριων συνιστωσών επιλέγουμε να μην την συμπεριλάβουμε στο σύνολο των παραμέτρων του μοντέλου μας.

	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
ΜΔΥ	0.9403	0.8756	0.9102	0.8864
ΜΔΥ με PCA	0.9261	0.8629	0.8965	0.8746

Πίνακας 4.5: Αποτελέσματα βέλτιστου μοντέλου ΜΔΥ



Σχήμα 4.5: Αξιολόγηση PCA για βέλτιστη ΜΔΥ

4.4.2 Δέντρα Αποφάσεων

Τα δέντρα αποφάσεων όπως αναφέραμε και στη θεωρία βασίζονται στη λογική κατασκευής μιας δεντρικής δομής η οποία θα ταξινομεί τα δείγματα λαμβάνοντας αποφάσεις σε κάθε κλαδί του δέντρου μέχρι να καταλήξει σε κάποιο φύλλο. Δεδομένου, ότι στοχεύσαμε στην εξαγωγή ουσιαστικών χαρακτηριστικών τότε πιθανή προ-επεξεργασία στα δεδομένα μας θα στερούσε αυτή την ιδιότητα από τα χαρακτηριστικά μας. Επιπλέον, δοκιμάσαμε και επαληθεύσαμε την θεωρητικά αναμενόμενη συμπεριφορά του ταξινομητή και παρατηρήσαμε ότι η προ-επεξεργασία μείωνε την απόδοση του αισθητά και ταυτόχρονα αύξανε τον χρόνο πρόβλεψης.

Σε αυτόν τον ταξινομητή ακολουθήσαμε την ίδια τακτική που ακολουθήσαμε και με τους υπόλοιπους προκειμένου να εξάγουμε το μοντέλο με τις υπερπαραμέτρους που βελτιστοποιούν την απόδοση του. Έπειτα λοιπόν από εφαρμογή αναζήτησης πλέγματος λάβαμε τα δέκα καλύτερα μοντέλα, τα οποία παρουσιάζονται στον Πίνακα 4.6

max_features	min_samples_split	max_depth	min_samples_leaf	accuracy
5	5	50	1	0.9471
5	5	100	1	0.9469
5	10	100	1	0.9451
10	5	100	10	0.9405
10	5	50	1	0.9481
10	5	100	1	0.9475
10	10	100	1	0.9472
10	10	50	1	0.9434
10	20	50	1	0.9402
10	20	100	1	0.9432

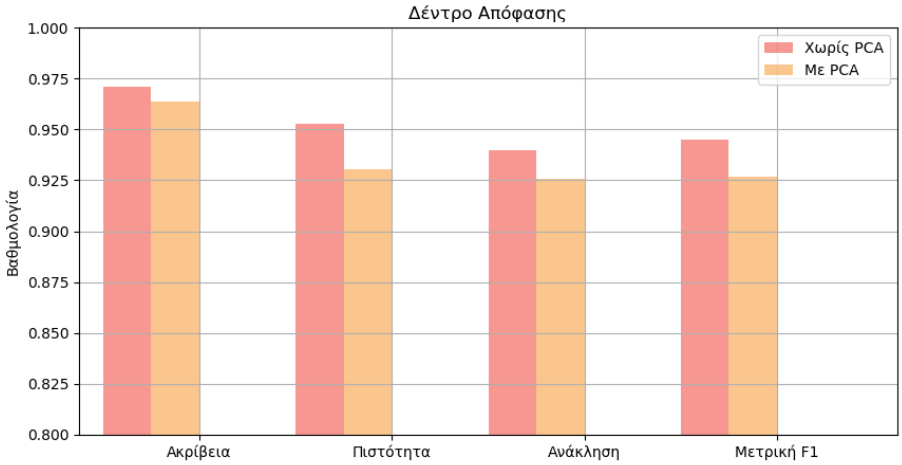
Πίνακας 4.6: Αναζήτηση πλέγματος υπεραπαραμέτρων για ΔΑ

Το καλύτερο μοντέλο μετά την αναζήτηση πλέγματος στο σύνολο υπερπαραμέτρων παρουσιάζεται για λόγους πληρότητας και μελλοντικής αναφοράς στον Πίνακα 4.7.

max_features	min_samples_split	max_depth	min_samples_leaf
10	5	50	1

Πίνακας 4.7: Βέλτιστες υπερπαράμετροι ΔΑ

Στο Πίνακα 4.8 παρουσιάζονται τα αναλυτικά αποτελέσματα που λάβαμε για τον ταξινομητή ΔΑ με τις υπερπαραμέτρους του Πίνακα 4.7. Η σχηματική απεικόνιση των αποτελεσμάτων δίνεται στο Σχήμα 4.6 όπου φαίνεται ότι η ανάλυση κύριων συνιστωσών μειώνει και επιβαρύνει την απόδοση του ταξινομητή.



