Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**(Ελληνικά)** **Τεχνικές Εντοπισμού Κακόβουλου Λογισμικού** **(Αγγλικά)** **Malware Detection Techniques**
Ονοματεπώνυμο Φοιτητή	**Δέσποινα Ανδρεαδάκη**
Πατρώνυμο	**Δημήτριος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 14005**
Επιβλέπων	**Κωνσταντίνος Πατσάκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2018**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Κωνσταντίνος Πατσάκης Επίκουρος Καθηγητής	Ευθύμιος Αλέπης Επίκουρος Καθηγητής	Ιωάννης Τασούλας Επίκουρος Καθηγητής

Περίληψη

Με το γενικό όρο malware ονομάζουμε το κακόβουλο λογισμικό που έχει σχεδιαστεί με σκοπό την πρόκληση βλάβης σε δίκτυα και υπολογιστές. Με σκοπό την αποφυγή του εντοπισμού του, οι συγγραφείς του έχουν αναπτύξει διάφορες τεχνικές απόκρυψης του κώδικα, όπως ο πολυμορφισμός και ο μεταμορφισμός. Η διπλωματική αυτή εργασία παρουσιάζει τεχνικές ανίχνευσης κακόβουλου λογισμικού από την σύγχρονη βιβλιογραφία, που ανήκουν είτε στην κατηγορία της εύρεσης ανωμαλιών είτε βάσει υπογραφών. Υποκατηγορία της πρώτης είναι και ο εντοπισμός βάσει προδιαγραφών (specification-based). Κάθε μία από τις παραπάνω κατηγορίες μπορεί να εφαρμοστεί είτε μέσω στατικής προσέγγισης, είτε μέσω δυναμικής, είτε μέσω συνδυασμού των δύο (υβριδική προσέγγιση).

Abstract

Malware is short for malicious software, which is designed to cause damage to networks and computers. To avoid detection, the intruders have developed various obfuscation techniques, such as polymorphism and metamorphism. This thesis presents malware detection techniques from modern bibliography, belonging to either anomaly-based or signature-based techniques, with specification-based techniques being a subcategory of the former. Each of the detection techniques can employ either a dynamic, a static or a hybrid approach (which is a combination of the other two).

Περιεχόμενα

1. Εισαγωγή στην Ασφάλεια του Διαδικτύου

Ένα από τα βασικότερα ίσως χαρακτηριστικά του Διαδικτύου, ειδικά όταν εξετάζεται από την οπτική γωνία της ασφάλειας, είναι ότι αυτό είναι ανοικτό και δημόσιο. Αυτό συνεπάγεται ότι, αφενός, ο καθένας μπορεί να το χρησιμοποιήσει και, αφετέρου, ότι το περιεχόμενό του είναι προσβάσιμο σε όλους. Το Διαδίκτυο δεν γνωρίζει ούτε ενδιαφέρεται για το ποιοι είναι οι χρήστες του, κάτι που, όμως, έχει και αρνητική πλευρά.

Λόγω της φύσης του αυτής, το Διαδίκτυο είναι ευάλωτο σε πολλών ειδών επιθέσεις από τους λεγόμενους κυβερνο-εγκληματίες (cyber-criminals), οι οποίοι έχουν αναπτύξει αρκετές τεχνικές με στόχο την παραβίαση της ιδιωτικότητας και ακεραιότητας οργανισμών, επιχειρήσεων, τραπεζικών λογαριασμών κτλ. Σήμερα, πολλοί είναι εκείνοι που πιστεύουν πως η ποσότητα κακόβουλου λογισμικού – malware, που τίθεται σε κυκλοφορία στο Διαδίκτυο, ενδέχεται και να ξεπερνάει ακόμα την έκδοση έγκυρου και ασφαλή λογισμικού.

Στις σελίδες που ακολουθούν αναλύεται ο όρος κακόβουλο λογισμικό και καταγράφονται κάποιοι από τους πιο γνωστούς τύπους αυτού. Ακολουθεί η παρουσίαση πιθανών τρόπων ανίχνευσης και αντιμετώπισης κακόβουλου λογισμικού, το οποίο αποτελεί το κεντρικό σημείο της παρούσας εργασίας. Όπως θα αναλυθεί περαιτέρω στην πορεία, οι τεχνικές αυτές χωρίζονται σε διάφορες κατηγορίες ανάλογα με την προσέγγιση που ακολουθούν. Τέλος, αναφέρονται τα συμπεράσματα που προέκυψαν βάση της έρευνας αυτής, καθώς και η ταξινόμηση των διαφόρων τεχνικών στις επιμέρους κατηγορίες.

1.1. Malware – Κακόβουλο Λογισμικό

1.1.1. Ορισμός

Τι ακριβώς εννοούμε, όμως, με τον όρο malware; Ο όρος αυτός είναι συντομία του όρου malicious software ή, στα ελληνικά, κακόβουλο λογισμικό. Αναφέρεται σε οποιοδήποτε κομμάτι λογισμικού που δημιουργήθηκε με σκοπό την εισβολή και πρόκληση ζημιάς σε δεδομένα ή συσκευές. Έχει τη δυνατότητα υποκλοπής ευαίσθητων πληροφοριών, μόλυνσης πολλαπλών αρχείων, καθώς και εξάπλωσης αυτής της μόλυνσης σε ολόκληρο το σύστημα. Σαν πιο γενικευμένος όρος περιλαμβάνει και τους διάφορους τύπους απειλών της ασφάλειας του υπολογιστή που υπάρχουν, όπως, για παράδειγμα, Trojan horses, spyware, adware, computer viruses. Μπορεί να πάρει τη μορφή εκτελέσιμου κώδικα, script, ανοικτού λογισμικού κτλ. Καθώς προσδιορίζεται από την κακόβουλη πρόθεσή του, δεν περιλαμβάνει λογισμικό που μπορεί να προκαλέσει ακούσια βλάβη (λόγω ανεπαρκειών στον προγραμματισμό του, για παράδειγμα). Η καλύτερη μορφή αντιμετώπισής του είναι η όσο το δυνατόν ταχύτερη ανίχνευση και αφαίρεσή του από το μολυσμένο σύστημα, με σκοπό τον περιορισμό της ζημιάς που προκαλεί.

Με σκοπό την αποφυγή του εντοπισμού του, οι συγγραφείς κακόβουλου λογισμικού έχουν αναπτύξει διάφορες τεχνικές απόκρυψης του κώδικά του. Έτσι προκύπτουν οι λεγόμενοι πολυμορφικοί και μεταμορφικοί ιοί, με τον πρώτο να κρυπτογραφεί το εαυτό του με διαφορετικό κλειδί ύστερα από κάθε μόλυνση και το δεύτερο να ακολουθεί ακόμα πιο πολύπλοκες τεχνικές αλλαγής της εμφάνισής του. Πολλές από τις τεχνικές που θα παρουσιαστούν στην πορεία επικεντρώνεται στην ανίχνευση αυτού του τύπου κακόβουλου λογισμικού.

1.1.2. Τύποι Malware

Με βάση τον τρόπο που λειτουργεί και τη ζημιά που προκαλεί το εκάστοτε κακόβουλο λογισμικό, κατηγοριοποιείται σε πιο συγκεκριμένους τύπους. Μερικοί από τους πιο γνωστούς είναι οι ακόλουθοι[1]:

- *Adware*: Τύπος malware που παρέχει αυτόματα διαφημίσεις, με πιο γνωστές τις λεγόμενες pop-up διαφημίσεις. Το μεγαλύτερο μέρος του δημιουργείται ή χορηγείται από διαφημιζόμενους με σκοπό τη δημιουργία εσόδων.
- *Bots*: Προγράμματα λογισμικού που χρησιμοποιούνται για την αυτόματη εκτέλεση συγκεκριμένων λειτουργιών. Αν και τις περισσότερες φορές δημιουργούνται για σχετικά αβλαβείς σκοπούς (στο χώρο των βιντεοπαιχνιδιών, για παράδειγμα), σχετίζονται επίσης με επιθέσεις DDoS, spam διαφημίσεων σε ιστότοπους και διανομή κακόβουλου λογισμικού μεταμφιεσμένου σε δημοφιλή στοιχεία αναζήτησης προς λήψη.
- *Bug*: Το bug αναφέρεται σε κάποιο ελάττωμα ή ανεπάρκεια του κώδικα, που οδηγεί σε κάποιο ανεπιθύμητο αποτέλεσμα, και είναι συνήθως αποτέλεσμα ανθρώπινου σφάλματος. Οι σοβαρότεροι τύποι bugs μπορεί να προκαλέσουν από το λεγόμενο «κρασάρισμα» του συστήματος, μέχρι και την παράκαμψη των δικαιωμάτων πρόσβασης και την υποκλοπή στοιχείων.
- *Ransomware*: Ο τύπος αυτός κακόβουλου λογισμικού περιορίζει την πρόσβαση του χρήστη στον υπολογιστή είτε μέσω της κρυπτογράφησης αρχείων στον σκληρό δίσκο είτε με κλείδωμα του συστήματος. Παράλληλα, προβάλλει μηνύματα που σκοπό έχουν να αναγκάσουν το χρήστη να πληρώσει κάποιο ποσό, ως αντίτιμο, προκειμένου να καταργηθούν οι περιορισμοί και να ανακτήσει ο χρήστης την πρόσβαση στον υπολογιστή του.

[1] Βασισμένο σε αποσπάσματα από τα άρθρα: *An Introduction to Internet Security and Firewall Policies*, William Hugh Murray και *Common Malware Types: Cybersecurity 101*, Neil DuPaul, October 2012

- *Rootkit*: Ο τύπος αυτός έχει σχεδιαστεί με σκοπό την απομακρυσμένη πρόσβαση και έλεγχο του υπολογιστή, χωρίς τον εντοπισμό του από τους χρήστες ή τα προγράμματα ασφαλείας. Μπορεί να τροποποιήσει τις ρυθμίσεις του συστήματος, να κλέψει δεδομένα ή, ακόμα, και να εγκαταστήσει κρυμμένο κακόβουλο λογισμικό.
- *Spyware*: Το spyware, όπως υποδηλώνει και το όνομά του, κατασκοπεύει τον χρήστη εν αγνοία του. Αυτό μπορεί να σημαίνει παρακολούθηση δραστηριοτήτων, συλλογή και καταγραφή των πλήκτρων που πιο συχνά χρησιμοποιεί ο χρήστης, συλλογή δεδομένων (όπως, για παράδειγμα, στοιχεία σύνδεσης και πληροφορίες λογαριασμού), καθώς και τροποποίηση των στοιχείων ασφαλείας.
- *Trojan_Horse*: Συχνά, αναφέρεται και απλά ως Trojan. Μεταμφιέζεται σε κανονικό αρχείο ή πρόγραμμα με σκοπό να ξεγελάσει τους χρήστες και αυτοί να κατεβάσουν και να εγκαταστήσουν κάποιο κακόβουλο λογισμικό. Από τη στιγμή που μολυνθεί ο υπολογιστής, είναι δυνατόν για τον εισβολέα να κλέψει δεδομένα, να τροποποιήσει αρχεία, καθώς και να εγκαταστήσει περισσότερα κακόβουλα λογισμικά, μεταξύ άλλων.
- *Virus*: Ο ιός είναι τύπος malware που μπορεί να αντιγραφεί και να εξαπλωθεί σε άλλους υπολογιστές, μέσω της προσκόλλησής του σε προγράμματα και κώδικες, τη στιγμή που ο χρήστης εκκινεί το μολυσμένο πρόγραμμα. Οι ιοί μπορούν να χρησιμοποιηθούν για την υποκλοπή δεδομένων και χρημάτων, τη βλάβη κεντρικών υπολογιστών και δικτύων, τη δημιουργία botnets (συλλογές ηλεκτρονικών υπολογιστών που ελέγχονται από τρίτους) και άλλα.
- *Worms*: Από τους πιο γνωστούς τύπους malware, μπορούν και αυτοί να αντιγραφούν και να εξαπλωθούν σε άλλους υπολογιστές, όπως οι ιοί, αλλά σε αντίθεση με αυτούς, δεν χρειάζονται την παρεμβολή του χρήστη για να το κάνουν. Αξιοποιούν τις αδυναμίες του λειτουργικού συστήματος για να εξαπλωθούν σε δίκτυα υπολογιστών και προκαλούν συνήθως ζημιά σε αυτά καταναλώνοντας το bandwidth (εύρος ζώνης) και υπερφορτώνοντας τους web servers (διακομιστές ιστού). Επίσης, μέσω της χρήσης ειδικού κώδικα, τα λεγόμενα «payloads», τα οποία είναι γραμμένα για να εκτελούν ενέργειες στους επηρεαζόμενους υπολογιστές πέραν της απλής εξάπλωσής τους, μπορούν να οδηγήσουν στην υποκλοπή στοιχείων, διαγραφή αρχείων και δημιουργία botnets.

1.1.3. Αντιμετώπιση του Κακόβουλου Λογισμικού

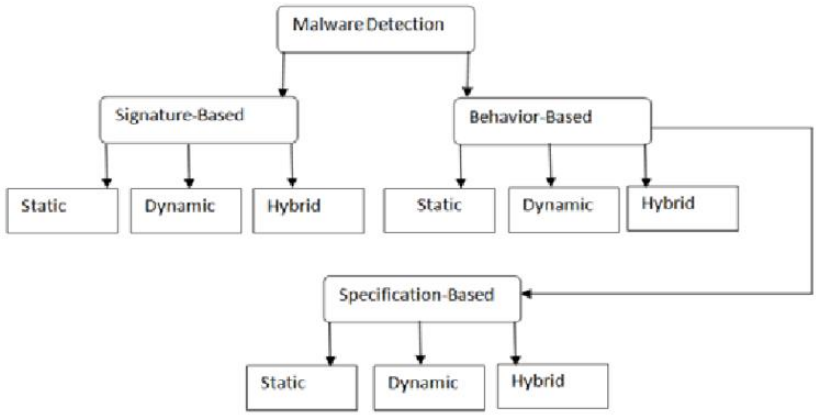
Όπως αναφέρθηκε και προηγουμένως, κάθε τύπος κακόβουλου λογισμικού έχει το δικό του τρόπο πρόκλησης βλάβης σε δεδομένα και συστήματα υπολογιστών. Ο όγκος και η πολυπλοκότητα τόσο των γνωστών, όσο και άγνωστων, μορφών κακόβουλου λογισμικού καθιστά δύσκολο τον εντοπισμό και αντιμετώπισή τους. Ενώ, φυσικά, η αποφυγή συνδέσμων, ιστότοπων και ηλεκτρονικών μηνυμάτων που φαίνονται ύποπτα και η συνεχής ενημέρωση λογισμικού με τις τρέχουσες διορθώσεις σε πιθανές αδυναμίες του συστήματος είναι πολύ καλές τακτικές, εντούτοις δεν είναι αρκετές. Για το σκοπό αυτό δημιουργήθηκαν τα λεγόμενα anti-malware προγράμματα.

Το anti-malware είναι τύπος λογισμικού κατάλληλα σχεδιασμένου με σκοπό την πρόληψη, τον εντοπισμό και την αφαίρεση κακόβουλου λογισμικού, καθώς και την αποκατάσταση ζημιάς που μπορεί να προκλήθηκε από αυτό. Τα anti-malware λογισμικά προστατεύουν, συνήθως, από τους περισσότερους τύπους malware, όπως αυτοί αναφέρθηκαν παραπάνω, και μπορούν να εγκατασταθούν σε μεμονωμένους υπολογιστές, διακομιστές πύλης ή συσκευές δικτύου.

2. Τεχνικές Εντοπισμού Κακόβουλου Λογισμικού

Οι τεχνικές που χρησιμοποιούν τα anti-malware λογισμικά μπορούν να χωριστούν σε δύο κύριες κατηγορίες[2]: τον εντοπισμό με βάση πιθανές ανωμαλίες στη συμπεριφορά (*anomaly-based* or *behavior-based*), υποκατηγορία της οποίας είναι και η ανίχνευση βάσει προδιαγραφών (*specification-based*), και με βάση τις υπογραφές (*signature-based*). Οι πρώτες χρησιμοποιούν νόρμες για το τι σημαίνει φυσιολογική συμπεριφορά ενός προγράμματος και αποτελούνται συνήθως από δύο φάσεις: τη φάση εκμάθησης (*training phase*), όπου δημιουργείται το προφίλ της θεωρούμενης ως κανονικής συμπεριφοράς του προγράμματος, και τη φάση ελέγχου (*testing phase*), όπου συγκρίνεται η συμπεριφορά του υπό εξέταση προγράμματος με το προφίλ της προηγούμενης φάσης. Το βασικό τους μειονέκτημα είναι το σχετικά μεγάλο ποσοστό ψευδώς θετικών αποτελεσμάτων που παρουσιάζουν. Οι δεύτερες συγκρίνουν κομμάτια του προς-εξέταση-κώδικα (συνήθως με τη μορφή ακολουθιών από bytes) με τις βάσεις γνωστών αναγνωριστικών που ανήκουν σε κακόβουλα λογισμικά, ενώ το μεγαλύτερό τους μειονέκτημα είναι ότι δεν μπορούν να ανιχνεύσουν νέες επιθέσεις των οποίων το μοτίβο δεν είναι ακόμα διαθέσιμο και, άρα, δεν είναι αποθηκευμένο στις εκάστοτε βάσεις προς σύγκριση.

Οι προαναφερθείσες τεχνικές μπορούν να υλοποιηθούν με τρεις διαφορετικούς τρόπους: *στατικά*, *δυναμικά* ή *υβριδικά*. Η κατηγοριοποίηση αυτή βασίζεται στον τρόπο με τον οποίο η εκάστοτε τεχνική συλλέγει τις απαραίτητες πληροφορίες για τον εντοπισμό του κακόβουλου λογισμικού. Η στατική ανάλυση χρησιμοποιεί τις συντακτικές και δομικές ιδιότητες του εκάστοτε προγράμματος που βρίσκεται υπό έλεγχο, πριν αυτό εκτελεστεί. Εν αντιθέσει, η δυναμική ανάλυση προσπαθεί να εντοπίσει το κακόβουλο λογισμικό κατά την εκτέλεση του προγράμματος αυτού. Η υβριδική ανάλυση, όπως υποδηλώνει και το όνομά της, χρησιμοποιεί συνδυαστικά τις δύο προηγούμενες τεχνικές.



Εικόνα 1 - Τύποι Τεχνικών Εντοπισμού Κακόβουλου

Στη συνέχεια, θα παρουσιαστούν κάποιες από τις τεχνικές που προτείνονται από τις παραπάνω κατηγορίες, με σκοπό την βελτιστοποίηση του τρόπου εντοπισμού κακόβουλου λογισμικού, καθώς και μία μικρή περιγραφή του τρόπου λειτουργίας τους.

[2] Οι τεχνικές ανίχνευσης βάσει προδιαγραφών (specification-based detection) αποτελούν υποκατηγορία των τεχνικών βάσει ανωμαλιών στη συμπεριφορά (anomaly-based detection) και για αυτό δεν αναφέρονται ως η Τρίτη κύρια κατηγορία. Εντούτοις, τέτοιου είδους προσεγγίσεις αποτελούν μεγάλο κομμάτι της σχετικής έρευνας και ακολουθούνται από πολλές από τις τεχνικές που αναλύονται στην πορεία.

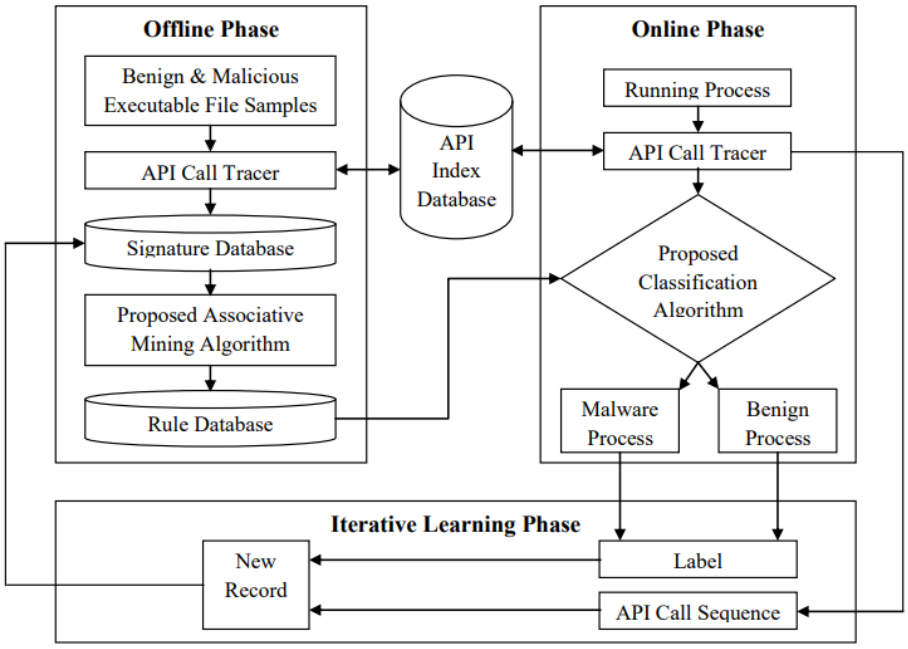
2.1. Τεχνικές

2.1.1. Χρήση της Μηχανικής Μάθησης και των API κλήσεων των Windows

Οι C. Ravi και R Manoharan στο έργο τους [1] προτείνουν μια τεχνική που χρησιμοποιεί ακολουθίες API κλήσεων των Windows. Γίνεται χρήση μιας Μαρκοβιανής Αλυσίδας τάξης 3[3] για να μοντελοποιήσουν τις API κλήσεις. Συνολικά, ελέγχεται ένα ελάχιστο σύνολο από API κατηγορίες.