Σχήμα 4.6: Αξιολόγηση PCA για βέλτιστο ΔΑ

	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
ΔΑ	0.9707	0.9528	0.9395	0.9447
ΔΑ με PCA	0.9581	0.9415	0.9245	0.9313

Πίνακας 4.8: Αποτελέσματα βέλτιστου μοντέλου ΔΑ

4.4.3 Τυχαία Δάση

Τα τυχαία δάση αποτελούνται από πολλά δέντρα αποφάσεων με αποτέλεσμα η προεπεξεργασία των δεδομένων να καθίσταται περιττή, όπως εξηγήσαμε στην προηγούμενη υποενότητα. Για την εξαγωγή ισχυρών μοντέλων χρησιμοποιήσαμε και πάλι την τεχνική της αναζήτησης πλέγματος και καταλήξαμε στα δέκα καλύτερα μοντέλα τα οποία παρουσιάζουμε στον Πίνακα 4.9.

max_features	min_samples_split	max_depth	min_samples_leaf	n_estimators	accuracy
3	10	110	4	100	0.9852
3	8	110	3	300	0.9843
3	8	100	3	100	0.9831
3	10	90	3	200	0.9820
3	10	90	3	300	0.9832
3	8	90	3	100	0.9845
3	8	90	3	300	0.9812
3	10	80	3	1000	0.9809
3	8	80	3	300	0.9851
3	8	80	3	1000	0.9851

Πίνακας 4.9: Αναζήτηση πλέγματος υπεραπαραμέτρων για ΔΑ

Το καλύτερο μοντέλο μετά την αναζήτηση πλέγματος στο σύνολο υπερπαραμέτρων παρουσιάζεται για λόγους πληρότητας και μελλοντικής αναφοράς στον Πίνακα 4.10.

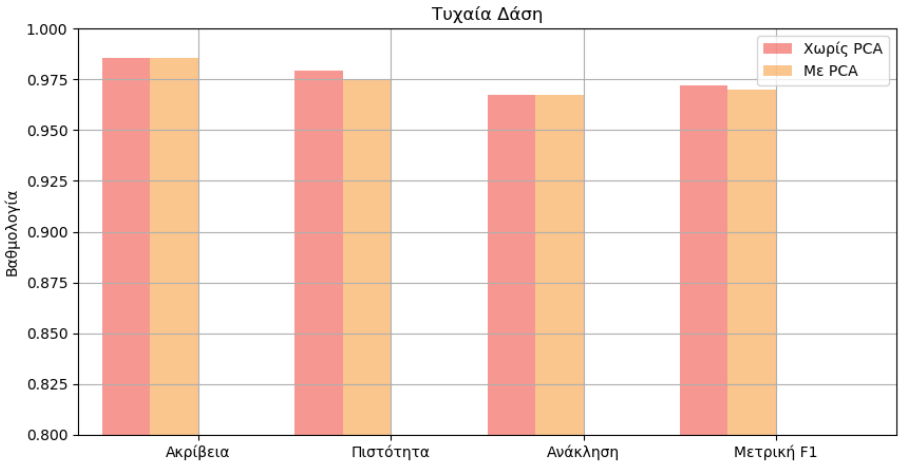
max_features	min_samples_split	max_depth	min_samples_leaf	n_estimators
3	10	110	4	100

Πίνακας 4.10: Βέλτιστες υπερπαράμετροι ΤΔ

Στο Πίνακα 4.11 παρουσιάζονται τα αναλυτικά αποτελέσματα που λάβαμε για τον ταξινομητή ΔΑ με τις υπερπαραμέτρους του Πίνακα 4.10. Η σχηματική απεικόνιση των αποτελεσμάτων δίνεται στο Σχήμα 4.7 όπου φαίνεται ότι η ανάλυση κύριων συνιστωσών μειώνει και επιβαρύνει την απόδοση του ταξινομητή.

	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
ΤΔ	0.9853	0.9794	0.9671	0.9721
ΤΔ με PCA	0.9741	0.9644	0.9527	0.9588

Πίνακας 4.11: Αποτελέσματα μετρικών καλύτερου μοντέλου ΤΔ



Σχήμα 4.7: Αξιολόγηση PCA για βέλτιστο ΤΔ

4.4.4 Διαβαθμιζόμενα Ενισχυμένα Δέντρα

Τα ενισχυμένα δέντρα είναι το μοντέλο που μας παρείχε τα καλύτερα αποτελέσματα από όλους τους ταξινομητές που πειραματιστήκαμε. Συγκεκριμένα είναι το μοντέλο το οποίο έχει επιλεχθεί από τις περισσότερες εργασίες που επικεντρώνονται στην εξαγωγή χαρακτηριστικών και έχει αποδεδειγμένα αρκετά ικανοποιητική απόδοση σε προβλήματα αναγνώρισης και κατηγοριοποίησης κακόβουλου λογισμικού. Το μοντέλο αυτό έχει έντονες υπολογιστικές απαιτήσεις ωστόσο είναι παραλληλοποιήσιμο και έχει υλοποιηθεί για να εκτελείται σε διαδεδομένα συστήματα κατανεμημένου υπολογισμού όπως το Spark[1].

Όσον αφορά το στάδιο της προ-επεξεργασίας, παρόλο που η εφαρμογή ανάλυσης κύριων συνιστωσών θα μείωνε σημαντικά τον χρόνο εκπαίδευσης του μοντέλου, οι δεντρικές δομές που δημιουργούνται κατά την εκπαίδευση του ταξινομητή δεν επωφελούνται από οποιαδήποτε μετρικής καθώς τα χαρακτηριστικά αρχίζουν και χάνουν την φυσική τους σημασία. Όπως και προηγουμένως επαληθεύσαμε τις προβλέψεις μας παρατηρώντας ότι η προσθήκη ανάλυσης κύριων συνιστωσών στην σωλήνευση του συστήματος μειώνει την επίδοση του συστήματος μας.

Σε πλήρη συμφωνία με τους προηγούμενους ταξινομητές συγκεντρώσαμε τις υπερπαραμέτρους που μας οδήγησαν στα δέκα καλύτερα μοντέλα μέσω πλεγματικής αναζήτησης και τις παρουσιάζουμε στον Πίνακα 4.12. Σε αντίθεση με τα προηγούμενα μοντέλα δεν είχαμε τη δυνατότητα να εκτελέσουμε εκτενή αναζήτηση πλέγματος για μεγάλα εύρη υπερπαραμέτρων λόγω των υπολογιστικών απαιτήσεων του αλγορίθμου εκπαίδευσης. Συνεπώς, διαλέξαμε ένα αρχικό σύνολο υπερπαραμέτρων με μεγάλη διακύμανση τιμών και στη συνέχεια συγκλίναμε όλο και περισσότερο προς τα καλύτερα αποτελέσματα μέ άπληστο τρόπο

Είναι φανερό από τον Πίνακα 4.12 ότι τα αποτελέσματα των ενισχυμένων δέντρων είναι αισθητά καλύτερα σε σχέση με τους υπόλοιπους ταξινομητές που δοκιμάσαμε οπότε το καλύτερο μοντέλο του πίνακα αυτού θα είναι και το τελικό προτεινόμενο μοντέλο του συστήματος μας.

[1] https://spark.apache.org/

gamma	colsample_bytree	learning_rate	max_depth	min_child_weight	n_estimators	accuracy
0	1	0.5	2	1	200	0.9953 (+/-0.004)
0	1	0.5	2	1	180	0.9953 (+/-0.004)
0	1	0.5	2	1	220	0.9952 (+/-0.004)
0	1	0.4	3	1	200	0.9951 (+/-0.003)
0	1	0.4	3	1	180	0.9951 (+/-0.003)
0	0.9	0.5	2	1	180	0.9949 (+/-0.004)
0	1	0.6	2	1	180	0.9948 (+/-0.005)
0	0.9	0.6	2	1	220	0.9948 (+/-0.005)
0	0.9	0.6	2	1	200	0.9948 (+/-0.005)
0	0.9	0.6	2	1	180	0.9948 (+/-0.005)

Πίνακας 4.12: Αναζήτηση πλέγματος υπερπαραμέτρων για Διαβαθμιζόμενα Ενισχυμένα Δέντρα

4.4.5 Προτεινόμενο μοντέλο

Το μοντέλο που προτείνουμε να χρησιμοποιηθεί για κατηγοριοποίηση κακόβουλου λογισμικού είναι ένας ταξινομητής διαβαθμισμένων ενισχυμένων δέντρων με σύνολο υπερπαραμέτρων που δίνεται στον Πίνακα 4.13.

gamma	colsample_bytree	learning_rate	max_depth	min_child_weight	n_estimators
0	1	0.5	2	1	200