Το σύστημα αποτελείται από τρεις φάσεις: Offline, Online και Iterative (επαναληπτική). Η Offline φάση περιλαμβάνει: δείγματα εκτελέσιμων αρχείων (κακόβουλων και μη), ανιχνευτή των API κλήσεων (*API Call Tracer*), βάση δεδομένων των υπογραφών και των κανόνων, καθώς και τον προτεινόμενο Association mining αλγόριθμο. Περιλαμβάνεται επίσης και μία βάση δεδομένων (*API Index database*), η οποία περιέχει τα IDs, με τη μορφή ακεραίων αριθμών, που αντιπροσωπεύουν την κάθε κλήση. Η ακολουθία που παράγεται από τον ανιχνευτή κλήσεων, μετατρέπεται σε ακολουθία ακεραίων αριθμών και αποθηκεύεται στη βάση δεδομένων των υπογραφών. Η Online φάση περιλαμβάνει: τη διαδικασία εκτέλεσης, ανιχνευτή των API κλήσεων και τον προτεινόμενο αλγόριθμο κατηγοριοποίησης της διαδικασίας αυτής σε κακόβουλη ή μη. Χρησιμοποιείται και εδώ η API Index database για την παραγωγή της ακολουθίας ακεραίων των κλήσεων, οι οποίες μετά χωρίζονται σε ακολουθίες των 4 (4-gram). Μέσω της βάσης δεδομένων των κανόνων, υπολογίζονται οι μέσοι συντελεστές εμπιστοσύνης για όλες τις 4-gram ακολουθίες του δείγματος και σε περίπτωση που ο συντελεστής του κακόβουλου μέρους είναι μεγαλύτερος από του ομαλού, τότε η διεργασία θεωρείται κακόβουλη. Τέλος, κατά την Iterative φάση εκμάθησης, γίνεται κατ' επανάληψη κατηγοριοποίηση της εφαρμογής και ενισχύεται το κομμάτι εκμάθησης του μοντέλου, μέσω της προσθήκης της ακολουθίας API κλήσεων και κατηγορίας στη βάση υπογραφών.

[3] Με βάση το λεξικό της Οξφόρδης: "*Μαρκοβιανή αλυσίδα ονομάζεται το στοχαστικό μοντέλο που περιγράφει μία ακολουθία από πιθανά γεγονότα, όπου η πιθανότητα να πραγματοποιηθεί κάποιο από αυτά εξαρτάται μόνο από την κατάσταση που βρέθηκε από το προηγούμενο γεγονός*." Η τάξη 3 της αλυσίδας υποδηλώνει ότι η μελλοντική κατάσταση εξαρτάται από τις προηγούμενες 3 καταστάσεις.



Εικόνα 2 - Αρχιτεκτονική του προτεινόμενου συστήματος ανίχνευσης κακόβουλου λογισμικού

Οι προτεινόμενοι αλγόριθμοι, υλοποιημένοι σε γλώσσα Visual C++, είναι οι εξής:

- *Call Tracer algorithm*, για την ανίχνευση των API κλήσεων
- *Rule Miner algorithm*, την εξόρυξη κανόνων και
- *Classifier algorithm*, για την κατηγοριοποίηση της διαδικασίας σε κακόβουλη ή μη.

Τα πειράματα που πραγματοποιήθηκαν έγιναν σε περιβάλλον Windows XP, με Intel Pentium D 2.80GHz CPU και 1GB RAM. Στην αρχή συγκεντρώθηκαν εκτελέσιμα αρχεία των Windows, κακόβουλα και μη. Τα κακόβουλα αρχεία περιλάμβαναν στην πλειονότητά τους worms, Trojan horses και backdoors, ενώ τα ασφαλή αρχεία προήλθαν από ένα πρόσφατα εγκαταστημένο Windows XP λειτουργικό σύστημα. Στο σύνολό τους τα ασφαλή αρχεία ήταν 94 και τα κακόβουλα 179 και, από αυτά, τα 68 ασφαλή και 95 κακόβουλα χρησιμοποιήθηκαν στη φάση της εξάσκησης (training) του μοντέλου. Τα υπόλοιπα 26 ασφαλή και 84 κακόβουλα χρησιμοποιήθηκαν για την testing φάση της τεχνικής.

Εν τέλει, σε σύγκριση με τα υπάρχοντα συστήματα εντοπισμού που βασίζονται στην εξόρυξη δεδομένων, το προτεινόμενο σύστημα επέδειξε αρκετά μεγαλύτερη ακρίβεια, της τάξης του 99% στο training κομμάτι και 90% στο testing. Για χάριν σύγκρισης, παραθέτουμε τα αντίστοιχα νούμερα ακρίβειας που επέδειξε το CIMDS, ένα ήδη υπάρχον σύστημα εντοπισμού κακόβουλου λογισμικού (και το οποίο είναι αρκετά πιο ακριβές από άλλα παρόμοια συστήματα): 71% και 67% αντίστοιχα.

2.1.2. Εξαγωγή των API κλήσεων των Windows

Από τις πιο συνηθισμένες τεχνικές που ακολουθούν οι δημιουργοί κακόβουλου λογισμικού είναι η χρήση εργαλείων απόκρυψής του[4], όπως είναι ο πολυμορφισμός (polymorphism) και ο μεταμορφισμός (metamorphism) του κώδικα, που στόχο έχουν την αποφυγή ανίχνευσής του από τα διάφορα antivirus προγράμματα. Είναι συνήθης πρακτική η στατική εξαγωγή του κρυμμένου αυτού ωφέλιμου φορτίου (το μέρος του κώδικα που είναι υπεύθυνο για την κακόβουλη συμπεριφορά) και η μετέπειτα ανάλυση των API κλήσεων για ανίχνευση πιθανού malware, κάτι που όμως απαιτεί πολύ χρόνο και καλή γνώση προγραμματισμού χαμηλού επιπέδου (επιπέδου πυρήνα-kernel και Assembly γλώσσας). Οι M. Alazab, S. Venkataraman και P. Watters στο έργο τους [2] παρουσιάζουν μία μεθοδολογία τεσσάρων βημάτων για την υλοποίηση ενός πλήρους αυτοματοποιημένου τρόπου εξαγωγής και ανάλυσης των API κλήσεων και μετέπειτα διαχωρισμού τους σε έξι κύριες κατηγορίες ύποπτης συμπεριφοράς.

Ανήκοντας στην κατηγορία των στατικών τεχνικών εντοπισμού βασισμένων στην συμπεριφορά, η τεχνική αυτή προσπαθεί να ξεχωρίσει διάφορα δυναμικά χαρακτηριστικά των προγραμμάτων σε απομονωμένο περιβάλλον. Πιο συγκεκριμένα, χρησιμοποιήθηκαν 386 δείγματα κακόβουλων αρχείων, η ονομασία των οποίων βασίστηκε στη μοναδική τιμή που τους δόθηκε από τον MD5 αλγόριθμο[5]. Μεταξύ Ιουλίου και Νοεμβρίου 2009, συλλέχθηκαν δείγματα κακόβουλου λογισμικού από το έργο Honeynet, καθώς και από άλλες πηγές. Έγινε χρήση του ειδικού εργαλείου IDAPro disassembler[6] για την «αποσυναρμολόγηση», ανάλυση και εξαγωγή των API κλήσεων από το δυαδικής μορφής περιεχόμενο του malware. Στη συνέχεια, εξετάστηκαν τα προγράμματα, χωρίς, όμως, να εκτελεστούν, με σκοπό την αξιολόγηση της συμπεριφοράς τους κατά την πραγματική εκτέλεση.

Τα 4 βήματα που ακολουθήθηκαν για την αυτοματοποίηση του συστήματος υλοποιήθηκαν σε γλώσσα Python και είναι τα ακόλουθα:

1. Εξαγωγή του malware κώδικα
2. “Αποσυναρμολόγηση” του εκτελέσιμου προγράμματος με σκοπό την ανάκτηση του Assembly κώδικα του προγράμματος
3. Εξαγωγή των API κλήσεων και του κώδικα μηχανής
4. Αντιστοίχιση των API κλήσεων στην MSDN (Microsoft Developer Network) βιβλιοθήκη και ανάλυση της ύποπτης συμπεριφοράς

Αρχικά, χρησιμοποιήθηκε ο ανιχνευτής PEiD[7] στο προαναφερθέν δείγμα των 386 malware εκτελέσιμων αρχείων, όπου βρέθηκε ότι το 77% του δείγματος είχε χρησιμοποιήσει τεχνικές packing (τεχνικές συμπίεσης και κρυπτογράφησης του κώδικα) με σκοπό την αλλαγή της αλληλουχίας των bytes, κάτι που κάνει πιο δύσκολη την ανίχνευσή τους από τα antivirus προγράμματα. Στη συνέχεια, χρησιμοποιήθηκε το εργαλείο IDAPro για την μετάφραση του προγράμματος σε γλώσσα υψηλού επιπέδου, καθώς και λόγω της συμβατότητάς του με Python προγράμματα, με αποτέλεσμα την παραγωγή ενός .idb αρχείου για το κάθε εκτελέσιμο. Το πρόγραμμα αυτόματα δημιούργησε και

[4] Εδώ αναφέρεται το λεγόμενο obfuscation του κώδικα, το οποίο επιτρέπει την κρυπτογράφηση του προγράμματος καθώς και την αξιολόγησή του, χωρίς να είναι απαραίτητη η πρότερη αποκρυπτογράφησή του (όπως είναι συνήθως ο κανόνας), σύμφωνα με τον Mark Zhandry, Recent Developments in Program Obfuscation, Princeton University, September 2016
Ανάκτηση από: https://www.cs.princeton.edu/~mzhandry/2016-Fall-COS597C/ln/ln1.pdf

[5] “Ο MD5 ανήκει στους αλγορίθμους κρυπτογράφησης” και “χρησιμοποιείται συνήθως για την επικύρωση της αυθεντικότητας αρχείων ή ακολουθιών χαρακτήρων”, σύμφωνα με τους Alok Kumar Kasgar, Mukesh Kumar Dhariwal, Neeraj Tantubay και Hina Malviya στο έργο τους *A Review Paper of Message Digest 5 (MD5).*

[6] Disassembler ονομάζεται το λογισμικό που μετατρέπει κώδικα από γλώσσα μηχανής στην Assembly γλώσσα.

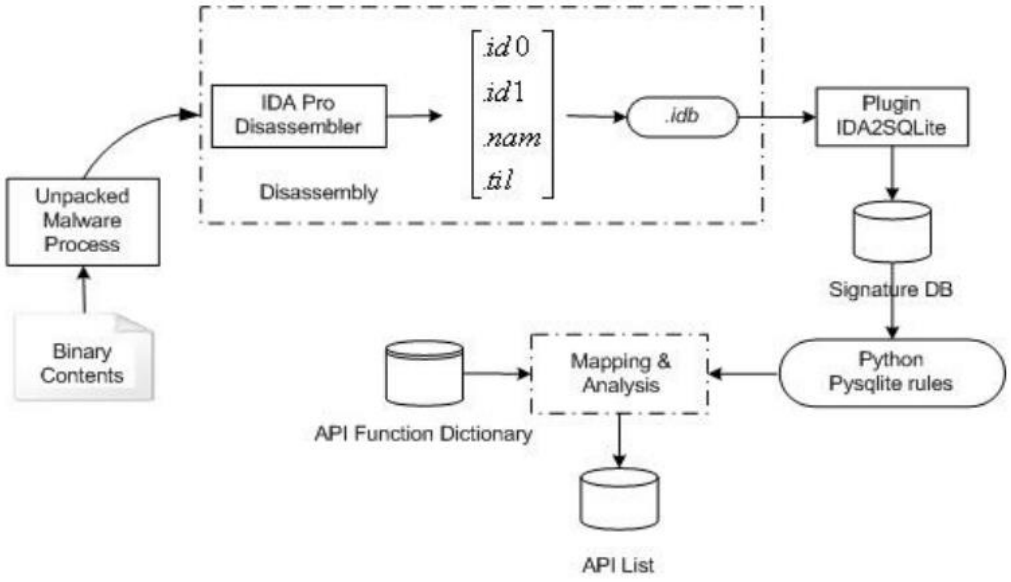
[7] Το PEiD είναι σε θέση να ανιχνεύσει τους περισσότερους συσκευαστές (packers), κρυπτογράφους (cryptors) και μεταγλωττιστές (compilers) για PE (Win32 Portable Executable) αρχεία, που είναι η συνήθης μορφή εκτελέσιμου αρχείου για όλες τις εκδόσεις των Windows σε όλους τους υποστηριζόμενους επεξεργαστές, ενώ ταυτόχρονα μπορεί να ανιχνεύσει πάνω από 470 υπογραφές στα αρχεία αυτά.

έτρεξε κατάλληλο plugin (IDASQLit) για τη χρήση της SQLite[8], με σκοπό τη δημιουργία βάσης δεδομένων. Αναπτύχθηκε ένα σύστημα διεπαφής για πρόσβαση στη βάση, ώστε τα αποτελέσματα από τον assembly κώδικα του κακόβουλου λογισμικού να μπορούν να χρησιμοποιηθούν για καλύτερη ανάλυση του δυαδικού κώδικα. Συγκεκριμένα, το plugin IDASQLit παρήγαγε οκτώ πίνακες, με διαφορετικές πληροφορίες ο καθένας για το δυαδικό κώδικα (μεταξύ των οποίων και οι ακολουθίες των API κλήσεών του). Έπειτα, οι απαραίτητες διαδικασίες υλοποιήθηκαν σε Python για την ανάλυση και αντιστοίχιση των API κλήσεων μεταξύ της MSDN βιβλιοθήκης και του δείγματος από malware που αποθηκεύτηκε στη βάση δεδομένων. Για την ανάλυση της ύποπτης συμπεριφοράς, λήφθηκαν υπόψη χαρακτηριστικά όπως η συχνότητα των κλήσεων, μοτίβα στην ακολουθία των κλήσεων καθώς και οι ενέργειες που εκτελέστηκαν αμέσως μετά ή πριν την κλήση.

Οι έξι κύριες κατηγορίες ύποπτης συμπεριφοράς που προέκυψαν είναι οι εξής:

1. Ψάξιμο αρχείων με σκοπό τη μόλυνσή τους
2. Αντιγραφή / Διαγραφή αρχείων
3. Ανάκτηση πληροφοριών των αρχείων
4. Μετακίνηση αρχείων
5. Ανάγνωση / Επεξεργασία αρχείων
6. Αλλαγή των χαρακτηριστικών των αρχείων

Τέλος, η εικόνα που ακολουθεί, αποτελεί γραφική αναπαράσταση των τεσσάρων βημάτων της διαδικασίας αυτής.



Εικόνα 3 - Αυτοματοποιημένη αρχιτεκτονική για την εξαγωγή API κλήσεων και την ανάλυση της ύποπτης συμπεριφοράς

[8] "Η SQLite είναι μία βιβλιοθήκη λογισμικού που υλοποιεί Transactional SQL μηχανισμό βάσεων δεδομένων χωρίς διακομιστές και με μηδενικές παραμέτρους, ενώ αποτελεί τον πιο ευρέως διαδεδομένο μηχανισμό βάσεων δεδομένων SQL στον κόσμο." Βασισμένο σε απόσπασμα από: https://www.tutorialspoint.com/sqlite/index.htm

2.1.3. Κατηγοριοποίηση Κακόβουλου Λογισμικού Μέσω Δομημένης Ροής Ελέγχου

Ο προσδιορισμός των διάφορων παραλλαγών κακόβουλου λογισμικού μπορεί να αποδειχθεί μεγάλο πλεονέκτημα στον έγκαιρο εντοπισμό και την ανίχνευσή τους. Ένα από τα χαρακτηριστικά που μπορεί να προσδιοριστεί σε διάφορες από τις παραλλαγές αυτές είναι η ροή ελέγχου, καθώς συχνά αυτές ξανά-χρησιμοποιούν κώδικα από νωρίτερες εκδόσεις του malware. Αυτό οδηγεί στην κατηγοριοποίηση κακόβουλου λογισμικού βάσει διαγραμμάτων ροών (flow graphs)[9].

Όπως έχουμε αναφέρει ξανά, ενώ η στατική ανάλυση χρησιμοποιείται ευρέως για σκοπούς κατηγοριοποίησης, εντούτοις μπορεί να μην είναι το ίδιο αποτελεσματική όταν το κακόβουλο λογισμικό έχει περάσει από διαδικασίες μετασχηματισμού του κώδικά του με σκοπό την απόκρυψη του περιεχομένου του. Οι Silvio Cesare και Yang Xiang στο έργο τους [3] προτείνουν έναν νέο αλγόριθμο κατασκευής υπογραφής μέσω δομημένων γραφημάτων ροών ελέγχου, χρησιμοποιώντας τόσο δυναμική όσο και στατική ανάλυση. Πιθανές ομοιότητες μεταξύ δομημένων γραφημάτων μπορούν εύκολα να προσδιοριστούν μέσω της επεξεργασίας αποστάσεων (edit distance)[10] των συμβολοσειρών (strings). Προτείνεται ένας γρήγορος εξομοιωτής (επιπέδου εφαρμογής και όχι ολόκληρου συστήματος, για λόγους επίδοσης) για να αντιστρέψει πιθανό μετασχηματισμό (packing) του κώδικα, ενώ, για σκοπούς επίδειξης της αποτελεσματικότητάς του, καθώς και της κατηγοριοποίησης μέσω διαγραμμάτων ροών (flow graphs), αξιολογήθηκε τόσο σε συνθετικό όσο και σε πραγματικό δείγμα malware. Η αξιολόγηση έδειξε ότι το αυτοματοποιημένο αυτό σύστημα είναι αποτελεσματικό τόσο στην αποκάλυψη κρυμμένου κώδικα και στους χρόνους εκτέλεσης που χρειάζεται για να το πετύχει, όσο και ακριβές στην κατηγοριοποίηση.

Πιο αναλυτικά, το σύστημα θεωρείται ότι έχει πρόσβαση σε ένα σύνολο γνωστών κακόβουλων προγραμμάτων, το οποίο και θα βοηθήσει στην κατασκευή μίας αρχικής βάσης δεδομένων υπογραφών κακόβουλου λογισμικού. Η βάση αυτή κατασκευάζεται μέσω της αναγνώρισης των σταθερών χαρακτηριστικών του κάθε κακόβουλου λογισμικού και της δημιουργίας μίας σχετικής υπογραφής που αποθηκεύεται στη βάση με τη μορφή δομημένου γραφήματος. Το σύστημα δέχεται σαν είσοδο ένα προηγουμένως άγνωστο δυαδικό σύστημα, το οποίο πρόκειται να ταξινομηθεί ως κακόβουλο ή μη. Η δυαδική αυτή είσοδος, όπως και τα κακόβουλα λογισμικά που χρησιμοποιήθηκαν κατά την αρχικοποίηση, μπορεί να έχει υποστεί μετασχηματισμό του κώδικα, στην οποία περίπτωση επιστρατεύεται η χρήση εξομοιωτή (που προσφέρει ένα ασφαλές και απομονωμένο περιβάλλον) για την αποκάλυψη του κρυμμένου κώδικα (δυναμική ανάλυση), η παύση της λειτουργίας του οποίου αποφασίζεται χρησιμοποιώντας ανάλυση εντροπίας[11].

Στη συνέχεια, η στατική ανάλυση χρησιμοποιεί μεθόδους δόμησης (structuring)[12] για τον προσδιορισμό των χαρακτηριστικών και την κατασκευή υπογραφών για τα διαγράμματα ροών ελέγχου. Κάθε μία υπογραφή αναπαρίσταται με τη μορφή συμβολοσειράς (string) και, μέσω της επεξεργασίας απόστασης (edit distance) που αναφέρθηκε παραπάνω, βρίσκονται κατά προσέγγιση

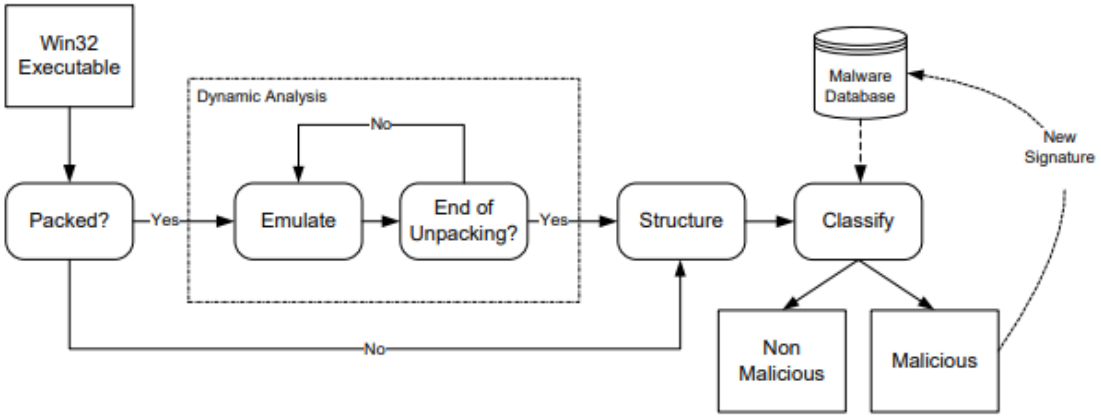
[9] "Στην επιστήμη των υπολογιστών, ένα γράφημα ροής ελέγχου (CFG – Control Flow Graph) είναι η γραφική αναπαράσταση της ροής ελέγχου ή των υπολογισμών που πραγματοποιούνται κατά τη διάρκεια εκτέλεσης προγραμμάτων ή εφαρμογών. Τα γραφήματα αυτά χρησιμοποιούνται, ως επί το πλείστο, στον τομέα της στατικής ανάλυσης, όπως και σε εφαρμογές μεταγλωττιστών, καθώς μπορούν με ακρίβεια να αναπαραστήσουν τη ροή ενός προγράμματος."
Βασισμένο σε απόσπασμα από: https://www.techopedia.com/definition/6426/control-flow-graph-cfg

[10] "Levenshtein ή edit distance ονομάζεται ο μικρότερος αριθμός εισαγωγών, διαγραφών ή αντικαταστάσεων χαρακτήρων (που χρειάζονται) ώστε να γίνουν δύο συμβολοσειρές (π.χ. λέξεις) ίσες.", σύμφωνα με τον Gonzalo Navarro στο έργο του *A Guided Tour to Approximate String Matching*, University of Chile, Μάρτιος 2001

[11] Η εντροπία περιγράφει το μέγεθος της πληροφορίας που περιλαμβάνεται σε ένα σύνολο από δεδομένα. Συμπιεσμένα και κρυπτογραφημένα δεδομένα έχουν σχετικά μεγάλο βαθμό εντροπίας, σε αντίθεση με τα κανονικά δεδομένα και προγράμματα, οπότε μεγάλος βαθμός εντροπίας σε συνεχόμενα κομμάτια κώδικα συνήθως υποδεικνύει μετασχηματισμό του κώδικα (packing) και, κατά συνέπεια, κακόβουλο λογισμικό.