Πίνακας 4.13: Βέλτιστες υπερπαράμετροι XGB

Στον Πίνακα 4.14 παρουσιάζονται όλες οι μετρικές που λάβαμε από την εφαρμογή διασταυρωμένης επικύρωσης για $k = 5$. Αυτά τα αποτελέσματα λόγω του τρόπου λειτουργίας της διασταυρωμένης επικύρωσης εξασφαλίζουν με μεγάλη πιθανότητα ότι το μοντέλο θα κυμαίνεται κοντά σε εκείνες τις τιμές και στο σύνολο επικύρωσης. Αυτές ήταν και οι τιμές που χρησιμοποιήσαμε για να αξιολογήσουμε και τα προηγούμενα μοντέλα. Επιπλέον, επειδή το συγκεκριμένο μοντέλο είναι και το τελικό στον Πίνακα 4.15 παρουσιάζουμε τις τιμές των μετρικών που προέκυψαν μετά από αξιολόγηση στο σύνολο επικύρωσης.

	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
XGB	0.995284	0.991967	0.98932	0.989537
XGB με PCA	0.982378	0.976967	0.97002	0.976477

Πίνακας 4.14: Αποτελέσματα προτεινόμενου μοντέλου 5-CV

	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
XGB	0.99632	0.994527	0.994939	0.995228

Πίνακας 4.15: Αποτελέσματα προτεινόμενου μοντέλου σε σύνολο επικύρωσης

4.4.6 Συγκεντρωτικά αποτελέσματα - Σύγκριση ταξινομητών

Σε αυτή την υποενότητα θα παρουσιάσουμε συγκεντρωτικά την απόδοση όλων των ταξινομητών που αναλύσαμε παραπάνω ως πρός τα αποτελέσματα τις διασταύρωσης επικύρωσης αλλά και του συνόλου επικύρωσης.

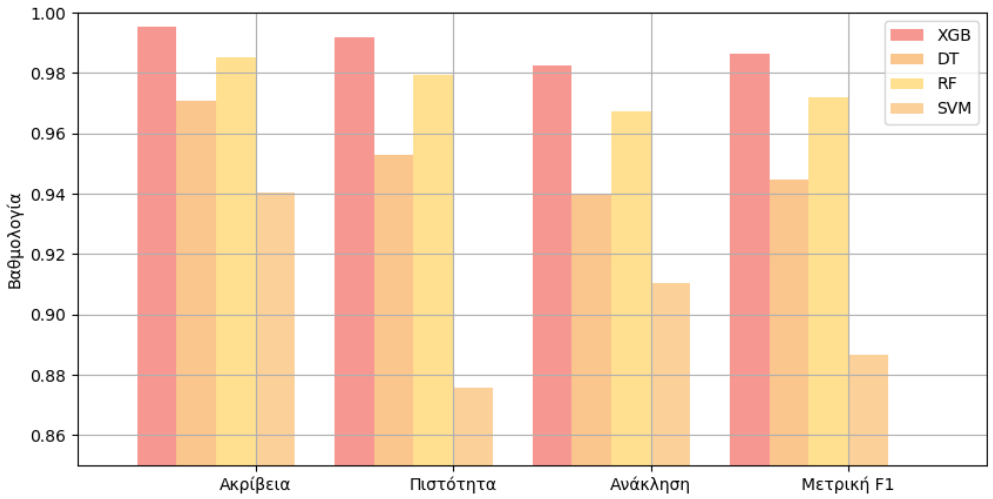
	5-CV			
	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
XGB	0.995284	0.991967	0.98232	0.986537
DT	0.970786	0.952858	0.939545	0.944771
RF	0.985278	0.979437	0.967141	0.972129
SVM	0.940299	0.875576	0.910227	0.886458

Πίνακας 4.16: Τελικά αποτελέσματα διασταυρωμένης επικύρωση $k = 5$

	Σύνολο Επικύρωσης			
	Ακρίβεια	Πιστότητα	Ανάκληση	F1-macro
XGB	0.99632	0.994527	0.993939	0.994228
DT	0.973321	0.922732	0.951383	0.933122
RF	0.986201	0.982377	0.967385	0.97417
SVM	0.944342	0.87645	0.919654	0.892052

Πίνακας 4.17: Τελικά αποτελέσματα επικύρωσης

Τέλος, στο σχήμα 4.8 παρουσιάζεται μια σχηματική σύγκριση των βέλτιστων αποτελεσμάτων ανά ταξινομητή για κάθε μετρική αξιολόγησης.



Σχήμα 4.8: Σύγκριση βέλτιστων αποτελεσμάτων ταξινομητών

Κεφάλαιο 5

Συμπεράσματα και Μελλοντικές Κατευθύνσεις

Σε αυτό το Κεφάλαιο θα παρουσιάσουμε συνοπτικά τα συμπεράσματα στα οποία καταλήξαμε μέσω της ενασχόλησης μας με το πρόβλημα της κατηγοριοποίησης κακόβουλου λογισμικού. Επιπλέον, θα προτείνουμε μελλοντικές κατευθύνσεις ώστε να βελτιωθεί περισσότερο η προβλεπτική ικανότητα των προτεινόμενων ταξινομητών και να συνεισφέρουμε στην περαιτέρω έρευνα στον τομέα.

5.1 Συμπεράσματα

Στην εργασία μας χρησιμοποιήσαμε τεχνικές μηχανικής μάθησης προκειμένου να κατηγοριοποιήσουμε εκτελέσιμα αρχεία στην αντίστοιχη οικογένεια κακόβουλου λογισμικού που ανήκουν. Προτείναμε, ένα ισχυρό σύνολο χαρακτηριστικών βασισμένο σε ειδικευμένη γνώση του τομέα της Ασφάλειας των Υπολογιστών το οποίο και χρησιμοποιήθηκε για την εκπαίδευση των ταξινομητών. Η απόδοση του τελικού μοντέλου απέδειξε και την καταλληλότητα χρήσης μηχανικής μάθησης στο πρόβλημα της ταξινόμησης κακόβουλου λογισμικού.

Με βάση τα αποτελέσματα που παρουσιάστηκαν στο Κεφάλαιο 4 παρατηρήθηκε ότι η εξαγωγή χαρακτηριστικών απέφερε αρκετά ικανοποιητικά αποτελέσματα τα οποία συγκρίνονται επάξια με εκείνα που έχουν προταθεί από state-of-the-art ερευνητικές εργασίες. Συμπεραίνουμε ότι είναι ιδιαίτερα σημαντική η εξαγωγή χαρακτηριστικών που σχετίζονται άμεσα με τον τομέα του προβλήματος. Η προσέγγιση να προσομοιώσουμε την χειροκίνητη εργασία ενός στατικού αναλυτή κακόβουλου λογισμικού απέδωσε διότι εκμεταλλευτήκαμε τον τρόπο εκπαίδευσης των δέντρων απόφασης, τυχαίων δασών και ενισχυμένων δέντρων. Ουσιαστικά, διάφορα χαρακτηριστικά που θα έλεγχε ένας αναλυτής κακόβουλου λογισμικού φροντίσαμε να εξαχθούν και να εκπαιδεύσουμε ταξινομητές που να μπορούν να τα ερμηνεύσουν κατάλληλα και να καταλήξουν στην τελική τους απόφαση.

Επιπλέον, αποδείξαμε ότι η μοντελοποίηση της συμπεριφοράς ενός κακόβουλου αρχείου δεν απαιτεί αποκλειστικά την εξαγωγή χρονοβόρων χαρακτηριστικών αλλά μπορεί να προκύψει και μέσω της προσεγμένης εξαγωγής συγκεκριμένων χαρακτηριστικών τα οποία προσδίδουν αρκετή πληροφορία και είναι λιγότερο επιρρεπή στον κίνδυνο υπερπροσαρμογής. Πολλά από τα μοντέλα που έχουν προταθεί στην βιβλιογραφία δεν ανταποκρίνονται στις απαιτήσεις ενός συστήματος πραγματικού χρόνου ενώ εμείς εξασφαλίσαμε ότι η μέση εξαγωγή χαρακτηριστικών και πρόβλεψη εκτελείται στην τάξη του δευτερολέπτου και χωρίς την απαίτηση υπερυπολογιστών ή ειδικευμένου υλικού. Επιπρόσθετα, λόγω του ότι ο χρόνος εξαγωγής αυξάνεται γραμμικά ως πρός το μέγεθος του αρχείου εξασφαλίζουμε ότι η κατηγοριοποίηση ενός αρχείου δεν θα οδηγηθεί ποτέ σε εξωφρενικούς χρόνους εκτέλεσης.

Τέλος, καταλήξαμε στο ότι η προ-επεξεργασία επιδρά αρνητικά στην απόδοση των συστημάτων μας καθώς περιπλέκεται στην προσπάθεια μας να προσομοιώσουμε τον τρόπο απόφασης ενός στατικού αναλυτή. Ο λόγος είναι ότι οι ταξινομητές που χρησιμοποιήσαμε εκπαιδεύονται με την κατασκευή δέντρων αποφάσεων οπότε θέλουμε τα χαρακτηριστικά να είναι τα ίδια που θα έβλεπε ένας στατικός αναλυτής προκειμένου να αποφανθεί. Επιπλέον, είναι πιο εύκολο να αξιολογήσουμε την αξία των χαρακτηριστικών που εξάγαμε καθώς δεν έχουμε την βοήθεια τεχνικών που θα απαλείψουν τα χαρακτηριστικά που δεν προσθέτουν κέρδος στην επίδοση του ταξινομητή μας.