[12] Δόμηση ή structuring περιγράφεται ως η διαδικασία της αποσυμπίλισης (decompiling) μη δομημένων ροών ελέγχου σε υψηλότερου επιπέδου δομές, όπως ο πηγαίος κώδικας, στις οποίες επίσης αποτυπώνονται πιθανές συνθήκες και επαναλήψεις που περιλαμβάνονται στον αρχικό κώδικα.

αντιστοιχίες με τα διαγράμματα των γνωστών κακόβουλων λογισμικών που βρίσκονται στη βάση δεδομένων. Η ομοιότητα μεταξύ αυτών μετριέται ως ένας πραγματικός αριθμός από το 0 έως το 1 - με το 0 να σημαίνει ότι δεν υπάρχει καμία ομοιότητα και το 1 να σημαίνει ταυτοποίηση. Αν ο αριθμός αυτός για συνολικά για ολόκληρο το εξεταζόμενο πρόγραμμα γίνει ίσος ή υπερβεί ένα συγκεκριμένο όριο (στην προκειμένη περίπτωση το όριο αυτό έχει τεθεί να είναι ίσο με 0,6) για κάποιο από τα κακόβουλα προγράμματα της βάσης δεδομένων, τότε το προς εξέταση δυαδικό αρχείο εισόδου θεωρείται μια παραλλαγή του συγκεκριμένου malware και, συνεπώς, κακόβουλο. Σε αυτήν την περίπτωση, η βάση δεδομένων μπορεί να ενημερωθεί για να συμπεριλάβει το δυνητικά νέο σύνολο παραγόμενων υπογραφών που σχετίζονται με την παραλλαγή αυτή. Η παρακάτω εικόνα αποτελεί γραφική αναπαράσταση του συστήματος που περιγράφθηκε παραπάνω.



Εικόνα 4 - Σχηματικό διάγραμμα του συστήματος κατηγοριοποίησης κακόβουλου λογισμικού

Τα τεχνικά χαρακτηριστικά του επιτραπέζιου υπολογιστή (desktop) που χρησιμοποιήθηκε για την αξιολόγηση της απόδοσης του συστήματος ήταν: 4GB μνήμη, 4πύρηνος επεξεργαστής των 2.4GHz και λειτουργικό σύστημα Windows Vista Home Premium με Service Pack 1. Το σύστημα επιβεβαιώθηκε ότι εκτελεί σωστά την εξαγωγή του κρυφού κώδικα, αφού ελέγχθηκε έναντι 14 δημόσιων εργαλείων συσκευασίας (packing)[13]. Τα προγράμματα που επιλέχθηκαν για να υποστούν το μετασχηματισμό του κώδικά τους ήταν τα hostname.exe και calc.exe του Microsoft Windows XP λειτουργικού συστήματος. Τα δείγματα κακόβουλου λογισμικού που χρησιμοποιήθηκαν επιλέχθηκαν κατάλληλα ώστε να μιμούνται κακόβουλα λογισμικά από προηγούμενη έρευνα (Carrera και Erdélyi, 2004)[14], πολλά από τα οποία είχαν υποστεί μετασχηματισμό του κώδικα (packing). Τέλος, παραλλαγές malware από τις Netsky, Klez και Roron οικογένειες κακόβουλου λογισμικού αποκτήθηκαν από δημόσιες βάσεις δεδομένων.

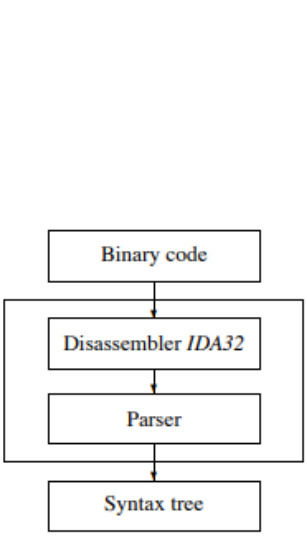
[13] Τα εργαλεία αυτά εκτελούν διάφορες τεχνικές για να επιτύχουν απόκρυψη του κώδικα: συμπίεση, κρυπτογράφηση, obfuscation, καθώς και εντοπισμό σφαλμάτων (debugger detection).

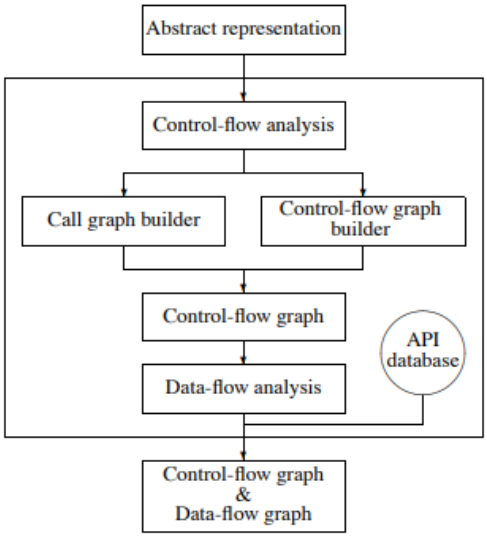
[14] *Digital genome mapping-advanced binary malware analysis*, E. Carrera & G. Erdélyi, Virus Bulletin Conference, 2004, 187-197

2.1.4. Στατικός Εντοπισμός Κακόβουλου Λογισμικού σε Εκτελέσιμα Προγράμματα

Οι J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui Y. Lavoie και N. Tawbi στο έργο τους [4] προτείνουν μία μέθοδο ανάλυσης της κακόβουλης πρόθεσης ενός εκτελέσιμου αρχείου πριν αυτό εκτελεστεί. Αυτή η μορφή εντοπισμού κακόβουλου λογισμικού κάνει χρήση της στατικής ανάλυσης για τον προσδιορισμό του κακόβουλου κώδικα. Η προσέγγιση αυτή επιτυγχάνεται σε τρία βήματα, τα οποία θα αναλυθούν περαιτέρω στην πορεία: 1) κατασκευή μίας αφηρημένης ενδιάμεσης αναπαράστασης, 2) ανάλυση βασισμένη στα διαγράμματα ροής, η οποία αποτυπώνει τη συμπεριφορά του προγράμματος σχετικά με θέματα ασφαλείας και, τέλος, 3) στατικός έλεγχος των κρίσιμων συμπεριφορών έναντι στις πολιτικές ασφαλείας.

Αρχικά, το δυαδικό εκτελέσιμο αποσυναρμολογείται σε κώδικα assembly, μέσω της χρήσης του εργαλείου IDA32 Pro. Ο κώδικας αυτός, στη συνέχεια, αναλύεται για να παραγάγει ένα συντακτικό δένδρο (Εικόνα 4.1), από το οποίο δένδρο δημιουργούνται ένα γράφημα ροής ελέγχου[15] και ένα γράφημα ροής δεδομένων[16] (Εικόνα 4.2).



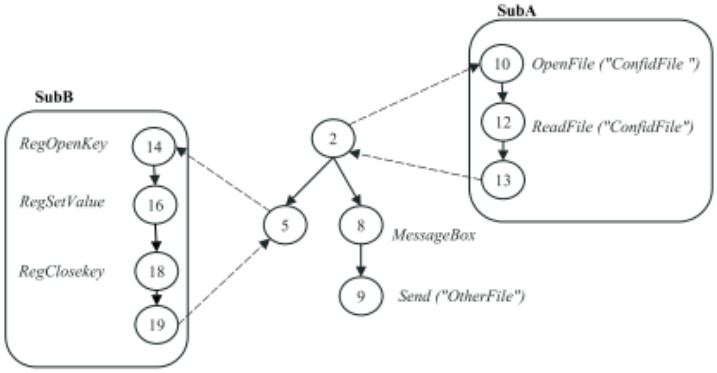


Εικόνα 4.1 - Παραγωγή της ενδιάμεσης αναπαράστασης Εικόνα 4.2 – Ανάλυση γραφημάτων

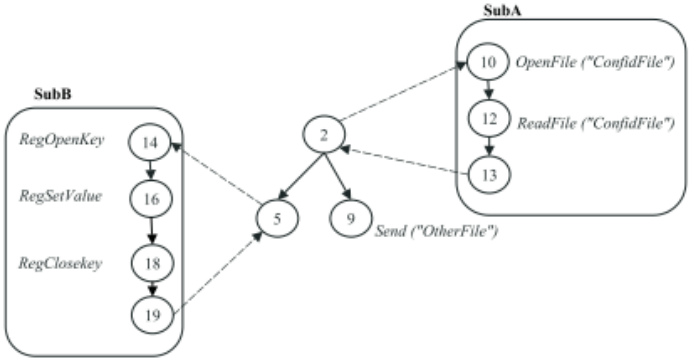
Στη συνέχεια, δημιουργείται ένα API-γράφημα στο οποίο αναπαρίστανται μόνο οι API κλήσεις και όχι άλλες εντολές (Εικόνα 4.3). Από αυτό το API-γράφημα δημιουργείται ένα κρίσιμο API-γράφημα (Εικόνα 4,4), όπου, επειδή η κρισιμότητα εξαρτάται από την εκάστοτε εφαρμογή, ο χρήστης αποφασίζει ποιες από τις API κλήσεις θεωρούνται κρίσιμες βάσει πολιτικών ασφαλείας. Οι πολιτικές αυτές ασφαλείας αποτελούν ένα σύνολο κανόνων που χαρακτηρίζουν ως αποδεκτές ή μη αποδεκτές τις εκτελέσεις ενός προγράμματος και εκπροσωπούνται από αυτόματα, επίσης αναφερόμενα και ως αυτόματα ασφαλείας.

[15] Ένα γράφημα ροής ελέγχου (control-flow graph) είναι ένα κατευθυνόμενο γράφημα, όπου σε κάθε κόμβο αντιστοιχεί μία εντολή ή ένα σύνολο εντολών του προγράμματος. Η ακμή μεταξύ δύο κόμβων αντιπροσωπεύει την απευθείας ροή ελέγχου μεταξύ τους.

[16] Ένα γράφημα ροής δεδομένων (data-flow graph) είναι ένα κατευθυνόμενο γράφημα, όπου οι κόμβοι αντιπροσωπεύουν λειτουργίες του προγράμματος ενώ οι ακμές αντιπροσωπεύουν διαφόρου τύπου πληροφορίες μεταξύ αυτών.



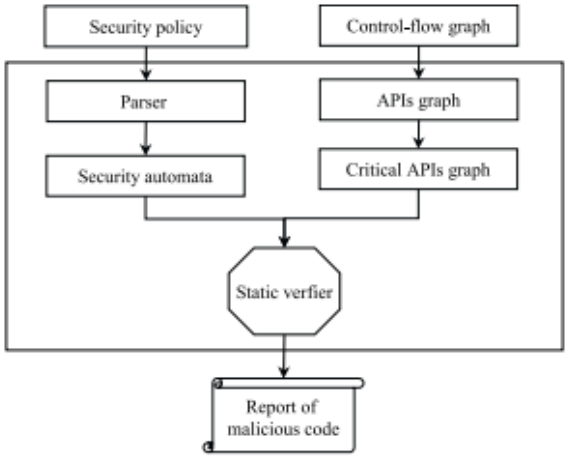
Εικόνα 4.3 - API-γράφημα



Εικόνα 4.4 - Κρίσιμο API-γράφημα

Οι μεταβάσεις στο αυτόματο ασφαλείας συμβαίνουν όταν το κεντρικό σύστημα εκτελεί κάποια ενέργεια. Τουλάχιστον μία από τις καταστάσεις του αυτόματου ασφαλείας πρέπει να αναγνωρίζεται ως κακή κατάσταση. Για παράδειγμα, αφού το σύστημα διαβάσει ένα αρχείο προς εξέταση, το αυτόματο ασφαλείας μπορεί να ορίσει ότι η διαδικασία δεν μπορεί να χρησιμοποιήσει υπηρεσίες δικτύου, όπως η εντολή send(). Αν το σύστημα εισέλθει σε κακή κατάσταση βάσει του κρίσιμου API-γραφήματος του προγράμματος, τότε το σύστημα παραβιάζει τις πολιτικές ασφαλείας που προσδιορίζονται από το αυτόματο και, άρα, το εκτελέσιμο θεωρείται κακόβουλο. Επομένως, το κρίσιμο API-γράφημα ελέγχεται από το αυτόματο ασφαλείας με σκοπό να προσδιοριστεί η ύπαρξη ύποπτης συμπεριφοράς (Εικόνα 4.5).

Τέλος, να σημειωθεί ότι ένα δείγμα κακόβουλου λογισμικού στο οποίο εξετάστηκε η προσέγγιση αυτή με επιτυχία ήταν το πρόγραμμα WINIPX.exe, παραγόμενο από τον Win32/Semisoft ιό.

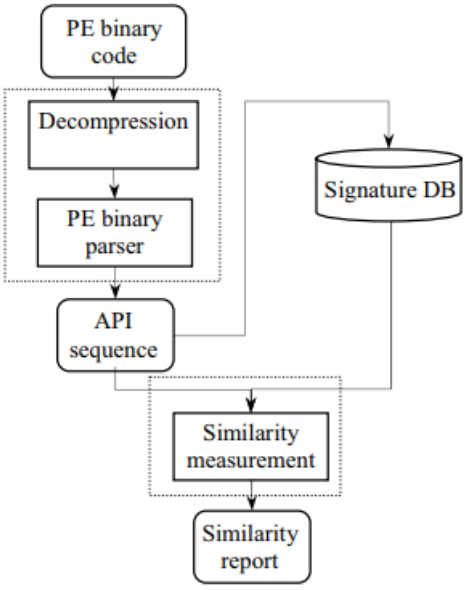


Εικόνα 4.5 - Στατική επαλήθευση

2.1.5. Στατικός Αναλυτής Κακόβουλων Εκτελέσιμων (Static Analyzer of Vicious Executables - SAVE)

Οι A. H. Sung, P. Chavez και S. Mukkamala στο έργο τους [5] προτείνουν μία καινούρια προσέγγιση ονόματι SAVE (συντομογραφία των όρων Static Analyzer of Vicious Executables), η οποία επικεντρώνεται κυρίως σε πολυμορφικό ή μεταμορφικό κακόβουλο λογισμικό. Η υπόθεση που ακολουθείται είναι ότι όλες οι εκδόσεις του ίδιου κακόβουλου λογισμικού μοιράζονται μία κοινή, βασική υπογραφή, η οποία αποτελεί συνδυασμό πολλών χαρακτηριστικών του κώδικα. Όταν κάποιο κακόβουλο λογισμικό αναγνωριστεί για πρώτη φορά, μπορεί να αναλυθεί ώστε να αποκτηθεί η υπογραφή του, η οποία παρέχει τη βάση για τον εντοπισμό των παραλλαγών του ίδιου αυτού malware στο μέλλον.

Αρχικά, εισάγεται ένα εκτελέσιμο πρόγραμμα με κώδικα σε δυαδική μορφή, το οποίο αναλύεται σε μια ακολουθία από API κλήσεις (ενδιάμεση αναπαράσταση). Κάθε κλήση API αντιπροσωπεύεται από έναν ακέραιο αριθμό των 32-bit. Τα 16 πιο σημαντικά bits του αριθμού αντιπροσωπεύουν μία συγκεκριμένη DLL (Dynamic-link Library), ενώ τα 16 λιγότερο σημαντικά bits αντιπροσωπεύουν μία συγκεκριμένη API κλήση σε αυτή τη Library. Οπότε η ακολουθία των API κλήσεων είναι μία ακολουθία αριθμών, η οποία λειτουργεί ως η υπογραφή του εκτελέσιμου. Χρησιμοποιείται μία βάση δεδομένων, όπου είναι αποθηκευμένες οι αντίστοιχες υπογραφές γνωστών κακόβουλων λογισμικών. Υπολογίζεται η απόσταση μεταξύ των υπογραφών της βάσης και της αντίστοιχης υπογραφής του υπό εξέταση προγράμματος. Για το σκοπό αυτό χρησιμοποιούνται συνολικά τρεις συναρτήσεις ομοιότητας[17], καθώς καμία από αυτές δεν μπορεί να δώσει από μόνη της τα καλύτερα δυνατά αποτελέσματα για όλες τις μετρήσεις ακολουθιών. Ο μέσος όρος αυτών δίνει τη συνολική ομοιότητα μεταξύ της API ακολουθίας του εξεταζόμενου προγράμματος και καθενός από τις γνωστές υπογραφές της βάσης δεδομένων. Εάν η διαφορά είναι μικρότερη ή ίση του 10%, τότε το πρόγραμμα θεωρείται κακόβουλο.



Εικόνα 5 - Τεχνική SAVE

Οι συγγραφείς συνέκριναν τη μέθοδο SAVE με 8 προγράμματα ανίχνευσης κακόβουλου λογισμικού: NORTON, McAfee Unix Scanner, McAfee, Dr. Web, Panda, Kaspersky, F-Secure και Anti Ghostbusters. Όλοι αυτά το προγράμματα δοκιμάστηκαν απέναντι σε παραλλαγές των ιών W32.Mydoom, W32.Bika, W32.Beagle και W32.Blaster.Worm, οι οποίες είχαν υποστεί πολυμορφισμό του κώδικα, και η μέθοδος SAVE ήταν η μόνη τεχνική που κατάφερε να εντοπίσει όλες τις παραλλαγές των προαναφερθέντων κακόβουλων λογισμικών.

[17] Οι συναρτήσεις αυτές είναι οι εξής: μέτρηση συνημίτονου (Cosine measure), εκτεταμένη μέτρηση Jaccard (extended Jaccard measure) και συσχέτιση Pearson (Pearson correlation measure).

2.1.6. Ανίχνευση Εισαγόμενου, Δυναμικά Παραγόμενου και Αποκρυμμένου Κακόβουλου Κώδικα

Οι Jesse C. Rabek, Roger I. Khazan, Scott M. Lewandowski και Robert K. Cunningham στο έργο τους [6] προτείνουν μία τεχνική ονόματι DOME, η οποία αποτελεί συντομογραφία των όρων Detection of Malicious Executables. Το DOME σχεδιάστηκε να εντοπίζει εισαγόμενο[18], δυναμικά παραγόμενο[19], καθώς και αποκρυμμένο[20] κώδικα, ενώ έγιναν κάποιες παραδοχές σχετικά με τη λειτουργία του:

- Οποιοσδήποτε εισαγόμενος, δυναμικά παραγόμενος ή αποκρυμμένος κώδικας θεωρείται ότι είναι κακόβουλος.
- Ο κακόβουλος κώδικας αλληλοεπιδρά άμεσα με το λειτουργικό σύστημα[21].
- Στην αλληλεπίδραση αυτή, ο κώδικας χρησιμοποιεί Win32 API κλήσεις.
- Όταν ο κώδικας κρύβει τον εαυτό του από την ανίχνευση και την ανάλυση χρησιμοποιώντας δυναμικά παραγόμενο κώδικα ή τεχνικές απόκρυψης αυτού, η χρήση των Win32 API του είναι επίσης αποκρυμμένη[22].
- Τα εκτελέσιμα που είναι προς προστασία μπορούν με επιτυχία να αποσυναρμολογηθούν και οι APIs που χρησιμοποιούνται από αυτά να εποπτευθούν κατά τη διάρκεια της εκτέλεσης.

Η τεχνική χαρακτηρίζεται από δύο βήματα. Στο πρώτο βήμα, γίνεται στατική προ-επεξεργασία του προγράμματος προς εξέταση, με τη χρήση του εργαλείου IDA Pro disassembler. Η προ-επεξεργασία αυτή αποτελείται από:

1. την αποθήκευση των διευθύνσεων των κλήσεων του συστήματος,
2. τα ονόματα αυτών και
3. τη διεύθυνση της εντολής που απευθείας ακολουθεί την κάθε κλήση, ή αλλιώς, τη διεύθυνση επιστροφής των κλήσεων του συστήματος στο εκτελέσιμο.

Σε περίπτωση ενημέρωσης των εκτελέσιμων, το βήμα της προ-επεξεργασίας θα πρέπει να επαναληφθεί. Στο δεύτερο βήμα, όπου γίνεται χρήση του Detours wrapper πακέτου, το DOME παρακολουθεί το εκτελέσιμο κατά τη διάρκεια εκτέλεσής του, διασφαλίζοντας ότι όλες οι κλήσεις συστήματος που πραγματοποιούνται κατά την εκτέλεση ταιριάζουν με αυτές που έχουν καταγραφεί από τη στατική ανάλυση του πρώτου βήματος. Σε περίπτωση αστοχίας, θεωρείται ότι ανιχνεύθηκε κακόβουλος κώδικας και το σύστημα μπλοκάρει την API κλήση. Ο εντοπισμός γίνεται πριν ο κακόβουλος κώδικας έχει την ευκαιρία να αλληλοεπιδράσει με το λειτουργικό σύστημα, δηλαδή πριν αυτό έχει πρόσβαση στα προστατευμένα στοιχεία του λειτουργικού συστήματος, όπως οι φάκελοι και τα αρχεία. Στην έρευνα που διεξήχθη από τους συγγραφείς, διαπιστώθηκε ότι η τεχνική DOME ήταν σε θέση να ανιχνεύσει όλες τις κλήσεις του συστήματος που έγιναν από τον κακόβουλο κώδικα, ενώ στο μέλλον θα προσπαθήσουν να επεκτείνουν το μοντέλο, ώστε να είναι σε θέση να αντιμετωπίσει και απειλές που προσπαθούν να το παρακάμψουν.

[18] Κώδικας που εισάγεται στο χώρο διευθύνσεων μιας διεργασίας κατά το χρόνο εκτέλεσης.

[19] Κώδικας που δημιουργείται από μια διεργασία κατά το χρόνο εκτέλεσης.

[20] Κώδικας που υπάρχει ήδη στον αρχικό κώδικα, αλλά που οι πραγματικές προθέσεις του είναι κρυμμένες πίσω από πολύπλοκους υπολογισμούς και παραποίηση δεδομένων.

[21] Εξαιρούνται περιπτώσεις που δεν είναι απαραίτητη η άμεση αλληλεπίδραση με το λειτουργικό, όπως σε περιπτώσεις αλλοίωσης δεδομένων ή επιθέσεις άρνησης υπηρεσιών (denial of service attacks).

[22] Από τη στιγμή που οι API κλήσεις του αποτελούν την ουσία του τι κάνει και πώς λειτουργεί ο κακόβουλος κώδικας, είναι λογικό να αποκρύπτεται και η δική τους χρήση.