5.2 Μελλοντικές Κατευθύνσεις

Σε αυτή την Ενότητα προτείνουμε πιθανές επεκτάσεις για ακόμα καλύτερη κατηγοριοποίηση κακόβουλου λογισμικού με χρήση επιπρόσθετων τεχνικών από αυτές που ασχοληθήκαμε στην εργασία.

5.2.1 Υβριδική Ανάλυση

Στην παρούσα εργασία ασχοληθήκαμε με την εξαγωγή χαρακτηριστικών μόνο μέσω στατικής ανάλυσης και παρόλα αυτά πετύχαμε ιδιαίτερα ικανοποιητικά αποτελέσματα. Μελετώντας ταυτόχρονα διάφορες εργασίες που έχουν ασχοληθεί με το κομμάτι της δυναμικής ανάλυσης πιστεύουμε ότι ο συνδυασμός των δύο μεθόδων θα μπορούσε να επιτύχει αποτελέσματα που θα άγγιζαν την αλάνθαστη ταξινόμηση. Ο λόγος είναι ότι πολλά χαρακτηριστικά των δυο μεθόδων αλληλοσυμπληρώνονται και μπορούν να φανερώσουν ακόμα και συμπεριφορές για τις οποίες έχει γίνει εκτενής προσπάθεια από τον συγγραφέα του κακόβουλου προγράμματος να αποκρυφτούν. Ωστόσο, πολλά κακόβουλα προγράμματα εντοπίζουν ότι τρέχουν σε εικονικό περιβάλλον και εκτελούν διαφορετικό payload, γεγονός το οποίο μπορεί να αποπροσανατολίσει την συνολική ανάλυση.

5.2.2 Ειδικά Χαρακτηριστικά

Το σύνολο δεδομένων που χρησιμοποιήσαμε περιείχε 9 διαφορετικές οικογένειες κακόβουλου λογισμικού, οι οποίες είναι αρκετά διαδεδομένες και υπάρχει αρκετό υλικό στη βιβλιογραφία που περιγράφει τον τρόπο λειτουργία τους και τις ιδιαιτερότητες τους. Συνεπώς, εξάγοντας χαρακτηριστικά που αναδεικνύουν στοιχεία της κάθε διαφορετικής οικογένειας θα μπορούσαμε να αυξήσουμε την προβλεπτική ισχύ του ταξινομητή για αναγνώριση τέτοιων οικογενειών. Ένας από τους λόγους που επιλέξαμε να μην το κάνουμε είναι γιατί προτιμήσαμε να προτείνουμε ένα μοντέλο το οποίο δε θα εξαρτάται από το σύνολο δεδομένων αλλά θα εξάγει γενικά χαρακτηριστικά τα οποία θα κατηγοριοποιούν οποιεσδήποτε οικογένειες κακόβουλου λογισμικού.

Επιπρόσθετα, θα μπορούσαμε να διασχίζουμε το αρχείο για να ανακατασκευάσουμε στοιχεία που αφορούν την ροή της εκτέλεσης του προγράμματος. Δεδομένου, ότι τα αρχεία τα λαμβάναμε από το πρόγραμμα IDA Pro θα ήταν αρκετά επίπονη χρονικά η εξαγωγή τέτοιων χαρακτηριστικών καθώς θα απαιτούνταν η υλοποίηση parser που θα κάνει την συγκεκριμένη δουλειά.

Βιβλιογραφία

[Abou04] Tony Abou-Assaleh, Nick Cercone, Vlado Keselj and Ray Sweidan, "N-Gram-Based Detection of New Malicious Code", in *Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - Volume 02*, COMPSAC '04, pp. 41–42, Washington, DC, USA, 2004, IEEE Computer Society.

[Ahma16] Mansour Ahmadi, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov and Giorgio Giacinto, "Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification", in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pp. 183–194, New York, NY, USA, 2016, ACM.

[Aike16] Mary Aiken, Ciaran Mc Mahon, Ciaran Haughton, Laura O'Neill and Edward O'Carroll, "A consideration of the social impact of cybercrime: examples from hacking, piracy, and child abuse material online", *Contemporary Social Science*, vol. 11, no. 4, pp. 373–391, 2016.

[AVTE19] AV-TEST, "Malware Population Growth Statistics", 2019.