2.1.7. Εργαλείο για την Ανάλυση και Ανίχνευση Κινητού Κακόβουλου Κώδικα

Οι Akira Mori, Tomonori Izumida, Toshimi Sawada και Tadashi Inoue στο έργο τους [7] προτείνουν ένα εργαλείο για την ανίχνευση κινητών αυτό-κρυπτογραφημένων[23] και πολυμορφικών[24] ιών και σκουληκιών υπολογιστή. Οι παραδοσιακές τεχνικές αντιστοίχισης μοτίβων δεν μπορούν να ανιχνεύσουν αυτούς τους τύπους ιών, που έχουν σχεδιαστεί να τις παρακάμπτουν, καθώς, μέσω της αυτό-κρυπτογράφησης, δεν παρουσιάζουν τα μοτίβα στα οποία βασίζονται οι τεχνικές αυτές. Ως εκ τούτου, οι συγγραφείς δημιούργησαν αυτό το εργαλείο για την αντιμετώπιση του ζητήματος αυτού, που επικεντρώνεται σε Win32 εκτελέσιμα αρχεία, που ενεργοποιούνται σε επεξεργαστή IA32.

Η διαδικασία ανάλυσης περιλαμβάνει την αποκρυπτογράφηση του κώδικα σε εξομοιωτή, τη μετέπειτα στατική ανάλυση αυτού και την προσομοίωση των API λειτουργιών. Αρχικά, ο κώδικας αποκρυπτογραφείται σε έναν προσομοιωτής[25] του λειτουργικού συστήματος. Με αυτόν τον τρόπο, τίθεται δυνατή η ανάλυση της συμπεριφοράς του κατά τη διάρκεια εκτέλεσης, χωρίς να χρειαστεί να ενεργοποιηθεί ο ιός σε κανονικό περιβάλλον. Χρησιμοποιείται επίσης εξομοιωτής εκτέλεσης του λειτουργικού συστήματος, με σκοπό τη συλλογή και διατήρηση απαραίτητων για τη μετέπειτα ανάλυση πληροφοριών, όπως καταχωρητές, περιβαλλοντικές μεταβλητές[26], αρχεία και φάκελοι που τροποποιούνται ή δημιουργούνται από τις API κλήσεις κτλ.

Μόλις ο ιός αποκρυπτογραφήσει το φορτίο του, εκτελείται στατική ανάλυση σε αυτόν για τον εντοπισμό των κλήσεων του συστήματος που προέρχονται από αυτό. Αρχικά, εκτελείται ανάλυση της ροής ελέγχου για την ανάγνωση των οδηγιών προς τις API κλήσεις που πρέπει να ελεγχθούν και, στη συνέχεια, ο προσομοιωτής ακολουθεί τα μονοπάτια εκτέλεσης που περιέχουν αυτές οι API κλήσεις.

Οι πολιτικές ανίχνευσης που χρησιμοποιούνται για τον έλεγχο των API κλήσεων καθορίζονται από το χρήστη και περιγράφουν σενάρια (όπως η μόλυνση φακέλων) στο επίπεδο των API κλήσεων. Είναι διαμορφωμένες ως αυτόματα καταστάσεων που αντιπροσωπεύουν την κακόβουλη συμπεριφορά, με την κάθε μετάβαση να συμβαίνει όταν καλούνται συγκεκριμένες API λειτουργίες κάτω από συγκεκριμένες συνθήκες. Από τη γλώσσα των αυτομάτων γράφονται σε C++, μεταγλωττίζονται σε βιβλιοθήκες κατά το χρόνο εκτέλεσης, φορτώνονται στο εργαλείο και, μετά, καλούνται από εικονικές λειτουργίες για να οδηγήσουν το αυτόματο μετάβασης καταστάσεων. Βασισμένες σε αυτές τις περιγραφές των πολιτικών ανίχνευσης, το εργαλείο ελέγχει τα αποτελέσματα της εξομοίωσης του κώδικα, της στατικής του ανάλυσης και της προσομοίωσης του λειτουργικού συστήματος για να αποφασίσει κατά πόσο καθορισμένοι τύποι συμπεριφορών αναγνωρίζονται μέσα στον κώδικα προς εξέταση. Όταν το πρόγραμμα προς εξέταση φτάσει σε μία τελική κατάσταση «άρνησης» (“denial”), τότε θεωρείται κακόβουλο. Οι βασικότερες πολιτικές που εφαρμόστηκαν στο εργαλείο είναι οι ακόλουθες[27]:

[23] Ένας αυτό-κρυπτογραφημένος ιός αποτελείται από το κρυπτογραφημένο φορτίο του και τον απαιτούμενο κώδικα για αποκρυπτογράφηση, που χρησιμοποιεί μόλις βρεθεί στη μνήμη του συστήματος. Από τη στιγμή που το κακόβουλο μέρος του είναι κρυπτογραφημένο, η συμπεριφορά του ιού δεν μπορεί να προσδιοριστεί.

[24] Ο πολυμορφικός ιός αποτελεί βελτιωμένη έκδοση του αυτό-κρυπτογραφημένου ιού. Ο ιός αυτός προσπαθεί να εφαρμόζει διαφορετικό κώδικα αποκρυπτογράφησης κάθε φορά που ενεργοποιείται, μέσω αλλαγών στη μέθοδο κρυπτογράφησής του.

[25] Ο προσομοιωτής μιμείται με ακρίβεια αλλαγές στην εσωτερική δομή ενός IA32 επεξεργαστή (π.χ. αλλαγές στη μνήμη, στους καταχωρητές–registers κτλ.) μέσω της εκτέλεσης οδηγιών μηχανής. Είναι λειτουργικός τόσο σε Windows όσο και σε Linux συστήματα και περιέχει τις βασικές λειτουργίες που χρειάζονται για εντοπισμού σφαλμάτων (debugging).

[26] Οι περιβαλλοντικές μεταβλητές βοηθούν τα προγράμματα στο να γνωρίζουν το φάκελο που θα εγκατασταθούν αρχεία, πού να αποθηκευτούν προσωρινά αρχεία και πού βρίσκονται οι ρυθμίσεις του συγκεκριμένου προφίλ χρήστη. Με λίγα λόγια, βοηθούν στη διαμόρφωση του περιβάλλοντος στο οποίο τρέχουν τα προγράμματα στον υπολογιστή.

[27] Να σημειωθεί ότι διατηρήθηκε η αγγλική τους ονομασία για λόγους πιστότητας.

- **Mass email policy:** Ελέγχεται η περίπτωση να αποκτήσει το πρόγραμμα μία διεύθυνση διακομιστή SMTP και πολλές τυχαίες διευθύνσεις προορισμού με σκοπό την αποστολή ενός μηνύματος. Εκτελείται προσομοίωση χρησιμοποιώντας το SMTP πρωτόκολλο για να αποφασιστεί κατά πόσο όντως συμβαίνει αυτή η περίπτωση.
- **Registry modification policy:** Γίνεται έλεγχος για οποιαδήποτε τροποποίηση στις ρυθμίσεις του συστήματος, ειδικά στα κλειδιά των καταχωρητών (registry keys), όπου είναι καταχωρημένα τα προγράμματα αυτό-εκκίνησης.
- **File modification policy:** Γίνεται έλεγχος για οποιαδήποτε τροποποίηση σε αρχεία ή φακέλους που δεν έχουν δικαιώματα εγγραφής.
- **File infection policy:** Ελέγχεται το κατά πόσο το πρόγραμμα υπό εξέταση γράφεται σε φακέλους. Ουσιαστικά περιγράφεται το σενάριο κατά το οποίο το πρόγραμμα αποκτά ένα χειριστή φακέλων (file handler) σαρώνοντας τους καταλόγους, το τροποποιεί στη μνήμη και γράφει ξανά πίσω στο φάκελο.
- **Process scan policy:** Ελέγχεται το σενάριο στο οποίο το πρόγραμμα αποκτά τα IDs μίας διεργασίας μέσα από μία λίστα διεργασιών προς εκτέλεση. Το σενάριο αυτό ορίζεται ως πολιτική καθώς τα συνηθισμένα προγράμματα είναι απίθανο να συμπεριφερθούν κατά αυτόν τον τρόπο.
- **Self-code modification policy:** Ελέγχεται το κατά πόσο το πρόγραμμα τροποποιεί τις επικεφαλίδες (headers), ειδικά αυτές των πινάκων εισαγωγής, στη μνήμη.
- **Anti-debugger / emulation policy:** Ελέγχεται η περίεργη συμπεριφορά απόπειρας εντοπισμού κάποιου debugger (που χρησιμοποιείται για τον εντοπισμό σφαλμάτων) ή εξομοιωτή.
- **Out of bounds execution (OBE) policy:** Ελέγχεται το κατά πόσο το πρόγραμμα υπό εξέταση φθάνει σε διεύθυνση που δε δηλώνεται στην επικεφαλίδα του.
- **External execution policy:** Ελέγχεται το κατά πόσο το πρόγραμμα υπό εξέταση ξεκινά την εκτέλεση ενός εξωτερικού προγράμματος με τις κλήσεις CreateProcess ή ShellExecute.
- **Illegal address call (IAC) policy:** Γίνεται έλεγχος για κλήσεις όπου οι διευθύνσεις μνήμης αποκτήθηκαν μέσω της απευθείας σάρωσης της εικόνας μνήμης των λειτουργιών βιβλιοθήκης.
- **Self-duplication policy:** Ελέγχεται κατά πόσο το πρόγραμμα κάνει αντίγραφα του εαυτού του κατά τη διάρκεια εκτέλεσής του.
- **Network connection policy:** Ελέγχεται κατά πόσο το πρόγραμμα εκτελεί FTP ή HTTP επικοινωνίες. Εδώ, οι διευθύνσεις δεν ελέγχονται, εν αντιθέσει με την πολιτική μαζικών email που είδαμε νωρίτερα (*Mass email policy*). Τέλος, απαιτείται η εξομοίωση των FTP και HTTP πρωτοκόλλων.

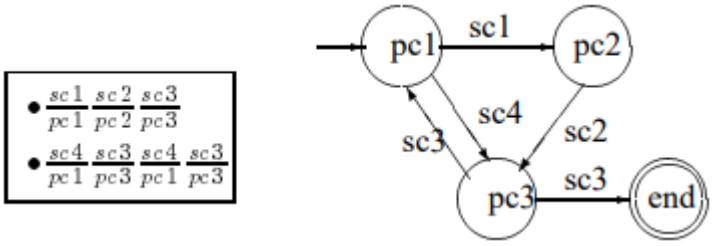
Σε πείραμα που διενεργήθηκε, το εργαλείο αυτό κατάφερε να ανιχνεύσει δείγμα από 600 ιούς / worms. Ειδικότερα, χρησιμοποιώντας τις IAC, OBE self-duplication και file-modification πολιτικές, μπόρεσε να ανιχνεύσει το 95% του δείγματος, ενώ και χωρίς τις δύο πρώτες, επιτεύχθηκε ποσοστό ανίχνευσης πάνω από 80%. Τέλος, να σημειωθεί ότι πάνω από το 80% των ιών του δείγματος δεν είχαν ξανά-αναλυθεί πριν γίνει το πείραμα και, ως εκ τούτου, ήταν άγνωστοι για το εργαλείο.

2.1.8. Εντοπισμός Κακόβουλων Προγραμμάτων με τη χρήση Αυτομάτων Πεπερασμένων Καταστάσεων (FSA)

Οι R. Sekar, M. Bendre, D. Dhurjati και P. Bollineli στο έργο τους [8] δημιούργησαν μία μέθοδο για τον εντοπισμό μη-ομαλής συμπεριφοράς, κάνοντας χρήση αυτομάτου πεπερασμένων καταστάσεων (Finite state Automaton – FSA). Κάθε κόμβος στο FSA αντιπροσωπεύει μία κατάσταση στο υπό εξέταση πρόγραμμα, η οποία αντιστοιχεί στην απόλυτη τιμή που έχει ο μετρητής προγράμματος (program counter[28]) τη στιγμή της κλήσης και την οποία ο αλγόριθμος χρησιμοποιεί για την ταχύτερη εκμάθηση των κανονικών δεδομένων και την καλύτερη ανίχνευση. Οι μεταβάσεις από την μία κατάσταση στην επόμενη γίνονται από τις κλήσεις συστήματος.

Καθώς σε δυναμικώς συνδεδεμένα προγράμματα οι ίδιες λειτουργίες μπορούν να κληθούν σε διαφορετικές τοποθεσίες και αυτό, κατά συνέπεια, καθιστά την τιμή του μετρητή προγράμματος ως μη αξιόπιστη, το αυτόματο είναι αποτελεσματικό μόνο σε στατικώς συνδεδεμένα προγράμματα. Η χρήση σχετικών τιμών έναντι των απολύτων δεν είναι αποτελεσματική, καθώς οι σχετικές τοποθεσίες των λειτουργιών σε δύο διαφορετικές βιβλιοθήκες μπορούν να αλλάξουν από την μία εκτέλεση του προγράμματος στην άλλη.

Το αυτόματο μπορεί να καταγράψει άπειρο πλήθος από κλήσεις συστήματος, μη-σταθερού μήκους, κάτι που συμβάλλει στην πιο ακριβή ανίχνευση κακόβουλου λογισμικού, χρησιμοποιώντας πεπερασμένο αποθηκευτικό χώρο, της τάξης των kilobytes (KB). Οι καταστάσεις του μπορούν να απομνημονεύσουν μικρής και μεγάλης εμβέλειας συσχετισμούς, ενώ το ίδιο το αυτόματο είναι σε θέση να καταγράψει δομές όπως βρόγχους και παρακλάδια στα προγράμματα, κάτι που βοηθάει στην μείωση σφαλμάτων τύπου α (ψευδώς θετικά). Προκειμένου να κατασκευαστεί ο αλγόριθμος εκπαίδευσης, το πρόγραμμα εκτελείται πολλές φορές, κατά τη διάρκεια των οποίων μπορούν εν δυνάμει να προστεθούν επιπλέον καταστάσεις και/ή μεταβάσεις (βλ. Εικόνα 7).



Εικόνα 6 - Δύο ακολουθίες ενός προγράμματος και το παραγόμενο αυτόματο

Όταν μία κλήση συστήματος καλείται, προστίθεται μία νέα μετάβαση στο αυτόματο. Το αυτόματο που προκύπτει από τις πολλαπλές εκτελέσεις θα είναι αυτό που ο αλγόριθμος θεωρεί ως φυσιολογικό. Κατά τη διάρκεια της εκτέλεσης, οι κλήσεις συστήματος αναχαιτίζονται και η κατάσταση του προγράμματος καταγράφεται. Αν εμφανιστεί σφάλμα κατά τη διαδικασία ανάκτησης της τοποθεσίας που έγινε η κλήση, τότε έχει παρουσιαστεί μία ανωμαλία. Ταυτόχρονα, ο αλγόριθμος ελέγχει για κάποια έγκυρη μετάβαση από την τρέχουσα κατάσταση του αυτομάτου στην πιο πρόσφατη κλήση συστήματος που αναχαιτίστηκε. Αν δεν υπάρχει καμία τέτοια μετάβαση, τότε υπάρχει ανωμαλία. Αν τα προηγούμενα δύο βήματα ολοκληρωθούν με επιτυχία, τότε το αυτόματο μεταβαίνει στην επόμενη κατάσταση. Για να εξασφαλιστεί το γεγονός ότι μεμονωμένες αναντιστοιχίες δεν χαρακτηρίζονται άμεσα ως απόπειρες εισβολής, χρησιμοποιείται ένας μετρητής για τη σταδιακή συσσώρευση των ανωμαλιών. Κάθε φορά που ανιχνεύεται μία ανωμαλία, ο μετρητής αυτός αυξάνει ανάλογα με το

[28] Μετρητής προγράμματος (program counter) είναι ένας καταχωρητής (register) σε έναν επεξεργαστή του υπολογιστή που περιλαμβάνει τη διεύθυνση της εντολής που εκτελείται τη δεδομένη χρονική στιγμή. Όταν διαβάζεται μία εντολή από τη μνήμη, ο μετρητής αυξάνει την τιμή του κατά 1 και στη συνέχεια δείχνει στην επόμενη εντολή της ακολουθίας. Όταν γίνεται επανεκκίνηση του υπολογιστή, η τιμή του μετρητή ξεκινάει πάλι από το 0. Δείχνει ουσιαστικά τη σειρά εκτέλεσης της κάθε εντολής στο πρόγραμμα.

βάρος που έχει αποδοθεί στην εκάστοτε ανωμαλία και όταν ξεπεράσει ένα συγκεκριμένο όριο, τότε το σύστημα θεωρεί ότι υπάρχει εισβολή. Ο μετρητής αυτός μειώνεται περιοδικά, το οποίο βοηθά στο να αγνοηθούν μεμονωμένες ανωμαλίες.

Οι συγγραφείς συνέκριναν αυτή τους την προσέγγιση με την προσέγγιση του Hofmeyr και της ομάδας του[29], τη λεγόμενη n-gram[30] προσέγγιση, αξιολογώντας και τις δύο σε httpd, ftpd και nsfd server προγράμματα. Η δικιά τους προσέγγιση φάνηκε να έχει μικρότερο ποσοστό στα σφάλματα τύπου α (ψευδώς θετικά) σε σχέση με την άλλη. Ωστόσο δεν υπάρχουν σαφείς ενδείξεις για το κατά πόσο επιλέχθηκαν οι βέλτιστες παράμετροι για τη n-gram προσέγγιση. Εκτός από τα αποτελέσματά τους, οι συγγραφές ισχυρίζονται, χωρίς εμπειρική γνώση, ότι η τεχνική τους είναι δυνατή απέναντι σε:

- *Buffer overflow attacks* (επιθέσεις υπερχείλισης της προσωρινής μνήμης)
- *Trojan horses και άλλες αλλαγές σε κώδικα*
- *Maliciously crafted input* (κακόβουλα δημιουργημένα δεδομένα εισόδου)
- *Dictionary or Password guessing attacks* (επιθέσεις εικασίας κωδικών)
- *Denial-of-Service attacks*

Εν αντιθέσει, η τεχνική αυτή δεν κρίνεται αποτελεσματική απέναντι σε:

- *Επιθέσεις που περιλαμβάνουν διαφορές στις παραμέτρους των κλήσεων συστήματος* και όχι στην ακολουθία των κλήσεων.
- *Επιθέσεις που δεν αλλάζουν τη συμπεριφορά του επιτιθέμενου προγράμματος,* όπως επιθέσεις που εκμεταλλεύονται λάθη στις ρυθμίσεις του συστήματος ή αδυναμίες των πρωτόκολλων.
- *Επιθέσεις που έχουν γνώση της τεχνικής που χρησιμοποιήθηκε εδώ και του τρόπου που δουλεύει* και τροποποιήθηκαν κατάλληλα για να την αντιμετωπίσουν.

Τέλος, αναφέρονται πιθανές τροποποιήσεις στην τεχνική, όπως

- Η ενσωμάτωση πληροφοριών συχνότητας μαζί με τις μεταβάσεις. Αυτό θα συμβάλλει στην ανίχνευση επιθέσεων που περιλαμβάνουν πολλές μεταβάσεις που, υπό κανονικές συνθήκες, έχουν χαμηλή πιθανότητα εμφάνισης (ειδικά χρήσιμες σε Denial-of-Service επιθέσεις), και
- Η ενσωμάτωση πληροφοριών σχετικά με τις τιμές των παραμέτρων των κλήσεων συστήματος. Αυτό θα επεκτείνει το σύνολο των επιθέσεων που μπορούν να ανιχνευθούν από την τεχνική αυτή, ώστε να περιλαμβάνονται και επιθέσεις που χρησιμοποιούν συμβολικούς συνδέσμους[31].

[29] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, pages 151 – 180, 1998.

[30] Χαρακτηριστικό της συμπεριφοράς των κανονικών προγραμμάτων στην προσέγγιση αυτή είναι οι ακολουθίες των κλήσεων συστήματος που γίνονται, ενώ στα μη-ομαλά προγράμματα η εμφάνιση κλήσεων που δεν έχουν παρατηρηθεί κατά την ομαλή λειτουργία. Οι ακολουθίες αυτές χωρίζονται σε substrings σταθερού μήκους N, τα οποία ονομάζονται N-grams.

[31] Συμβολικός σύνδεσμος (ή symbolic link στα αγγλικά) είναι όρος που χρησιμοποιείται για οποιοδήποτε αρχείο περιλαμβάνει αναφορά σε άλλο αρχείο ή κατάλογο. Λειτουργεί ως εικονικό αρχείο ή φάκελος, που μπορεί να χρησιμοποιηθεί για να συνδεθεί με μεμονωμένα αρχεία ή φακέλους, δίνοντας την εντύπωση ότι είναι αποθηκευμένοι μαζί με το σύνδεσμο, ενώ στην πραγματικότητα ο σύνδεσμος αυτός απλά δείχνει την πραγματική τους τοποθεσία..

2.1.9. Συμπεριφορική Προσέγγιση για τον Εντοπισμό Σκουληκιών Υπολογιστή (Worms)

Οι Daniel R. Ellis, John G. Aiken, Kira S. Attwood και Scott D. Tenaglia στο έργο τους [9] προτείνουν μία προσέγγιση για τον εντοπισμό σκουληκιών (worms), που βασίζεται σε γνωστές κακόβουλες συμπεριφορές. Σε αντίθεση με τις συνηθισμένες τεχνικές που ψάχνουν τις υπογραφές ως κανονικές εκφράσεις στα πακέτα δικτύου, ο τύπος αυτής της προσέγγισης επικεντρώνεται στην ανίχνευση μοτίβων σε πιο αφηρημένο επίπεδο. Ιδανικά, τα μοτίβα αυτά αποτελούν έμφυτες συμπεριφορές των σκουληκιών που εξαπλώνονται και διαφέρουν από την κανονική κυκλοφορία δικτύου. Η συχνότητα και οι αλληλοσυσχετίσεις μεταξύ των συμπεριφορών βελτιώνουν την ακρίβεια της ανίχνευσης, ενώ, για να αποφευχθεί μία συμπεριφορική υπογραφή, απαιτείται αλλαγή στη θεμελιώδη συμπεριφορά του σκουληκιού και όχι μόνο στο διαδικτυακό του ίχνος.

Πιο συγκεκριμένα, μία συμπεριφορική υπογραφή περιγράφει πτυχές της συμπεριφοράς οποιουδήποτε σκουληκιού υπολογιστή που είναι κοινές ανάμεσα σε όλες τις παραλλαγές του και που επεκτείνεται σε χρονική σειρά. Χαρακτηριστικά μοτίβα της συμπεριφοράς των σκουληκιών στην ανταλλαγή δεδομένων μέσω δικτύου ανά πάσα στιγμή περιλαμβάνουν τα εξής:

1. Αποστολή παρόμοιων δεδομένων από τη μία μηχανή στην άλλη
2. Εξάπλωση του ιού σε δενδρική μορφή, και
3. Αλλαγή ενός διακομιστή σε πελάτη[32].

Οι συγγραφείς παρουσιάζουν τρεις διαφορετικές συμπεριφορικές υπογραφές, οι οποίες ονομάζονται υπογραφές βάσης. Οι υπογραφές βάσης (*base signatures*) μπορούν να αναγνωριστούν μέσω της παρακολούθησης των ροών δεδομένων που έρχονται και φεύγουν από ένα μόνο κόμβο (π.χ. server, υπολογιστή κτλ.). Μία τέτοια υπογραφή είναι όταν ο διακομιστής[33] (server) αλλάζει σε πελάτη. Από τη στιγμή που το σκουλήκι πρέπει να εξαπλωθεί, αφού μολύνει το διακομιστή, πρέπει για άλλη μια φορά να ενεργήσει σαν πελάτης σε κάποιον άλλον εξυπηρετητή (internet server) με την ελπίδα να μολύνει ακόμα περισσότερες μηχανές μέσω της ίδιας αδυναμίας. Αυτή η προσέγγιση δεν είναι το ίδιο αποτελεσματική αν εφαρμοστεί σε περιβάλλον peer-to-peer[34].

Μία άλλη υπογραφή βάσης είναι η λεγόμενη *alpha-in/alpha-out*. Αυτή η υπογραφή λέει απλά ότι τα σκουλήκια υπολογιστών τυπικά στέλνουν παρόμοια δεδομένα μεταξύ των κόμβων και, επομένως, έχουν παρόμοιους, αν όχι τους ίδιους, συνδέσμους ροής δεδομένων εισόδου και εξόδου. Επειδή, όμως, ορισμένες υπηρεσίες δεν είναι ασυνήθιστο να στέλνουν παρόμοια δεδομένα, όπως, για παράδειγμα, οι διακομιστές αρχείων (file servers), η προσέγγιση έχει περιορισμένη αποτελεσματικότητα εδώ.

Μία άλλη μορφή συμπεριφορικής υπογραφής ονομάζεται *fanout*. Η *fanout* αποτελεί ειδική κατηγορία της σχέσης απογόνου (που θα παρουσιαστεί στη συνέχεια), όπου λαμβάνονται υπόψη μόνο οι απόγονοι βάθους 1. Η fanout υπογραφή βάζει απλά ένα όριο στον αριθμό απογόνων που μπορεί να έχει ένας εξυπηρετητής (host) ανά πάσα στιγμή και, αν ο εξυπηρετητής ξεπεράσει αυτό το όριο, τότε τον μπλοκάρει.

---

[32] Στην Επιστήμη των Υπολογιστών, πελάτης ονομάζεται κάποιος υπολογιστής ή λογισμικό που αποκτά πρόσβαση σε μία υπηρεσία που γίνεται διαθέσιμη από ένα διακομιστή (server).

[33] Στο Client-Server μοντέλο, ο πελάτης (client) είναι το κομμάτι του μοντέλου που ζητάει δεδομένα ή υπηρεσίες και είναι αυτός που ξεκινάει την επικοινωνία μεταξύ τους, ενώ ο διακομιστής (server) είναι το κομμάτι που του τις παρέχει. Και οι δύο είναι προγράμματα που τρέχουν σε εξυπηρετητές (hosts).

[34] Ένα δίκτυο υπολογιστών peer-to-peer (P2P) είναι ένα δίκτυο που επιτρέπει σε δύο ή περισσότερους υπολογιστές να μοιράζονται τους πόρους τους ισοδύναμα. Το δίκτυο αυτό χρησιμοποιεί την επεξεργαστική ισχύ, τον αποθηκευτικό χώρο και το εύρος ζώνης (bandwidth) των κόμβων, οι οποίοι έχουν όλοι ίσα δικαιώματα. Πληροφορίες που βρίσκονται στον έναν κόμβο, ανάλογα με τα δικαιώματα που καθορίζονται, μπορούν να διαβαστούν από όλους τους άλλους και αντίστροφα.

Η σχέση απογόνου είναι παράδειγμα επαγωγικής υπογραφής (*inductive signature*) και έχει μεγαλύτερη ευαισθησία από την προηγούμενη. Οι επαγωγικές υπογραφές θεωρούν ότι υπάρχει κάποιο σύνολο από μολυσμένους εξυπηρετητές, οι οποίοι είναι υπεύθυνοι για τη μόλυνση άλλων εξυπηρετητών, που με τη σειρά τους και αυτοί μολύνουν ακόμα περισσότερους, προκαλώντας έτσι μία εκθετική αύξηση των μολύνσεων και, ταυτόχρονα, μία ένδειξη ότι τα σκουλήκια του υπολογιστή εξαπλώνονται γρήγορα. Για να μπορέσει να δημιουργηθεί μία υπογραφή για αυτού του τύπου τη συμπεριφορά, πρέπει να τεθούν όρια στα εξής σημεία:

- Στο βάθος του μακρινότερου απογόνου σε δεδομένο χρόνο,
- Στον αριθμό των απογόνων στο δένδρο,
- Στο μέσο συντελεστή διακλάδωσης[35] και
- Στο μέσο χρόνο που χρειάζεται για να φτάσει σε ένα συγκεκριμένο βάθος στο δένδρο.

Οι συγγραφείς ανέλυσαν τις υπογραφές διακομιστή-πελάτη, τις alpha-in/alpha-out και τις επαγωγικές, τόσο σε επίπεδο ακρίβειας όσο και σε απόδοσης. Η πρώτη βρέθηκε να είναι απόλυτα ακριβής στα ενεργά σκουλήκια[36] που αλλάζουν από διακομιστή σε πελάτη, με πολύ γρήγορες αποδόσεις, ακόμα και όταν δεν είναι γνωστοί εκ των προτέρων οι εξυπηρετητές και οι πελάτες, αλλά ανακαλύπτονται δυναμικά. Η δεύτερη εξαρτάται από την τιμή του ορίου που επιλέγεται για το alpha, ενώ η ταχύτητα εξαρτάται από το μηχανισμό που χρησιμοποιείται για να αναγνωριστεί το κοινό σήμα. Σχετικά με τις επαγωγικές υπογραφές, όσο περισσότερες μολύνσεις συμβαίνουν μεταξύ παρακολουθούμενων δρομολογητών (routers), τόσο ποιο ακριβής θα είναι η σχέση απογόνου. Επειδή όμως η απόδοση εδώ μετριέται σε εκθετικό χρόνο, μπορεί να καταστεί προβληματική η εφαρμογή της σε πραγματικό χρόνο.

Τέλος, οι συγγραφείς πρότειναν μία, κατά τη γνώμη τους, βέλτιστη αρχιτεκτονική δικτύου, η οποία αξιοποιεί τις δυνατότητες των παραπάνω συμπεριφορικών υπογραφών, με σκοπό την ακριβέστερη ανίχνευση των σκουληκιών.

[35] Βαθμός διακλάδωσης, ή βαθμός εξόδου, σε ένα δενδρικό διάγραμμα ονομάζεται ο αριθμός των τόξων ή ο αριθμός των απογόνων που ξεκινούν από μία κορυφή.

[36] Ενεργά (active worms) ονομάζονται τα σκουλήκια που ξεκινούν συνδέσεις προς εξυπηρετητές με σκοπό να εξαπλωθούν. Εν αντιθέσει, ένα παθητικό σκουλήκι περιμένει τον εξυπηρετητή να συνδεθεί με εκείνο (π.χ. τα μεταδοτικά σκουλήκια – contagion worms).

2.1.10. Ανίχνευση Εισβολών μέσω Στατικής Ανάλυσης

Οι David Wagner και Drew Dean στο έργο τους [10] προτείνουν τρία βασικά μοντέλα αντιμετώπισης επιθέσεων που προκαλούν την αντιφατική συμπεριφορά ενός προγράμματος σε σχέση με το σκοπό που αυτό είχε γραφτεί, δίνοντας, ταυτόχρονα, μεγάλη προσοχή στην εξάλειψη ψευδών συναγερμών. Η προσέγγισή τους βασίζεται τόσο σε στατική ανάλυση, όσο και σε δυναμική παρακολούθηση. Αυτό, αρχικά, προϋποθέτει ότι το πρόγραμμα δεν είχε γραφτεί για κακόβουλο σκοπό και ότι τα ίχνη των κλήσεων συστήματος του προγράμματος κατά την εκτέλεσή του συνάδουν με τον πηγαίο του κώδικα. Η βασική τους προσέγγιση είναι ο εντοπισμός των ιχνών των κλήσεων συστήματος που δεν μπορούν να έχουν παραχθεί από το υποβόσκων σύστημα μεταβάσεων, πάνω στο οποίο έχει μοντελοποιηθεί το πρόγραμμα. Οι συγγραφείς σημειώνουν ότι το σύστημα που προτείνουν δεν ανιχνεύει επιθέσεις που εκμεταλλεύονται λογικά λάθη του προγράμματος, όπως, για παράδειγμα, προγράμματα που δεν κάνουν ελέγχους επαλήθευσης στοιχείων πριν προχωρήσουν στη διαδικασία της εκτέλεσης, αλλά είναι πολύ αποτελεσματικό έναντι εκτελέσεων αλλοιωμένου κώδικα. Παρατηρήθηκε στην πράξη ότι, μόλις ένας επιτιθέμενος μολύνει μία εφαρμογή, στη συνέχεια τείνει να κατεβάσει κάποιο κώδικα εκμετάλλευσης αδυναμιών της επιλογής του και να το χρησιμοποιήσει κατά την εκτέλεση διάφορων λειτουργιών με τα δικαιώματα που έχει η εφαρμογή. Από τη στιγμή όμως που ο κώδικας αυτός δεν είναι μέρος του πηγαίου κώδικα της εφαρμογής, αν ποτέ εκτελεστεί, αναμένεται να υπάρξει συμπεριφορά που δεν ταιριάζει με τον πηγαίο κώδικα και, άρα, να ενεργοποιηθεί ο συναγερμός για πιθανή επίθεση.

Αρχικά, προτείνεται ένα βασικό μοντέλο, που δεν ανήκει στα τρία που αναφέρονται στην εισαγωγή, για να εξυπηρετήσει σκοπούς παρουσίασης της βασικής ιδέας, πάνω στην οποία βασίζονται τα υπόλοιπα. Το μοντέλο θεωρεί ένα σύνολο *S* από κλήσεις συστήματος που μπορεί να κάνει η εφαρμογή και *S** το σύνολο των αποδεκτών ιχνών των κλήσεων συστήματος, όπου S* η κανονική γλώσσα[37]. Αν, κατά τη διάρκεια της εκτέλεσης, παρατηρηθεί η εφαρμογή να κάνει κλήση συστήματος εκτός του συνόλου S, τότε εμποδίζεται η εκτέλεση της κλήσης αυτής, σταματάει η εφαρμογή και ενεργοποιείται ο συναγερμός. Η προσέγγιση αυτή είναι απλή και εύκολη στην υλοποίηση, αλλά όχι ιδιαίτερα αποτελεσματική στην πράξη, ειδικά αν το σύνολο S είναι αρκετά μεγάλο, κάτι που συμβαίνει στις μεγάλες εφαρμογές.

Στη συνέχεια, προτείνεται μία τεχνική που αναλύει τον πηγαίο κώδικα ενός προγράμματος και εξάγει ένα διάγραμμα ροής ελέγχου (control-flow graph), το οποίο αντιπροσωπεύει τα ίχνη των κλήσεων συστήματός του. Έτσι, ενεργοποιείται ένας συναγερμός αν διαπιστωθεί ότι, κατά τη διάρκεια της εκτέλεσης, πραγματοποιήθηκε κλήση συστήματος που δεν ήταν στο μοντέλο. Αυτός ο τύπος μοντέλου αναφέρεται ως μοντέλο γραφημάτων κλήσεων και η αναπαράστασή του γίνεται μέσω ενός μη-ντετερμινιστικού πεπερασμένου αυτομάτου[38]. Να σημειωθεί εδώ ότι το μοντέλο αυτό επιτρέπει την ύπαρξη ανέφικτων διαδρομών, όπως, για παράδειγμα, να γίνει μία κλήση και να επιστρέψει σε διεύθυνση διαφορετική από τη διεύθυνση επιστροφής της, καθώς το αυτόματο δεν μπορεί να εκφράσει τέτοιου είδους περιορισμούς, με συνέπεια επιθέσεις που ακολουθούν αυτές τις αδύνατες διαδρομές να παραμείνουν απαρατήρητες.

Για να καταπολεμηθεί το παραπάνω, προτάθηκε το αφηρημένο μοντέλο στοίβας, το οποίο είναι ουσιαστικά μία εικονική στοίβα κλήσεων[39], και το οποίο αναπαρίσταται μέσω της χρήσης μη-ντετερμινιστικού αυτόματου με στοίβα. Η κατασκευή του αυτόματου εξασφαλίζει ότι κάθε πιθανή ακολουθία λειτουργιών στη στοίβα κλήσεων του προγράμματος κατά τη διάρκεια της ομαλής εκτέλεσής του θα είναι εντός του συνόλου που έχει δημιουργηθεί κατά την προσομοίωση του

[37] Στην επιστήμη των υπολογιστών, κανονική ονομάζεται η γλώσσα εκείνη που αναγνωρίζεται από ένα πεπερασμένο αυτόματο.

[38] Μη ντετερμινιστικό ονομάζεται το αυτόματο που από μία κατάσταση, διαβάζοντας ένα σύμβολο εισόδου, μπορεί να μεταβεί σε μία ή και παραπάνω καταστάσεις, σε αντίθεση με το ντετερμινιστικό που μπορεί να μεταβεί μόνο σε μία. Πεπερασμένα ονομάζονται τα αυτόματα που ο αριθμός των καταστάσεών τους είναι πεπερασμένος.

[39] Στοίβα κλήσεων είναι μία δομή δεδομένων στοίβας που κρατά πληροφορίες σχετικά με τις ενεργές διαδικασίες ενός προγράμματος.

αυτομάτου. Η διατήρηση του εικονικού μοντέλου μπορεί να είναι ακριβής υπολογιστικά, ανάλογα με την υλοποίησή του.

Οι συγγραφείς πρότειναν, τέλος, το λεγόμενο διγραφικό μοντέλο, όπου μία ακολουθία από δύο διαδοχικές κλήσεις συστήματος, που ξεκινούν από ένα τυχαίο σημείο του προγράμματος κατά τη διάρκεια της εκτέλεσης, μπορεί να χρησιμοποιηθεί για να αποφασιστεί κατά πόσο το ίχνος μίας κλήσης συστήματος είναι κακόβουλο ή όχι. Σε γενικές γραμμές, η προσέγγιση αυτή είναι δύσκολη και, για αριθμό διαδοχικών κλήσεων συστήματος k μεγαλύτερο του 2, πολύ αργή. Αφού γίνουν οι απαιτούμενοι υπολογισμοί για να κατασκευαστεί η λίστα με τις επιτρεπόμενες k-ακολουθίες, διατηρείται ιστορικό με τις τελευταίες k-1 κλήσεις συστήματος και, όταν εμφανιστεί μία νέα κλήση, ελέγχεται κατά πόσο η επακόλουθη k-ακολουθία επιτρέπεται.

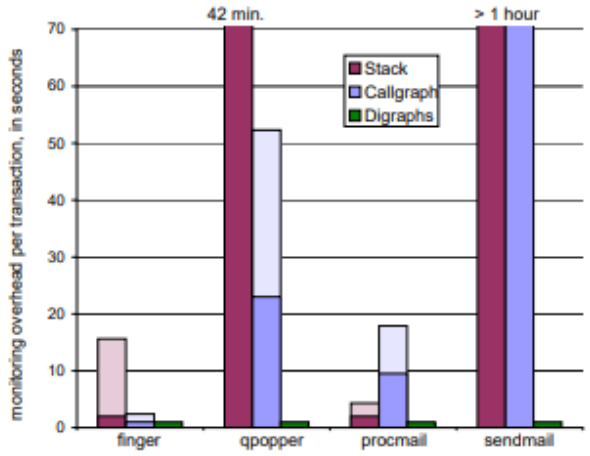
Για την αξιολόγηση των μοντέλων, χρησιμοποιήθηκαν τέσσερις αντιπροσωπευτικές εφαρμογές με γνωστές αδυναμίες ασφαλείας:

1. *finger*,
2. *qpopper*,
3. *procmail* και
4. *sendmail*.

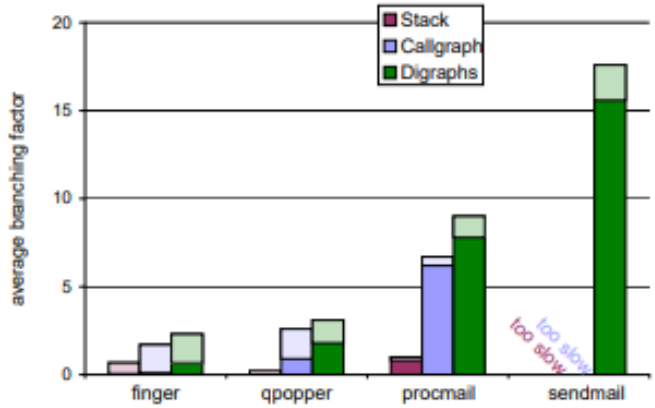
Για σκοπούς μέτρησης της απόδοσης, η υπολογιστική επιβάρυνση του κάθε μοντέλου μετρήθηκε σε δευτερόλεπτα για κάθε επιπλέον υπολογισμό ανά διαδραστικό γεγονός, όπως, για παράδειγμα, η παράδοση ενός email. Προκειμένου να αξιολογηθεί η ακρίβεια των μοντέλων, οι συγγραφείς χρησιμοποίησαν το μέσο συντελεστή διακλάδωσης. Εάν η εκτέλεση της εφαρμογής επρόκειτο να παγώσει κάποια στιγμή κατά τη διάρκεια της εκτέλεσης, τότε ο συντελεστής διακλάδωσης θα ήταν το σύνολο των κλήσεων συστήματος που θα μπορούσαν να εκτελεστούν στη συνέχεια, χωρίς να ενεργοποιηθούν καθόλου συναγερμοί. Επιθυμητός θεωρείται ένας μικρός παράγοντας διακλάδωσης καθώς αυτό υποδηλώνει ότι εισβολείς που προσπαθούν να παρακάμψουν το σύστημα ανίχνευσης απειλών, μιμούμενοι την κανονική συμπεριφορά του προγράμματος υπό εξέταση, θα έχουν λιγότερους τρόπους να επιτεθούν με επιτυχία σε ένα σύστημα χωρίς να εντοπιστούν.

Οι συγγραφείς βρήκαν ότι ο έλεγχος των παραμέτρων στις κλήσεις συστήματος βοήθησε σε θέματα απόδοσης και ακρίβειας, στην οποία περίπτωση το αφηρημένο μοντέλο στοίβας βρέθηκε να είναι το πιο ακριβές από τα τρία, ακολουθούμενο από το μοντέλο γραφημάτων κλήσεων, ακολουθούμενο, τέλος, από το διγραφικό μοντέλο. Σε αυτήν την περίπτωση, η βελτίωση στην απόδοση είναι αποτέλεσμα της μειωμένης ασάφειας του μοντέλου και, άρα, της μείωσης του αριθμού των πιθανών διαδρομών του μοντέλου που πρέπει να εξεταστούν κατά τη διάρκεια εκτέλεσης. Η μείωση αυτή είναι που συμβάλει και στη βελτίωση στην ακρίβεια, μαζί με το γεγονός ότι πολλές κλήσεις συστήματος είναι ακίνδυνες όταν οι παράμετροί τους είναι από πριν σταθερές.

Συγκεντρωτικά, οι Εικόνες 7 και 8 παρουσιάζουν τα αποτελέσματα αξιολόγησης των τριών μοντέλων σε θέματα απόδοσης και ακρίβειας αντίστοιχα, για τις τέσσερις προαναφερθείσες εφαρμογές.



Εικόνα 7 – Υπολογιστική επιβάρυνση των μοντέλων



Εικόνα 8 – Μέτρηση ακρίβειας των μοντέλων

Να σημειωθεί για τις δύο εικόνες εδώ ότι το αφηρημένο μοντέλο στοίβας αντιπροσωπεύεται από την αριστερή κάθετη μπάρα, το μοντέλο κλήσεων από τη μεσαία και το διγραφικό μοντέλο από τη δεξιά. Κάθε μπάρα χρησιμοποιεί σκίαση για να παρουσιάσει δύο μετρήσεις: το μικρότερο τμήμα, με τα πιο έντονα χρώματα αντιπροσωπεύει τις περιπτώσεις όπου είναι ενεργός ο έλεγχος των παραμέτρων των κλήσεων συστήματος, ενώ το συνολικό ύψος της κάθε μπάρας, συμπεριλαμβανομένου τόσο του έντονου χρωματικά τμήματος όσο και του πιο ανοιχτού, αντιπροσωπεύει τις περιπτώσεις όπου οι παράμετροι αυτοί αγνοούνται. Τέλος, σχετικά με τις εφαρμογές που επιλέχθηκαν, η finger είναι η μικρότερη, της τάξης των χιλίων (1K) γραμμών κώδικα, και η sendmail η μεγαλύτερη με περίπου 32 χιλιάδες (32K) γραμμές κώδικα. Οι υπόλοιπες δύο εφαρμογές είναι στη μέση.