[Bays13] Donabelle Baysa, Richard M. Low and Mark Stamp, "Structural entropy and metamorphic malware", *J. Computer Virology and Hacking Techniques*, vol. 9, no. 4, pp. 179–192, 2013.

[Bert96] Dimitri P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*, Athena Scientific, 1 edition, 1996.

[Bila06] Daniel Bilar, "Statistical Structures: Fingerprinting Malware for Classification and Analysis", 01 2006.

[Bose92] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", in *Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152, ACM, June 1992.

[Brad97] Andrew P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", *Pattern Recognition*, vol. 30, no. 7, pp. 1145 – 1159, 1997.

[Brei96] Leo Breiman, "Bagging Predictors", *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996.

[Brei99] Leo Breiman, "Pasting Small Votes for Classification in Large Databases and On-Line", *Machine Learning*, vol. 36, no. 1-2, pp. 85–103, 1999.

[Burg98] Christopher J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition.", *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[Cesa13] Silvio Cesare, Yang Xiang and Wanlei Zhou, "Malwise—An Effective and Efficient Classification System for Packed and Polymorphic Malware", *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1193–1206, June 2013.

[Chri05] Mihai Christodorescu, Somesh Jha, Sanjit A. Seshia, Dawn Song and Randal E. Bryant, "Semantics-Aware Malware Detection", in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pp. 32–46, Washington, DC, USA, 2005, IEEE Computer Society.

[Diet98] Thomas G. Dietterich, "Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms", *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[Dola] Stephen Dolan, "mov is Turing-complete".

[Eagl08] Chris Eagle, *The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler*, No Starch Press, San Francisco, CA, USA, 2008.

[Fern18] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk and Francisco Herrera, *Learning from Imbalanced Data Sets*, Springer, 2018.

[Ghia12] Mahboobe Ghiasi, Ashkan Sami and Zahra Salehi, "Dynamic malware detection using registers values set analysis", pp. 54–59, 09 2012.

[Ghia15] Mahboobe Ghiasi, Ashkan Sami and Zahra Salehi, "Dynamic VSA", *Eng. Appl. Artif. Intell.*, vol. 44, no. C, pp. 111–122, September 2015.

[Gran03] Yves Grandvalet and Stéphane Canu, "Adaptive Scaling for Feature Selection in SVMs", in S. Becker, S. Thrun and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pp. 569–576, MIT Press, 2003.

[Grif09] Kent Griffin, Scott Schneider, Xin Hu and Tzi-Cker Chiueh, "Automatic Generation of String Signatures for Malware Detection", in *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pp. 101–120, Berlin, Heidelberg, 2009, Springer-Verlag.

[Hu09] Xin Hu, Tzi-cker Chiueh and Kang G. Shin, "Large-scale Malware Indexing Using Function-call Graphs", in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pp. 611–620, New York, NY, USA, 2009, ACM.

[Jame14] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, Springer Publishing Company, Incorporated, 2014.

[Kast02] Daniel Kästner and Stephan Wilhelm, "Generic Control Flow Reconstruction from Assembly Code", in *Proceedings of the Joint Conference on Languages, Compilers and Tools for Embedded Systems: Software and Compilers for Embedded Systems*, LCTES/SCOPES '02, pp. 46–55, New York, NY, USA, 2002, ACM.

[Kaus] Kevadia Kaushal, Prashant B. Swadas and Nilesh B. Prajapati, "Metamorphic Malware Detection Using Statistical Analysis".

[Kerk14] Max Kerkers, José Jair Santanna and Anna Sperotto, "Characterisation of the Kelihos.B Botnet", in *Monitoring and Securing Virtualized Networks and Services - 8th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2014, Brno, Czech Republic, June 30 - July 3, 2014. Proceedings*, pp. 79–91, 2014.

[Koha95] Ron Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection", in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pp. 1137–1143, San Francisco, CA, USA, 1995, Morgan Kaufmann Publishers Inc.

[Kolt04] Jeremy Z. Kolter and Marcus A. Maloof, "Learning to Detect Malicious Executables in the Wild", in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 470–478, New York, NY, USA, 2004, ACM.

[Lee10] Byoungyoung Lee, Yuna Kim and Jong Kim, "binOb+: a framework for potent and stealthy binary obfuscation.", in Dengguo Feng, David A. Basin and Peng Liu, editors, *AsiaCCS*, pp. 271–281, ACM, 2010.

[Lyda07] Robert Lyda and James Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware", *IEEE Security and Privacy*, vol. 5, no. 2, pp. 40–45, March 2007.