2.1.11. Ανίχνευση Ανωμαλιών βάσει Προδιαγραφών[40]: Μία Καινούρια Προσέγγιση για τον Εντοπισμό Επιθέσεων Δικτύου

Οι R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang και S. Zhou στο έργο τους [11] προτείνουν τη μοντελοποίηση της συμπεριφοράς ενός δικτύου μέσω ενός εκτεταμένου πεπερασμένου αυτόματου (Extended Finite State Automaton – EFSA) και, πιο συγκεκριμένα, το σύστημα διεπαφής της πύλης δικτύου του δικτύου-στόχος. Ένα τέτοιο αυτόματο είναι παρόμοιο με ένα πεπερασμένο αυτόματο, με τη διαφορά ότι το συγκεκριμένο μπορεί, επιπλέον, αφενός να κάνει μεταβάσεις σε γεγονότα με παραμέτρους και, αφετέρου, να χρησιμοποιήσει ένα πεπερασμένο σύνολο μεταβλητών όπου οι τιμές τους μπορούν να αποθηκευτούν. Σε γενικές γραμμές, τα αυτόματα πρωτοκόλλων είναι μη-ντετερμινιστικά. Η προσομοίωση του μη-ντετερμινισμού επιτυγχάνεται μέσω της δημιουργίας *k* αντιγράφων του αυτομάτου κάθε φορά που μπορεί να κάνει κάποια από *k* διαφορετικές μεταβάσεις. Η διαδικασία αυτή δημιουργεί αντίγραφα όχι μόνο των καταστάσεων του αυτομάτου, αλλά και των μεταβλητών του. Οι περιπτώσεις που φτάνουν κάθε φορά στην τελική κατάσταση[41] αυτόματα διαγράφονται.

Η τεχνική που προτείνεται εδώ αποτελεί συνδυασμό ανίχνευσης βάσει ανωμαλιών και ανίχνευσης βάσει προδιαγραφών, μεγεθύνοντας τα δυνατά σημεία των δύο τεχνικών και ελαχιστοποιώντας τα αδύναμα. Η ανίχνευση ανωμαλιών επικεντρώνεται στις συμπεριφορές του συστήματος που το αυτόματο θεωρεί κανονικές και χαρακτηρίζεται από δύο φάσεις: τη φάση εκπαίδευσης και τη φάση ανίχνευσης. Στη φάση εκπαίδευσης (training phase), παρατηρείται η συμπεριφορά του συστήματος σε περιβάλλον άνευ επιθέσεων και χρησιμοποιούνται τεχνικές μηχανικής μάθησης με σκοπό τη δημιουργία ενός προφίλ κανονικής συμπεριφοράς. Τα χαρακτηριστικά που κρίνονται ως πιο χρήσιμα για τη φάση της εκπαίδευσης, επιλέγονται βάσει των προδιαγραφών του αυτομάτου. Στη φάση ανίχνευσης (detection phase), συγκρίνεται το προφίλ που δημιουργήθηκε στην προηγούμενη φάση με την τρέχουσα συμπεριφορά του συστήματος και οι όποιες παρεκκλίσεις μαρκάρονται ως πιθανές επιθέσεις. Αυτό, όμως, ειδικά στην περίπτωση κανονικής αλλά έως τώρα πρωτοφανούς συμπεριφοράς, μπορεί να οδηγήσει σε αύξηση του αριθμού ψευδών συναγερμών. Επιπλέον, να σημειωθεί ότι η αποτελεσματικότητα αυτού του είδους της τεχνικής επηρεάζεται άμεσα από τον τύπο των χαρακτηριστικών της συμπεριφοράς του συστήματος που μαθαίνονται κατά την πρώτη φάση.

Οι τεχνικές βάσει προδιαγραφών είναι παρόμοιες με τις παραπάνω στο ότι και αυτές κρίνουν τις επιθέσεις ως παρεκκλίσεις από κάποια νόρμα. Η διαφορά, όμως, είναι ότι αυτές οι τεχνικές δε βασίζονται σε τεχνικές μηχανικής μάθησης αλλά σε χειροκίνητα ανεπτυγμένες προδιαγραφές που επικεντρώνονται περισσότερο στις έγκυρες παρά σε ήδη γνωστές συμπεριφορές. Αυτό οδηγεί σε μικρότερο αριθμό ψευδών συναγερμών από προηγουμένως άγνωστες αλλά νόμιμες συμπεριφορές. Το μειονέκτημα εδώ, όμως, είναι το μεγάλο κόστος που χρειάζεται η ανάπτυξη από άποψη χρόνου και ο προσδιορισμός λεπτομερών προδιαγραφών, η έλλειψη των οποίων μπορεί να οδηγήσει σε αυξημένο αριθμό ψευδών αρνητικών – την πιθανότητα, δηλαδή, να μην ανιχνευτούν κάποιες από τις επιθέσεις.

Οι συγγραφείς αξιοποίησαν τις στατιστικές ιδιότητες που παρατηρούνται κατά την κίνηση του δικτύου, για να αποφανθούν για την τυχόν κακόβουλη συμπεριφορά των γεγονότων που λαμβάνουν μέρος στο δίκτυο-στόχος. Στις στατιστικές ιδιότητες περιλαμβάνονται:

- Η συχνότητα εμφάνισης συγκεκριμένων μεταβάσεων στο αυτόματο.
- Η πιο συχνά εμφανιζόμενη τιμή μιας μεταβλητής σε συγκεκριμένη κατάσταση του αυτομάτου.

[40] Η ανίχνευση βάσει Προδιαγραφών (Specification-based detection) είναι ειδική κατηγορία της ανίχνευσης ανωμαλιών. Οι τεχνικές αυτού του τύπου αξιοποιούν κάποια προδιαγραφή ή σύνολο κανόνων για το τι θεωρείται έγκυρη συμπεριφορά και κατόπιν αποφασίζουν αναλόγως για το αν είναι κακόβουλη ή όχι η συμπεριφορά του εκάστοτε προγράμματος υπό εξέταση.

[41] Η τελική κατάσταση στην προκειμένη περίπτωση δεν αναφέρεται σε κατάσταση αποδοχής όπως σε ένα πεπερασμένο αυτόματο, αλλά σε κατάσταση όπου δεν μπορεί να υπάρξει άλλη μετάβαση από κει και πέρα.

- Η κατανομή των τιμών των μεταβλητών που παρατηρείται κατά τη διάρκεια κάποιας χρονικής περιόδου.

Στην πρώτη περίπτωση, ανήκει και ο αριθμός των μεταβάσεων που κάνουν timeout (σε προκαθορισμένο χρονικό διάστημα - στο άρθρο δίνεται ο χρόνος των 60 δευτερολέπτων σαν όριο) από υποσύνολα ιχνών[42] του αυτομάτου, κάτι που μπορεί να βοηθήσει στην αναγνώριση χρήσιμων ιδιοτήτων για τη ροή των δεδομένων. Η επιλογή υποσυνόλου έναντι ολόκληρου του συνόλου των ιχνών μπορεί να προσδιοριστεί είτε βάσει των τιμών των μεταβλητών, όπως, για παράδειγμα, στην περίπτωση παρακολούθησης δεδομένων προς μία συγκεκριμένη τοπική μηχανή, είτε βάσει χρόνου, όπως, για παράδειγμα, στην περίπτωση παρακολούθησης ιχνών τα τελευταία *t* δευτερόλεπτα, με σκοπό την πιο πρόσφατη δυνατή συμπεριφορά.

Κατά τη διάρκεια της φάσης ανίχνευσης γίνεται σύγκριση μεταξύ των στατιστικών ιδιοτήτων που παρατηρούνται και των τιμών που συγκεντρώθηκαν κατά τη φάση εκπαίδευσης. Αν τα στατιστικά στοιχεία έχουν σημαντική απόκλιση από τις τιμές εκείνες, τότε επισημαίνεται ανωμαλία, ενώ γίνονται περαιτέρω έλεγχοι για τον ακριβή προσδιορισμό του όρου «σημαντική απόκλιση».

Οι συγγραφείς χρησιμοποίησαν για τα πειράματά τους τα δεδομένα αξιολόγησης Lincoln Labs 1999 και επικεντρώθηκαν σε επιθέσεις σε χαμηλού επιπέδου πρωτόκολλα, όπως τα IP και TCP, καθώς μέχρι εκείνη τη στιγμή είχαν αναπτυχθεί αυτόματα μόνο για αυτά τα δύο πρωτόκολλα. Σε αυτού του είδους τις επιθέσεις ανήκει και η κατηγορία των Denial-of-Service επιθέσεων (DoS). Καθώς η τεχνική τους βασίζεται στην επανάληψη για τον εντοπισμό ανωμαλιών, η επίθεση πρέπει να περιλαμβάνει τουλάχιστον δύο πακέτα[43], οπότε αυτό βγάζει εκτός του πεδίου εφαρμογής της μεθόδου πολλά είδη επιθέσεων που χρησιμοποιούν μόνο ένα. Η προσέγγισή τους κατάφερε να εντοπίσει όλες τις επιθέσεις από τα δεδομένα που ήταν εντός του πεδίου της εφαρμογής τους. Η μέθοδος αυτή παρήγαγε 5,5 ψευδείς συναγερμούς την ημέρα, που σα νούμερο είναι χαμηλό σε σύγκριση με το ποσοστό ψευδών συναγερμών που αναφέρεται στα δεδομένα αξιολόγησης του Lincoln Labs, ενώ η επεξεργασία των δεδομένων της μίας μέρας (περίπου 0,70GB σε μέγεθος) διήρκησε λιγότερο από δέκα λεπτά σε επεξεργαστή Pentium III των 700MHz με 1GB μνήμη.

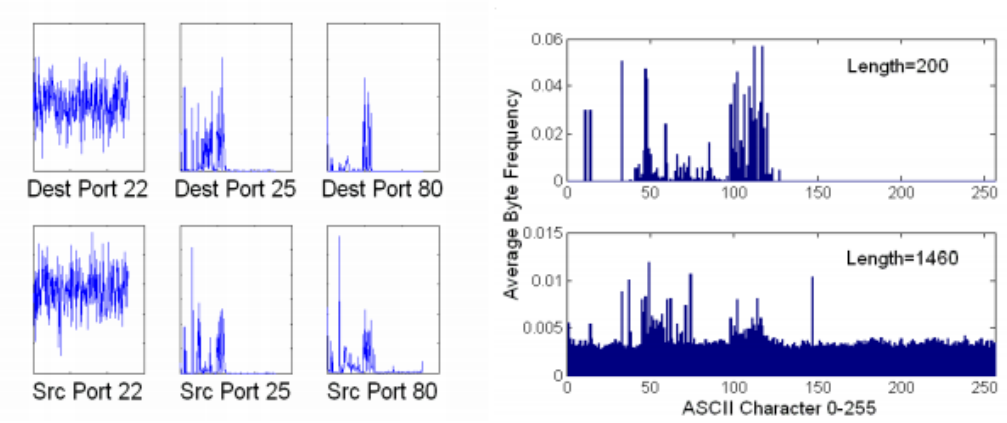
[42] Ίχνος ονομάζεται μία ακολουθία καταστάσεων. Με τον όρο κατάσταση εννοούμε τόσο τις καταστάσεις του αυτομάτου (όπως η αρχική και η τελική) όσο και τις τιμές των μεταβλητών.

[43] Στην επιστήμη των δικτύων υπολογιστών, το πακέτο αποτελεί τη βασική μονάδα πληροφορίας που μεταφέρεται μέσα στα δίκτυα.

2.1.12. Ανίχνευση Ανωμαλιών Ωφέλιμων Φορτίων (PAYL)

Οι Ke Wang και Salvatore J. Stolfo στο έργο τους [12] παρουσιάζουν το PAYL, ένα εργαλείο που υπολογίζει το αναμενόμενο ωφέλιμο φορτίο (payload)[44] μίας εφαρμογής ή υπηρεσίας δικτύου. Η μέθοδός τους στοχεύει, πρωτίστως, στα πρώτα γεγονότα εμφάνισης σκουληκιών του υπολογιστή (computer worms) είτε σε μία πύλη δικτύου, είτε σε ένα εσωτερικό δίκτυο, καθώς και στην αποτροπή της εξάπλωσής τους. Ενώ η κύρια χρήση της είναι έναντι των σκουληκιών, εντούτοις είναι χρήσιμη και σε μια ευρύτερη κλίμακα επιθέσεων εκμετάλλευσης αδυναμιών διαφόρων υπηρεσιών.

Το εργαλείο αρχικά μαθαίνει το προφίλ του αναμενόμενου ωφέλιμου φορτίου της υπηρεσίας, κατά τη διάρκεια της ομαλής λειτουργίας του συστήματος. Κατά τη φάση εκπαίδευσης, δημιουργείται μία συχνότητα κατανομής των bytes, η οποία επιτρέπει την ανάπτυξη ενός κεντροειδούς μοντέλου (*centroid model*) για το τι θεωρείται κανονικό ωφέλιμο φορτίο για κάθε μία από τις υπηρεσίες του εξυπηρετητή (host). Καθώς τα φορτία δεν είναι όλα ίδια μεταξύ τους, υπολογίζεται ένα διαφορετικό μοντέλο για κάθε εύρος μήκους, για κάθε πύλη και υπηρεσία και για κάθε κατεύθυνση ροής (inbound / outbound). Η Εικόνα 10 που ακολουθεί, το αριστερό κομμάτι, δείχνει διάφορα παραδείγματα κατανομής των bytes διαφορετικών μεταξύ τους φορτίων. Ο X-άξονας αντιπροσωπεύει τους ASCII χαρακτήρες 0-255 και ο Y-άξονας τη μέση συχνότητα των bytes, ενώ το δεξί παράδειγμα διαφορετικών μηκών αναφέρεται σε φορτία για την πύλη 80 του ίδιου κεντρικού διακομιστή.



Εικόνα 9 - Παραδείγματα κατανομής των bytes για διαφορετικές πύλες, μήκη και κατευθύνσεις ροής

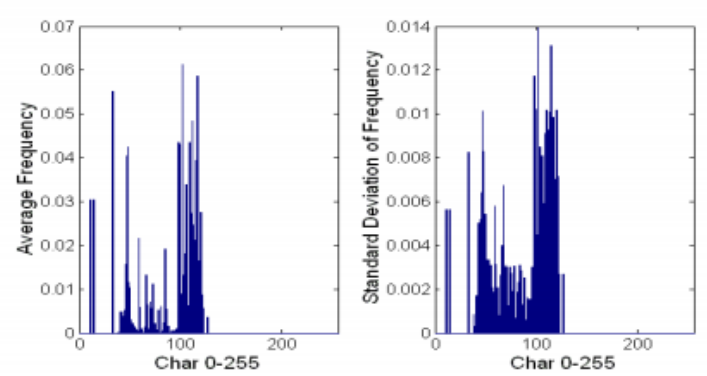
Η μοντελοποίηση γίνεται μέσω της n-gram ανάλυσης[45], για n = 1, όπου υπολογίζεται ένα σύνολο μοντέλων M_{ij}, όπου i το παρατηρούμενο μήκος και j η πύλη στην οποία αποστέλλεται το φορτίο. Μοντέλα των οποίων η ομοιότητα των μέσων συχνοτήτων κατανομής των bytes τους είναι αρκετά κοντά συγχωνεύονται μεταξύ τους. Η διαδικασία επαναλαμβάνεται μέχρι να μην μπορέσει να υπάρξει άλλη συγχώνευση. Το χαρακτηριστικό διάνυσμα[46] του εκάστοτε φορτίου είναι η σχετική μέτρηση συχνότητας του κάθε n-gram, η οποία υπολογίζεται διαιρώντας τον αριθμό των εμφανίσεων του εκάστοτε n-gram με το συνολικό αριθμό των n-grams. Οπότε, στην προκειμένη περίπτωση της

[44] Ωφέλιμο φορτίο, ή payload, είναι ουσιαστικά μία ροή από bytes, χωρίς κάποια συγκεκριμένη μορφή ή μήκος ή συγκεκριμένο εύρος τιμών. Στο χώρο των τηλεπικοινωνιών, αναφέρεται στο κύριο μέρος των μεταδιδόμενων δεδομένων, χωρίς να περιλαμβάνονται επικεφαλίδες (headers) και μετά-δεδομένα (metadata). Στα πλαίσια των κακόβουλων λογισμικών, αποτελεί το κομμάτι του κώδικα που είναι υπεύθυνο για την κακόβουλη συμπεριφορά.

[45] Ως n-gram χαρακτηρίζεται μία ακολουθία από n διαδοχικά bytes σε μία μονάδα ωφέλιμου φορτίου. Ένα συρόμενο παράθυρο πλάτους n περνάει από όλο το φορτίο και μετρείται η εμφάνιση του κάθε n-gram.

[46] Στη μηχανική μάθηση, χαρακτηριστικά διανύσματα (feature vectors) ή απλά χαρακτηριστικά (features), ονομάζονται οι μεμονωμένες, μετρήσιμες ιδιότητες ενός φαινομένου προς παρακολούθηση. Αντιπροσωπευτικό παράδειγμα αυτού είναι ο τρόπος που περιγράφεται ένα χρώμα από το πόσο κόκκινο (Red), μπλε (Blue) και πράσινο (Green) έχει, όπου το χαρακτηριστικό του διάνυσμα θα είναι της μορφής = [R, G, B].

1-gram, υπολογίζεται η μέση συχνότητα του κάθε ASCII χαρακτήρα 0-255. Το κάθε μοντέλο από αυτά αποθηκεύει τη μέση συχνότητα των bytes και την τυπική απόκλιση της συχνότητας του κάθε byte. Η Εικόνα 11 που ακολουθεί δείχνει το υπολογιζόμενο μοντέλο για φορτίο μήκους 185 για την πύλη 80.



Εικόνα 10 - Μέση συχνότητα και τυπική απόκλιση της συχνότητας του κάθε byte για φορτίο μήκους 185 για την πύλη 80.

Κατά τη φάση ανίχνευσης, το εργαλείο σκανάρει τα εισερχόμενα φορτία και υπολογίζει την κατανομή των bytes τους. Στη συνέχεια, οι καινούριες αυτές κατανομές συγκρίνονται με του μοντέλου M_{ij}, μετρώντας μία απλουστευμένη μορφή της Mahalanobis[47] απόστασης μεταξύ αυτών. Αν το εισερχόμενο ωφέλιμο φορτίο είναι αρκετά μακριά από το κεντροειδές μοντέλο, αν δηλαδή η τιμή στη Mahalanobis απόσταση ξεπεράσει ένα συγκεκριμένο όριο, τότε το φορτίο θεωρείται κακόβουλο και ενεργοποιείται ο συναγερμός. Ανάλογα με τις πολιτικές ασφαλείας του συστήματος υπό εξέταση, υπάρχουν σαν πιθανές επιλογές το φιλτράρισμα, η ανακατεύθυνση και η παγίδευση της σύνδεσης, προκειμένου η μόλυνση να μην εξαπλωθεί περαιτέρω στο σύστημα.

Οι συγγραφείς αξιολόγησαν την τεχνική τους στα δεδομένα Lincoln Labs 1999 του MIT και, πιο συγκεκριμένα, στο σύνολο δεδομένων 1999 DARPA IDS. Εδώ περιλαμβάνονται δεδομένα εκπαίδευσης 3 εβδομάδων (training data) και δεδομένα ελέγχου 2 εβδομάδων (testing data). Στα δεδομένα εκπαίδευσης, οι 2 εβδομάδες περιλαμβάνουν δεδομένα άνευ επιθέσεων, ενώ η τρίτη βδομάδα περιλαμβάνει και επισημασμένες επιθέσεις. Από τις 201 επιθέσεις που περιλαμβάνονται στα δεδομένα αυτά, η τεχνική θα έπρεπε να είναι σε θέση να ανιχνεύσει τις 97, αλλά τελικά κατάφερε να εντοπίσει τις 57. Συνολικά, δηλαδή, το ποσοστό ανίχνευσης της τεχνικής τους ήταν περίπου στο 60 τοις εκατό (≈60%), με το ποσοστό των ψευδώς θετικών αποτελεσμάτων να είναι περίπου στο 1 τοις εκατό (≤ 1%). Η Εικόνα 12 που ακολουθεί δίνει αναλυτικά τα συνολικά ποσοστά εντοπισμού για πέντε διαφορετικά μοντέλα:

- Per Packet Model, το οποίο χρησιμοποιεί ολόκληρο το ωφέλιμο φορτίο του εκάστοτε πακέτου
- First 100 Packet Model, το οποίο χρησιμοποιεί τα πρώτα 100 bytes του εκάστοτε πακέτου
- Tail 100 Packet Model, το οποίο χρησιμοποιεί τα τελευταία 100 bytes του εκάστοτε πακέτου
- Per Conn Model, το οποίο χρησιμοποιεί ολόκληρο το ωφέλιμο φορτίο της εκάστοτε σύνδεσης

[47] Η Mahalanobis απόσταση είναι μέθοδος μέτρησης της απόστασης μεταξύ δύο στατιστικών κατανομών και δίνεται από τον εξής τύπο: $d^2(x,\bar{y}) = (x-\bar{y})^T C^{-1}(x-\bar{y})$, όπου x και $\bar{y}$ είναι δύο χαρακτηριστικά διανύσματα, με κάθε στοιχείο τους να είναι μια μεταβλητή. Το x είναι το χαρακτηριστικό διάνυσμα της καινούριας παρατήρησης και το $\bar{y}$ το μέσο χαρακτηριστικό διάνυσμα που έχει υπολογιστεί από τα παραδείγματα κατά τη φάση της εκπαίδευσης. Ο C^{-1} είναι ο πίνακας αντίστροφης συνδιακυμανσης, ή αλλιώς πίνακας ακρίβειας, με $C^{-1} = Cov(y_i, y_j)$, όπου y_i είναι το i-οστό και y_j το j-οστό στοιχείο του διανύσματος της περιόδου εκπαίδευσης.

- Truncated Conn Model, το οποίο χρησιμοποιεί τα πρώτα 1000 bytes της εκάστοτε σύνδεσης

Per Packet Model	57/97 (58.8%)
First 100 Packet Model	55/97 (56.7%)
Tail 100 Packet Model	46/97 (47.4%)
Per Conn Model	55/97 (56.7%)
Truncated Conn Model	51/97 (52.6%)

Εικόνα 11 - Συνολικό ποσοστό ανίχνευσης του κάθε μοντέλου με ποσοστό ψευδώς θετικών μικρότερο του 1%

Οι συγγραφείς χρησιμοποίησαν επίσης το σύνολο δεδομένων του Columbia University Computer Science (CUCS) για να αξιολογήσουν την τεχνική τους, καθώς τα δεδομένα αυτά είναι πραγματικά δεδομένα, σε αντίθεση με του Lincoln Labs που, αν και πολύ λεπτομερή, είναι προσομοιωμένα. Τα δεδομένα αυτά είναι δύο ίχνη εισερχόμενης κίνησης πλήρους φορτίου στον web server του CS τμήματος (www.cs.columbia.edu). Τα δύο αυτά ίχνη συγκεντρώθηκαν ξεχωριστά, το πρώτο (A) τον Αύγουστο του 2003, μεγέθους περίπου 2GB, σε διάστημα 45 ωρών και το δεύτερο (B) το Σεπτέμβρη του ίδιου έτους, μεγέθους περίπου 1GB, σε διάστημα 24 ωρών. Δεν ήταν εξαρχής γνωστό κατά πόσο στο σύνολο αυτό περιλαμβάνονταν και επιθέσεις ή όχι, οπότε τα δεδομένα αντιμετωπίστηκαν σαν μη-χαρακτηρισμένα (δεδομένα, δηλαδή, που δεν μπορούν να κατηγοριοποιηθούν). Το εργαλείο μπόρεσε να εντοπίσει buffer overflow επιθέσεις (επιθέσεις υπερχείλισης της μνήμης) καθώς και το Code Red II worm.

Train	Test	Anomaly #	CR-II	Buffer
A	A	28(0.0084%)	----	----
A	B	2601(1.3%)	Yes	Yes
B	A	686(0.21%)	----	----
B	B	184(0.092%)	Yes	Yes
AB	AB	211(0.039%)	Yes	Yes

Εικόνα 12 - Αποτελέσματα μη-επιβλεπόμενης μάθησης[48] στο σύνολο δεδομένων CUCS

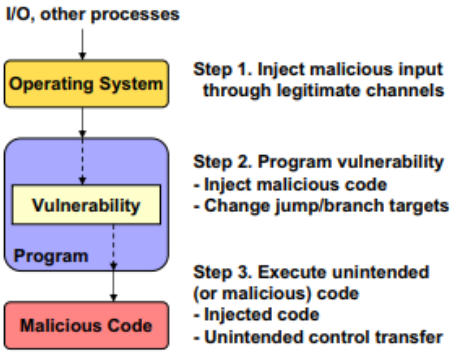
Η χρήση πραγματικών δεδομένων δίνει κάποια σιγουριά στους συγγραφείς ότι η τεχνική τους αυτή μπορεί να χρησιμοποιηθεί αρκετά αποτελεσματικά και σε πραγματικό περιβάλλον δικτύου. Λόγω των πολιτικών απορρήτου του Πανεπιστημίου της Columbia, τα σύνολο δεδομένων καταστράφηκε, με συνέπεια να μην μπορούν άλλοι ερευνητές να το χρησιμοποιήσουν ως άλλη βάση σύγκρισης για ανιχνευτές εισβολών.

[48] Η μη-επιβλεπόμενη μάθηση αποτελεί κατηγορία της μηχανικής μάθησης, στόχος της οποίας είναι η ανακάλυψη πιθανής δομής που μπορεί να υπάρχει σε μη-χαρακτηρισμένα δεδομένα.

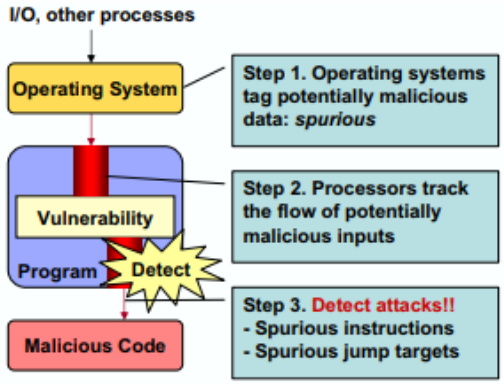
2.1.13. Ασφαλής Εκτέλεση Προγράμματος μέσω Δυναμικής Παρακολούθησης της Ροής Πληροφοριών

Οι G. Edward Suh, Jaewook Lee και Srinivas Devadas στο έργο τους [13] προτείνουν μία μέθοδο εντοπισμού κακόβουλου λογισμικού, η οποία απαιτεί τροποποίηση στο λειτουργικό σύστημα και στον επεξεργαστή. Η μέθοδός τους βασίζεται στην ιδέα ότι τα δεδομένα μπορούν να χαρακτηριστούν είτε ως ασφαλή είτε ως μη ασφαλή – αυθεντικά[49] ή ψευδή, αντίστοιχα.

Η μέθοδός τους επικεντρώνεται σε επιθέσεις οι οποίες προσπαθούν να πάρουν τον έλεγχο κάποιου ευάλωτου προγράμματος που έχει δικαιώματα και, έτσι, να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε ολόκληρο το σύστημα, μέσω της εισαγωγής κακόβουλου κώδικα ή μέσω της αλλαγής της ροής ελέγχου του προγράμματος. Η αλλαγή στη ροή ελέγχου ενός προγράμματος οδηγεί στην εκτέλεση κομματιών κώδικα που, υπό άλλες συνθήκες, δεν θα έπρεπε να εκτελεστεί, μέσω της τροποποίησης κάποιων από τους δείκτες[50] του προγράμματος στη μνήμη.



Οι συγγραφείς σημειώνουν ότι τα πιθανώς κακόβουλα κανάλια εισόδου, τα κανάλια, δηλαδή, από τα οποία μπορούν να έρθουν οι κακόβουλες επιθέσεις, τα διαχειρίζεται το λειτουργικό σύστημα. Το λειτουργικό σύστημα καθορίζει μία πολιτική ασφαλείας, της οποίας η ισχύ επιβάλλεται από τον επεξεργαστή και η οποία αποτελείται από τρία μέρη: ψευδή κανάλια εισόδου, εξαρτήσεις που πρέπει να μπουν υπό παρακολούθηση και περιορισμοί. Αυτή η πολιτική ασφαλείας θα εντοπίσει τα ψευδή δεδομένα, τα οποία είναι συνήθως τα δεδομένα εισόδου / εξόδου των μηχανισμών δικτύωσης, και θα τους αναθέσει ένα bit (δυαδικό ψηφίο), ώστε να μην μπορούν να χρησιμοποιηθούν για να αλλάξουν τη ροή ελέγχου του προγράμματος. Από την άλλη, άλλα δεδομένα, συμπεριλαμβανομένου του αρχικού προγράμματος όταν φορτώνεται την πρώτη φορά, χαρακτηρίζονται ως αυθεντικά. Στην προκειμένη περίπτωση, το 0 συμβολίζει τα αυθεντικά δεδομένα, ενώ το 1 τα ψευδή. Ο επεξεργαστής εξασφαλίζει ότι ο έλεγχος του προγράμματος δεν εξαρτάται από τα ψευδή δεδομένα. Αν όμως βρεθεί, μέσω των bits που τους έχουν επισυναφθεί, ότι ο έλεγχος βασίζεται σε δεδομένα που είναι ψευδή, ο επεξεργαστής θα πετάξει εξαίρεση που θα την πιάσει το λειτουργικό σύστημα, στο οποίο σημείο το λειτουργικό σύστημα θα σκοτώσει τη διαδικασία που την προκάλεσε.



Εικόνα 14 - Προτεινόμενο σχέδιο

Τα δεδομένα παρακολουθούνται μέσω εξαρτήσεων. Οι εξαρτήσεις αυτές χωρίζονται σε τέσσερις κατηγορίες[51]:

[49] Για τους σκοπούς του έργου αυτού ο όρος αυθεντικότητα χρησιμοποιείται για να δείξει κατά πόσο το δεδομένο είναι υπό τον έλεγχο του προγράμματος ή όχι.
[50] Δείκτης, ή pointer στα αγγλικά, είναι τύπος δεδομένων του οποίου η τιμή είναι η διεύθυνση μνήμης κάποιου άλλου δεδομένου.
[51] Διατηρήθηκαν οι αυθεντικές ονομασίες των εξαρτήσεων στα αγγλικά, ενώ οι περιγραφές και επεξηγήσεις δόθηκαν στα ελληνικά.

- ***Copy dependency***: Αν μία ψευδής τιμή αντιγραφεί σε διαφορετική τοποθεσία, τότε η τιμή της νέας τοποθεσίας θεωρείται και εκείνη ψευδής.
- ***Computation (Comp) dependency***: Μία ψευδής τιμή μπορεί να χρησιμοποιηθεί ως τελεστής εισόδου σε κάποιον υπολογισμό, το αποτέλεσμα του οποίου εξαρτάται άμεσα από τον τελεστή εισόδου, στην οποία περίπτωση το αποτέλεσμα θεωρείται και αυτό ψευδές. Για παράδειγμα, αν υπάρχει ένας τελεστής της λειτουργίας πρόσθεσης που έχει επισημανθεί ως ψευδής, τότε το αποτέλεσμα θα επισημανθεί ως ψευδές, ακόμα και αν ο άλλος τελεστής θεωρείται αυθεντικός.
- ***Load-address (LDA) dependency***: Όταν μία ψευδής τιμή χρησιμοποιείται για να καθορίσει τη διεύθυνση πρόσβασης, η φορτωμένη τιμή θεωρείται ψευδής. Εκτός και αν τα όρια της ψευδούς τιμής ελέγχονται ρητά από το πρόγραμμα, το αποτέλεσμα θα μπορούσε να είναι μία οποιαδήποτε τιμή, αφού προέρχεται από μία απρόβλεπτη διεύθυνση.
- ***Store-address (STA) dependency***: Η αποθηκευμένη τιμή γίνεται ψευδής αν η διεύθυνση αποθήκευσης καθορίζεται από ψευδή τιμή. Αν ένα πρόγραμμα δε γνωρίζει πού αποθηκεύει μία τιμή, δε θα περιμένει να αλλάξει η τιμή στην τοποθεσία, όταν φορτώσει στο μέλλον από τη διεύθυνση αυτή.

Οι συγγραφείς, βασιζόμενοι στις παραπάνω εξαρτήσεις, καθόρισαν δύο ειδών πολιτικές ασφαλείας τις οποίες και χρησιμοποίησαν μετά στις προσομοιώσεις τους. Η 1η πολιτική παρακολουθεί μόνο τις τρείς από τις τέσσερις παραπάνω εξαρτήσεις και, πιο συγκεκριμένα, αφήνοντας απ' έξω την Computation dependency. Η πολιτική αυτή αποτελεί μία ελαφριά έκδοση που προσφέρει ασφάλεια έναντι επιθέσεων με μικρή υπολογιστική επιβάρυνση (overhead). Η 2η πολιτική παρακολουθεί και τις τέσσερις παραπάνω εξαρτήσεις, προσφέροντας έτσι προστασία εναντίον ενός ευρύτερου αριθμού επιθέσεων.

Οι συγγραφείς αξιολόγησαν την τεχνική τους και με τις δύο πολιτικές ασφαλείας έναντι επιθέσεων υπερχείλισης της στοίβας μνήμης (stack buffer overflows), επιθέσεων υπερχείλισης του σωρού μνήμης (heap buffer overflows), επιθέσεων vudo (που είναι ειδικός τύπος heap buffer overflow επίθεσης) και format string επιθέσεων[52]. Η τεχνική τους κατάφερε να σταματήσει όλες τις επιθέσεις, χωρίς κανένα ψευδή συναγερμό, σε καμία από τις δύο πολιτικές. Τέλος, οι δύο πολιτικές αξιολογήθηκαν και σε θέματα υπολογιστικής επιβάρυνσης και απόδοσης, με την υλοποίηση της πρώτης να απαιτεί, κατά μέσο όρο, επιβάρυνση στη μνήμη της τάξης του 0,26% και της δεύτερης της τάξης του 4,5%. Αντίστοιχα, σε θέματα απόδοσης, η υλοποίηση της πρώτης πολιτικής επιδείνωσε την απόδοση κατά 0,02%, κατά μέσο όρο, ενώ της δεύτερης κατά 0,8%.

[52] Επιθέσεις τύπων buffer overflows και format string επιτρέπουν στον επιτιθέμενο να αντικαταστήσει τις θέσεις μνήμης, στο ευάλωτο χώρο μνήμης του προγράμματος, με κακόβουλο κομμάτι κώδικα. Εκμεταλλευόμενο την αδυναμία αυτή, το κακόβουλο λογισμικό μπορεί να πάρει τον έλεγχο του προγράμματος και να εκτελέσει οποιαδήποτε λειτουργία για την οποία έχει δικαιώματα το μολυσμένο πρόγραμμα.

2.1.14. Εντοπισμός Επιθέσεων στις Εφαρμογές Χρησιμοποιώντας Δυναμική Ανάλυση Ροής Πληροφοριών

Οι Wes Masri και Andy Podgurski στο έργο τους [14] περιγράφουν ένα εργαλείο που ονομάζεται Δυναμική Ανάλυση Ροής Πληροφοριών[53] (Dynamic Information Flow Analysis – DIFA). Το εργαλείο αυτό σχεδιάστηκε ειδικά για Java εφαρμογές. Έχει τη δυνατότητα παρακολούθησης των μεθόδων του κώδικα, κατά τη διάρκεια της εκτέλεσης του προγράμματος, μέσω των bytecodes[54] κλάσεων των εφαρμογών. Ως εκ τούτου, η ροή των πληροφοριών μπορεί να ληφθεί κάθε φορά που καλείται μία μέθοδος από την εφαρμογή και μπορεί να συγκριθεί με γνωστές πολιτικές ροών πληροφοριών. Ορισμένες τέτοιες ροές ενδέχεται να υποδηλώνουν είτε διαρροή ευαίσθητων πληροφοριών είτε παραποίηση αυτών.

Στη συνέχεια, παρατίθενται ορισμένοι από τους λόγους που η τεχνική αυτή είναι χρήσιμη για τον εντοπισμό επιθέσεων έναντι εφαρμογών:

- Ο σκοπός ορισμένων εξεζητημένων επιθέσεων είναι ακριβώς η εισαγωγή μη ασφαλών ροών πληροφοριών από ή προς συγκεκριμένα αντικείμενα και οι ροές αυτές είναι η μόνη ένδειξη αυτού του τύπου των επιθέσεων.
- Το DIFA επιτρέπει τον αυτόματο εντοπισμό επιθέσεων και ανεκμετάλλευτων αδυναμιών που παραβιάζουν κάποια πολιτική ροής πληροφοριών.
- Τα μοτίβα ροών πληροφοριών που εμφανίζονται κατά την εκτέλεση ενός προγράμματος χαρακτηρίζουν σε μεγάλο βαθμό τον υπολογισμό του. Ως εκ τούτου, συγκεκριμένες επιθέσεις προκαλούν την εμφάνιση κάποιων χαρακτηριστικών μοτίβων ροών, τα οποία με τη σειρά τους μπορούν να αναγνωριστούν από ένα σύστημα ανίχνευσης εισβολών βασισμένο στις υπογραφές (signature-based). (Να σημειωθεί επίσης ότι αυτές οι επιθέσεις δε χρειάζεται να περιλαμβάνουν ούτε διαρροή ούτε παραποίηση της πληροφορίας.)
- Ορισμένες ροές πληροφοριών είναι απαραίτητες σε ορισμένες επιθέσεις και διατηρούνται ακόμα και όταν η ακολουθία των επιθέσεων καλυφθεί με κάποιους τρόπους. Οπότε, ένα σύστημα ανίχνευσης που κοιτάει για υπογραφές επιθέσεων που είναι καθορισμένες από τις ροές πληροφοριών, θα είναι άτρωτο απέναντι σε τέτοιου είδους απόπειρες.
- Ορισμένες άγνωστες έως τώρα επιθέσεις προκαλούν μη ομαλά μοτίβα ροών πληροφοριών και αυτό με τη σειρά του μπορεί να αποτελέσει μία καλή βάση για την ανίχνευση αυτών.
- Η Δυναμική Ανάλυση Ροών Πληροφοριών μπορεί να πετύχει μεγαλύτερο ποσοστό ακρίβειας από την αντίστοιχη στατική ανάλυση, ακριβώς επειδή η πρώτη δε χρειάζεται τη συντηρητική υπόθεση ότι όλα τα μονοπάτια ροών ελέγχου του προγράμματος πρέπει να είναι εκτελέσιμα. Κατά συνέπεια, η DIFA θα πρέπει να έχει μικρότερο ποσοστό ψευδών συναγερμών.

Το εργαλείο είναι σχεδιασμένο να υποστηρίζει ρυθμιζόμενες πολιτικές ροών πληροφοριών, κάτι που λείπει από την στατική ανάλυση. Δεδομένης της δυναμικής φύσεώς του, το εργαλείο μπορεί εύκολα να χειριστεί λειτουργίες γλώσσας, όπως οι πίνακες και οι δείκτες. Χειρίζεται εξαρτήσεις τόσο εσωτερικές σε μια διαδικασία όσο και μεταξύ διαφόρων διαδικασιών (όχι όμως και εξαρτήσεις που προέρχονται από εξαιρέσεις), καθώς και ροές ελέγχου μεταξύ νημάτων σε πολύ-νηματικά προγράμματα.

Οι επιθυμητές πολιτικές ροών πληροφοριών μπορούν να καθοριστούν από το χρήστη μέσω του προγραμματιστικού προσδιορισμού των ακόλουθων τριών οντοτήτων:

---

[53] Αυτές οι ροές πληροφοριών περιλαμβάνουν ροές δεδομένων μεταξύ των μεταβλητών του προγράμματος και / ή τις τιμές αυτών ύστερα από την εκτέλεση συνθηκών στο πρόγραμμα.

[54] Ως bytecodes αναφέρεται η γλώσσα μηχανής της JVM (Java Virtual Machine). Όταν η JVM φορτώνει το .class αρχείο, παίρνει μία ροή από bytecodes για κάθε μέθοδο στην κλάση. Οι ροές αυτές αποθηκεύονται στο χώρο μεθόδων και αποτελούν ουσιαστικά ακολουθίες οδηγιών προς τη JVM. Εκτελούνται όταν καλεστεί η μέθοδός τους κατά τη διάρκεια εκτέλεσης του προγράμματος.
Πηγή: https://www.javaworld.com/article/2077233/core-java/bytecode-basics.html

1. Τα αντικείμενα προέλευσης προς παρακολούθηση, αναφερόμενα ως Sources.
2. Τα ευαίσθητα αντικείμενα μεταξύ των Sources που μπορεί να εμπλέκονται σε παράνομες ροές, αναφερόμενα ως SensitiveSources.
3. Τις μεθόδους “sink”[55], στις οποίες ελέγχονται οι παράνομες ροές, αναφερόμενες ως Sinks.

Τα Sources και SensitiveSources είναι ουσιαστικά το ίδιο πράγμα, ενώ μία μέθοδος “sink” θα ήταν οποιαδήποτε συνάρτηση εξόδου καλούμενη από μη εξουσιοδοτημένους χρήστες ή μη εξουσιοδοτημένες εφαρμογές, όπως η write() και η send().

Στο εργαλείο περιλαμβάνονται δύο βασικά στοιχεία: ο *Instrumenter*[56] και ο *Profiler*. Ο πρώτος εισάγει κάποιον αριθμό από μεθόδους στο δεύτερο σε συγκεκριμένα σημεία ενδιαφέροντος. Κατά τη διάρκεια της εκτέλεσης, η εφαρμογή καλεί τον Profiler, δίνοντάς του τις πληροφορίες που χρειάζονται για την παρακολούθηση των ροών πληροφοριών.

Οι συγγραφείς παρουσιάζουν περιπτώσεις μελέτης (case studies) σχετικές με παραβιάσεις των πολιτικών των ροών πληροφοριών (*MeetingScheduler* και *JConsole*), υπογραφές ροών πληροφοριών από επιθέσεις (παράδειγμα λογαριασμού χρήστη με το αντίστοιχο υπόλοιπο), καθώς και ανίχνευση ανώμαλων ροών πληροφοριών (*JSP Source Disclosure*). Η τελευταία περίπτωση αφορούσε μία αδυναμία του Apache Tomcat[57] σε θέματα ασφαλείας. Μετά το στήσιμο του Apache Tomcat, οι συγγραφείς έστειλαν κανονικά αιτήματα (requests), καθώς και κακόβουλα, στον διακομιστή (server). Με βάση τις συστάδες (clusters) που δημιουργήθηκαν από τα 150 αιτήματα που στάλθηκαν στον Apache Tomcat server, έγινε εμφανές ότι τα κακόβουλα αιτήματα, περίπου 20 στο σύνολο, μπορούσαν να ανιχνευθούν από τις γραφικές αναπαραστάσεις των ροών πληροφοριών του προφίλ του κάθε αιτήματος, καθώς τα κανονικά βρίσκονταν σε διαφορετικά τμήματα στο γράφημα σε σχέση με τα κακόβουλα.

[55] Στον προγραμματισμό, sink ονομάζεται μία κλάση ή συνάρτηση που έχει σχεδιαστεί να λαμβάνει εισερχόμενα γεγονότα από κάποια άλλη συνάρτηση ή αντικείμενο.
[56] Για την υλοποίησή του χρησιμοποιήθηκε η BCEL – Byte Code Engineering Library.
[57] Ο Apache Tomcat είναι ο application server της Java. Έχει υλοποιημένες πολλές από τις προδιαγραφές της Java και παρέχει ένα καθαρό περιβάλλον διακομιστή δικτύου (web server environment), όπου μπορεί να τρέξει κώδικας Java.

2.1.15. Στατικός Αναλυτής Εκτελέσιμων Αρχείων (Static Analyzer for Executables - SAFE)

Οι Mihai Christodorescu και Somesh Jha στο έργο τους [15] προτείνουν το SAFE, έναν στατικό αναλυτή εκτελέσιμων αρχείων. Αν και η μέθοδος που προτείνεται εδώ είναι γενικής χρήσεως, εντούτοις επικεντρώνεται πρωτίστως στην ανίχνευση ιών[58], ειδικά μετά τη χρήση τεχνικών απόκρυψης του κώδικα (obfuscation techniques).