[Mitc97] Thomas M. Mitchell, *Machine Learning*, McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[Nair09] Vinod P. Nair, Vijay Laxmi, Manoj Singh Gaur, G. V. S. S. Phani Kumar and Yadvendra S. Chundawat, "Static CFG analyzer for metamorphic Malware code", in *SIN*, pp. 225–228, ACM, 2009.

[Naro15] Masoud Narouei, Mansour Ahmadi, Giorgio Giacinto, Hassan Takabi and Ashkan Sami, "DLLMiner: structural mining for malware detection", *Security and Communication Networks*, vol. 8, no. 18, pp. 3311–3322, 2015.

[Parv11] Hamid Parvin, Behrouz Minaei, Hossein Karshenas and Akram Beigi, "A New N-gram Feature Extraction-selection Method for Malicious Code", in *Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms - Volume Part II*, ICANNGA'11, pp. 98–107, Berlin, Heidelberg, 2011, Springer-Verlag.

[Past19] Sergio Pastrana and Guillermo Suarez-Tangil, "A First Look at the Crypto-Mining Malware Ecosystem: A Decade of Unrestricted Wealth", *CoRR*, vol. abs/1901.00846, 2019.

[Perd08] Roberto Perdisci, Andrea Lanzi and Wenke Lee, "Classification of Packed Executables for Accurate Computer Virus Detection", *Pattern Recogn. Lett.*, vol. 29, no. 14, pp. 1941–1946, October 2008.

[Quin86] J. R. Quinlan, “Induction of Decision Trees”, *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, March 1986.

[Quin93] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Quin96] J. Ross Quinlan, “Learning Decision Tree Classifiers”, *ACM Comput. Surv.*, vol. 28, no. 1, pp. 71–72, 1996.

[Raff17] Edward Raff, Jared Sylvester and Charles Nicholas, “Learning the PE Header, Malware Detection with Minimal Domain Knowledge”, in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, pp. 121–132, 2017.

[Rahm14] Akhlaqur Rahman and Sumaira Tasnim, “Ensemble Classifiers and Their Applications: A Review”, *CoRR*, vol. abs/1404.4088, 2014.

[Refa09] Payam Refaeilzadeh, Lei Tang and Huan Liu, “Cross-Validation.”, in Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pp. 532–538, Springer US, 2009.

[Rone18] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov and Mansour Ahmadi, “Microsoft Malware Classification Challenge”, *CoRR*, vol. abs/1802.10135, 2018.

[Sami10] Ashkan Sami, Babak Yadegari, Hossein Rahimi, Naser Peiravian, Sattar Hashemi and Ali Hamze, “Malware Detection Based on Mining API Calls”, in *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC ’10, pp. 1020–1025, New York, NY, USA, 2010, ACM.

[Sant13] Igor Santos, Felix Brezo, Xabier Ugarte-Pedrero and Pablo G. Bringas, “Opcode Sequences As Representation of Executables for Data-mining-based Unknown Malware Detection”, *Inf. Sci.*, vol. 231, pp. 64–82, May 2013.

[Schu01] Matthew G. Schultz, Eleazar Eskin, Erez Zadok and Salvatore J. Stolfo, “Data Mining Methods for Detection of New Malicious Executables”, in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP ’01, pp. 38–, Washington, DC, USA, 2001, IEEE Computer Society.

[Shab09] Asaf Shabtai, Robert Moskovitch, Yuval Elovici and Chanan Glezer, “Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-art Survey”, *Inf. Secur. Tech. Rep.*, vol. 14, no. 1, pp. 16–29, February 2009.

[Vago08] David Vagoum, “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation”, *Mach. Learn. Technol.*, vol. 2, 01 2008.

[Vapn98] Vladimir N. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

[Wang09] Tzu-Yen Wang, Chin-Hsiung Wu and Chu-Cheng Hsieh, “Detecting Unknown Malicious Executables Using Portable Executable Headers”, pp. 278–284, 01 2009.

[Will07] Carsten Willems, Thorsten Holz and Felix Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox", *IEEE Security and Privacy*, vol. 5, no. 2, pp. 32–39, March 2007.

[Wong06] Wing Wong and Mark Stamp, "Hunting for metamorphic engines", *Journal in Computer Virology*, vol. 2, no. 3, pp. 211–229, 2006.

[Xu14] Liang Xu, Fangqi Sun and Zhendong Su, "Constructing Precise Control Flow Graphs from Binaries", 2014.

[Ye07] Yanfang Ye, Dingding Wang, Tao Li and Dongyi Ye, "IMDS: Intelligent Malware Detection System", in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pp. 1043–1047, New York, NY, USA, 2007, ACM.