Η τεχνική θεωρείται ότι είναι ανθεκτική σε τεχνικές απόκρυψης του κακόβουλου κώδικα (obfuscation techniques), όπως ο *πολυμορφισμός*[59] και ο *μεταμορφισμός*[60]. Τέσσερις από τους ιούς που ανέλυσε το SAFE είναι οι *Chernobyl (CIH)*, *zombie-6.b, f0sf0r0* και *Hare*. Οι συγγραφείς ανέπτυξαν, επίσης, ένα εργαλείο, το δυαδικό obfuscator, για να τεστάρουν την αποτελεσματικότητα εμπορικών antivirus λογισμικών έναντι τεχνικών απόκρυψης. Το εργαλείο αυτό υποστηρίζει τέσσερις κοινούς μετασχηματισμούς απόκρυψης: εισαγωγή κώδικα που δε χρησιμοποιείται (dead-code insertion), αντιμετάθεση κώδικα (με αποτέλεσμα η δυαδική εικόνα να είναι διαφορετική από τη σειρά εκτέλεσης), εκ νέου ανάθεση των καταχωρητών (όπου γίνεται ανταλλαγή στα ονόματα των καταχωρητών χωρίς να επηρεάζεται η συμπεριφορά του προγράμματος) και αντικατάσταση οδηγιών με άλλες ισοδύναμες οδηγίες. Τα τρία antivirus λογισμικά που εκλέχθηκαν ήταν: *Norton® Antivirus 7.0*, *McAfee® VirusScan 6.01* και *Command® Antivirus 4.61.2* και ενώ όλα εντόπισαν τις αυθεντικές εκδόσεις των προαναφερθέντων ιών, εντούτοις κανένα δεν κατάφερε να ανιχνεύσει εκδόσεις που είχαν έστω και ελαφρές τεχνικές απόκρυψης του κώδικα αυτών.

Το SAFE έχει τη δυνατότητα αφηρημένης αναπαράστασης του κακόβουλου κώδικα, κατασκευάζοντας το αντίστοιχο αυτόματο που τον αντιπροσωπεύει. Το αυτόματο, το οποίο αποτελεί μία γενίκευση πεπερασμένου αυτομάτου του οποίου η αλφάβητος είναι ένα πεπερασμένο σύνολο από μοτίβα, περιλαμβάνει γενικευμένα σύμβολα, τα οποία παρέχουν ένα γενικό τρόπο αντιπροσώπευσης των εξαρτήσεων μεταξύ των μεταβλητών. Πέρα από τη δυνατότητα δημιουργίας του παραπάνω αυτομάτου, το SAFE περιλαμβάνει και τα εξής δομικά στοιχεία:

- ***Pattern definition loader***, το οποίο χρησιμοποιεί αφηρημένα/γενικευμένα μοτίβα και δημιουργεί μία εσωτερική αναπαράσταση. Τα γενικευμένα μοτίβα είναι τα ίδια που χρησιμοποιούνται ως σύμβολα αλφαβήτου από το αυτόματο.
- ***The executable loader***, το οποίο χρησιμοποιεί τα *IDA Pro* (version 4.1.7.600) και *CodeSurfer* (version 1.5) εργαλεία για να μεταμορφώσει το εκτελέσιμο σε μία εσωτερική αναπαράσταση, στην προκειμένη περίπτωση μία συλλογή από γραφήματα ροών ελέγχου (CFG), ένα για την κάθε διαδικασία του προγράμματος. Κάθε κόμβος του γραφήματος είναι η μέγιστη ακολουθία οδηγιών που περιλαμβάνει στο τέλος της το πολύ μία οδηγία ροής

[58] Ένας ιός υπολογιστή μπορεί να πολλαπλασιαστεί εισάγοντας ένα αντίγραφο του κώδικά του στο πρόγραμμα-ξενιστή. Όταν ο χρήστης εκτελέσει το μολυσμένο πρόγραμμα, εκτελείται το αντίγραφο του ιού, μολύνοντας ακόμα περισσότερα προγράμματα, ενώ το αρχικό πρόγραμμα συνεχίζει κανονικά να εκτελείται. Μέχρι ο ιός να ενεργοποιήσει το κακόβουλο φορτίο του, δεν υπάρχει αντιληπτή διαφορά στο χρήστη μεταξύ κανονικού και μολυσμένου προγράμματος.

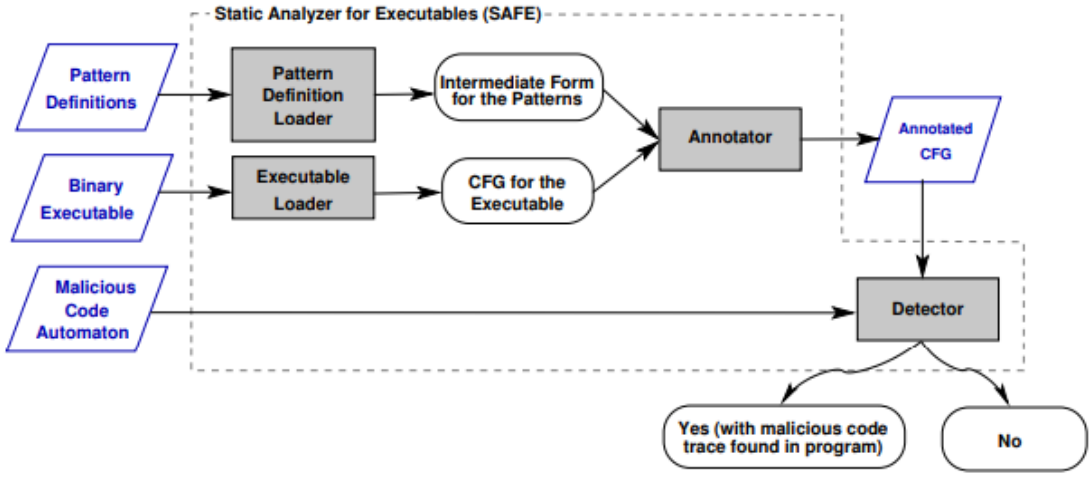
[59] Ένας *πολυμορφικός* ιός έχει, αρχικά, κρυπτογραφημένο κώδικα και μόνο ένα μικρό κομμάτι εντολών είναι σχεδιασμένο να αποκρυπτογραφήσει τον κώδικα πριν την εκτέλεσή του. Όταν ο πολυμορφικός ιός δημιουργεί αντίγραφο του εαυτού του μολύνοντας κάποιο άλλο πρόγραμμα, κρυπτογραφεί το κυρίως σώμα του με ένα πρόσφατα δημιουργημένο κλειδί, ενώ αλλάζει και τη ρουτίνα αποκρυπτογράφησης παράγοντας νέο κώδικα για αυτήν. Διάφορες μετατροπές που ακολουθούνται προς απόκρυψη του κώδικά της είναι: η εισαγωγή εντολών noop (εντολές που δεν κάνουν τίποτα), η αντιμετάθεση του κώδικα (αντιμετάθεση της σειράς οδηγιών, διατηρώντας την αρχική σημασιολογία) και η εκ νέου ανάθεση των καταχωρητών.

[60] Ένας *μεταμορφικός* ιός χρησιμοποιεί ακόμα πιο πολύπλοκες τεχνικές απόκρυψης του κώδικα. Όταν πολλαπλασιάζονται, μερικοί από τους τρόπους που αλλάζουν τον κώδικά τους οι ιοί αυτοί είναι: αντιμετάθεση του κώδικα, εκ νέου μετάθεση των καταχωρητών και αντικατάσταση ισοδύναμων ακολουθιών από οδηγίες. Μπορούν, επιπλέον, να ενσωματώσουν τον κώδικά τους στον κώδικα του προγράμματος, κάνοντας σχεδόν αδύνατη την ανίχνευσή τους.

ελέγχου (control-flow instruction). Οι ακμές μεταξύ των κόμβων χαρακτηρίζονται ως αληθείς ή ψευδείς, ανάλογα με τη συνθήκη που προηγήθηκε της μετάβασης.

- ***The annotator***, το οποίο χρησιμοποιεί το γράφημα ροής ελέγχου του εκτελέσιμου, μαζί με το σύνολο των γενικευμένων μοτίβων που αναφέρθηκαν προηγουμένως, για να παραγάγει μία αφηρημένη αναπαράσταση της κάθε διαδικασίας του προγράμματος, μέσω της μορφής γραφήματος με σχολιασμούς. Κάθε κόμβος του νέου γραφήματος έχει χαρακτηρισθεί με το σύνολο των μοτίβων που αντιστοιχίζονται σε αυτόν. Το παραγόμενο γράφημα περιλαμβάνει πληροφορίες που υποδεικνύουν τη θέση καθενός από τα γενικευμένα μοτίβα στο εκτελέσιμο.
- ***The detector***, το οποίο υπολογίζει κατά πόσο ο κακόβουλος κώδικας (που αντιπροσωπεύεται από το αυτόματο) εμφανίζεται στην αφηρημένη αναπαράσταση του εκτελέσιμου (που δημιουργείται από τον annotator). Αν οποιοσδήποτε από τους τύπους κακόβουλου κώδικα που υπάρχουν στο αυτόματο βρεθεί στο γράφημα ροής ελέγχου με τους σχολιασμούς, τότε το εκτελέσιμο θεωρείται κακόβουλο. Για το σκοπό αυτό γίνεται χρήση unification αλγορίθμου, που είναι η διαδικασία επίλυσης εξισώσεων μεταξύ συμβολικών εκφράσεων, για τον υπολογισμό της τομής μεταξύ της γλώσσας του γραφήματος με σχολιασμούς και της γλώσσας των γενικευμένων μοτίβων. Αν η τομή αυτή είναι μη-κενή, τότε αυτό σημαίνει ότι βρέθηκε κάποιο μοτίβο που είναι κοινό μεταξύ του αυτομάτου και του γραφήματος του εκτελέσιμου, στην οποία περίπτωση επιστρέφεται η ακολουθία οδηγιών του εκτελέσιμου που αντιστοιχεί στο κακόβουλο μοτίβο.

Στην παρακάτω εικόνα (Εικόνα 15) δίνεται μία γενική αναπαράσταση της αρχιτεκτονικής αυτής.



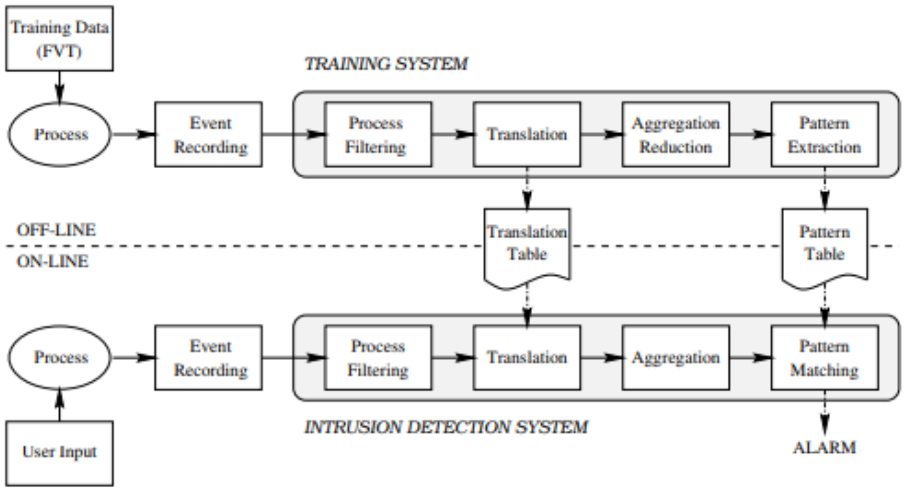
Εικόνα 15 - Αρχιτεκτονική του εργαλείου SAFE

Τα πειράματα, που διενεργήθηκαν από τους συγγραφείς σε λειτουργικό σύστημα Microsoft Windows 2000 με επεξεργαστή AMD Athlon 1GHz και 1GB μνήμη RAM, έδειξαν ότι το SAFE είχε ποσοστό μηδέν για τα ψευδώς θετικά και για τα ψευδώς αρνητικά, ενώ επιδέχεται βελτιώσεις σε θέματα απόδοσης, καθώς σε συγκεκριμένες περιπτώσεις είχε μη αποδεκτά μεγάλους χρόνους εκτέλεσης.

2.1.16. Ανίχνευση Εισβολών Χρησιμοποιώντας Μοτίβα Παρακολούθησης Ίχνους Μεταβλητού Μήκους

Οι Andreas Wespi, Marc Dacier και Hervé Debar στο έργο τους [16] προτείνουν τη χρήση μεταβλητού μήκους μοτίβων παρακολούθησης ίχνους για την ανίχνευση εισβολών. Η χρήση μεταβλητού μήκους μοτίβων, αν και πιο δύσκολα στην παραγωγή από τα σταθερού μήκους μοτίβα, βοηθάει στην αντιμετώπιση σημαντικών προβλημάτων, καθώς, αφενός η επιλογή μικρότερου μήκους οδηγεί σε μικρότερης ακρίβειας μοτίβα και, αφετέρου, τα μεγαλύτερου μήκους συνεπάγονται αύξηση των υπολογισμών που απαιτούνται για τη διαδικασία ανίχνευσης.

Το προτεινόμενο σύστημα αποτελείται από δύο μέρη: τη φάση εκπαίδευσης (offline μέρος) και τη φάση ανίχνευσης (online μέρος). Κατά τη φάση εκπαίδευσης (training phase), γίνεται πολλές φορές επανεκτέλεση του προγράμματος σε κατάλληλα απομονωμένο περιβάλλον, σε όσες περισσότερες καταστάσεις λειτουργίας γίνεται, με σκοπό να καταγραφεί η συμπεριφορά του, η οποία χαρακτηρίζεται ως ομαλή. Κατά τη διάρκεια εκτέλεσης, αποφασίζεται κατά πόσο η παρακολουθούμενη συμπεριφορά σχετίζεται με την ομαλή συμπεριφορά της προηγούμενης φάσης και, στην περίπτωση σημαντικών αποκλίσεων, θεωρείται ότι υπάρχει μεγάλη πιθανότητα εισβολής.



Εικόνα 16 - Αρχιτεκτονική του συστήματος ανίχνευσης εισβολών

Πιο αναλυτικά, η παρακολούθηση της συμπεριφοράς του προγράμματος, κατά τη φάση εκπαίδευσης, γίνεται μέσω της καταγραφής των γεγονότων ελέγχου[61] κατά τις διαφορετικές εκτελέσεις της διαδικασίας. Τα γεγονότα αυτά ταξινομούνται με βάση το process id τους, διατηρώντας τη χρονολογική τους σειρά (process filtering), και μεταφράζονται σε ακολουθίες χαρακτήρων (translation), παράγοντας τον αντίστοιχο πίνακα κανόνων (translation table). Στη συνέχεια, συγκεντρώνονται μαζί οι διαδοχικές εμφανίσεις του ίδιου χαρακτήρα, του ίδιου γεγονότος δηλαδή, και αφαιρούνται τα διπλότυπα, αφού δεν περιλαμβάνουν καινούρια μοτίβα (aggregation and reduction). Στο τέλος, οι ακολουθίες αυτές περνάνε στο κομμάτι εξαγωγής μοτίβων (pattern extraction), όπου και δημιουργείται ο αντίστοιχος πίνακας (pattern table). Σκοπός είναι η συγκέντρωση του μικρότερου δυνατού συνόλου μοτίβων, το οποίο καλύπτει όλες τις περιπτώσεις ακολουθιών που εμφανίστηκαν σε αυτό το βήμα. Το σύνολο αυτό θα είναι κατά προτίμηση μικρό και θα αποτελείται από μοτίβα σχετικά μεγάλου μήκους.

[61] Τα γεγονότα αυτά δείχνουν όλη την ακολουθία από πιθανές δραστηριότητες που επηρέασαν μία διαδικασία ή γεγονός, σε οποιαδήποτε χρονική στιγμή.

Σχετικά με τα μοτίβα, ένα μοτίβο μεταβλητού μήκους ορίζεται ως μία υπό-ακολουθία που θα έχει τουλάχιστον δύο γεγονότα (δηλαδή το μήκος της θα είναι τουλάχιστον ίσο με δύο) και θα συμβεί τουλάχιστον δύο φορές, είτε στην ίδια είτε σε διαφορετικές ακολουθίες. Επιπλέον, ως μέγιστο μοτίβο[62], που είναι κ ο τύπος μοτίβου στον οποίο επικεντρώνεται το σύστημα, ορίζεται εκείνο που θα συμβεί τις περισσότερες φορές από όλα τα υπόλοιπα και δεν θα είναι υπό-ακολουθία κάποιου άλλου που εμφανίζεται ίσο αριθμό φορών με αυτό.

Κατά τη διαδικασία της σύγκρισης των μοτίβων, ο αλγόριθμος επεξεργάζεται ακολουθίες από την αρχή ως το τέλος τους. Για κάθε ακολουθία, ο αλγόριθμος επιστρέφει έναν αριθμό συνόλων από διαδοχικά γεγονότα που δεν έχουν καλυφθεί από κάποια ακολουθία, μαζί με το μήκος του κάθε συνόλου. Αν υπάρχουν πολλά τέτοια σύνολα γεγονότων, τότε υπάρχει μεγάλη πιθανότητα ότι η υπό έλεγχο διαδικασία αποτελεί πιθανή εισβολή. Η πιθανότητα αυτή αυξάνεται ανάλογα με το μήκος του κάθε συνόλου, όσο μεγαλύτερο είναι το μήκος αυτό δηλαδή, τόσο μεγαλύτερη είναι και η πιθανότητα εισβολής.

Η φάση ανίχνευσης έχει παρόμοια δόμηση με τη φάση εκπαίδευσης. Παράγονται γεγονότα από τη διαδικασία και επεξεργάζονται σε πραγματικό χρόνο, ενώ η διαδικασία φιλτραρίσματος είναι ακριβώς η ίδια. Η μετάφραση εδώ βασίζεται στις εγγραφές που υπάρχουν στον παραχθέντα από την προηγούμενη φάση πίνακα, με τα γεγονότα για τα οποία δεν υπάρχει αντίστοιχη εγγραφή να θεωρούνται ασυνήθη και να τους αποδίδεται ένας εικονικός χαρακτήρας, χωρίς να σημαίνεται όμως συναγερμός. Το στοιχείο της αφαίρεσης δεν είναι αναγκαίο εδώ, καθώς η αντιστοίχιση μοτίβων, μεταξύ των εισερχόμενων ακολουθιών και των καταχωρήσεων στον αντίστοιχο πίνακα, η οποία γίνεται σε πραγματικό χρόνο, δεν περιμένει να τελειώσει η κάθε διαδικασία για να ξεκινήσει. Με βάση το πόσο καλή αντιστοίχιση μπορεί να γίνει σε αυτό το στάδιο, κάτι που είναι ανεξάρτητο από το μήκος που ορίζουμε ότι πρέπει να ‘χει η εκάστοτε ακολουθία, αποφασίζεται κατά πόσο συναντάται κακόβουλη συμπεριφορά στο πρόγραμμα, στην οποία περίπτωση ενεργοποιείται ο συναγερμός.

Οι συγγραφείς χρησιμοποίησαν την *ftpd*[63] διαδικασία για να αξιολογήσουν τη μέθοδό τους[64] , λόγω της ευρείας χρήσης της και του γεγονότος ότι περιλαμβάνει πολλές αδυναμίες[65]. Συνέκριναν τη μέθοδό τους με την αντίστοιχη της Forrest και της ομάδας της[66], η οποία χρησιμοποιεί μοτίβα σταθερού μήκους (αντί για μεταβλητού). Από την ftpd διαδικασία προέκυψαν 65 μοναδικές, ομαλές ακολουθίες, οι οποίες περιλάμβαναν 26.025 γεγονότα ελέγχου και οι οποίες είχαν αντιστοίχιση με το 17% (11 στις 65) των μοτίβων σταθερού μήκους και το 72% (οι 47 στις 65) με τα μοτίβα μεταβλητού μήκους. Βάσει των πειραμάτων, φάνηκε ότι η προσέγγιση μεταβλητού μήκους είναι πιο ακριβής από την σταθερού, με λιγότερα μοτίβα να είναι απαραίτητα για την περιγραφή της ομαλής συμπεριφοράς, ενώ είναι πολύ μικρότερος και ο κίνδυνος ψευδούς συναγερμού.

[62] Για την εύρεση αυτών των μοτίβων χρησιμοποιήθηκε ο αλγόριθμος Teiresias, ο οποίος είχε αρχικά αναπτυχθεί με σκοπό την ανακάλυψη άκαμπτων μοτίβων σε μη στοιχισμένες βιολογικές ακολουθίες, σε συνδυασμό με έναν αλγόριθμο αφαίρεσης μοτίβων (pattern-reduction algorithm).

[63] Ως FTPD (File Transfer Protocols Daemon) ονομάζεται ο DARPA Internet File Transfer Protocol διακομιστής (server). Το File Transfer Protocol (FTP) είναι ένα σταθερό πρωτόκολλο δικτύου που χρησιμοποιείται για τη μεταφορά αρχείων μεταξύ ενός πελάτη και ενός διακομιστή (client – server) σε ένα δίκτυο υπολογιστών.

[64] Οι συγγραφείς, επίσης, αναφέρουν ότι εφήρμοσαν τη μέθοδό τους και στις υπηρεσίας δικτύου finger και sendmail, αλλά λόγω περιορισμένου χώρου, δεν παρουσίασαν τα αποτελέσματα τους σε αυτή τη δημοσίευση.

[65] Οι αδυναμίες αυτές είναι γνωστές και οφείλονται σε ελαττωματικό λογισμικό και ρυθμιστικά σφάλματα.

[66] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for UNIX processes. In Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, pages 120–128. IEEE Computer Society, IEEE Computer Society Press, May 1996. 110, 111, 112, 114

2.1.17.Δικτυακή Ανάλυση των Μη-Ομαλών Γεγονότων (NATE)

Τα περισσότερα συστήματα ανίχνευσης εισβολών δεν είναι σχεδιασμένα για υψηλή επισκεψιμότητα δικτύου (network traffic). Οι Carol Taylor και Jim Alves-Foss στο έργο τους [17] προτείνουν μία χαμηλού υπολογιστικού κόστους προσέγγιση ανίχνευσης της ανώμαλης κυκλοφορίας στο δίκτυο, η οποία ονομάζεται Network Analysis Traffic Events (NATE), και η οποία επιδιώκει να λύσει το πρόβλημα αυτό. Οι κύριες διαφορές του NATE σε σχέση με άλλα συστήματα ανίχνευσης εισβολών δικτύου[67] είναι ότι χρειάζεται ελάχιστη μέτρηση κυκλοφορίας[68], ενώ έχει απλουστευμένο σύστημα διαχείρισης, με μικρή συμμετοχή από το χρήστη, αλλά και περιορισμένο πεδίο επίθεσης. Πιο συγκεκριμένα, επικεντρώνεται σε επιθέσεις που εκμεταλλεύονται αδυναμίες του πρωτοκόλλου δικτύου και βασίζονται στην υπόθεση ότι κακόβουλης φύσεως πακέτα τείνουν να έχουν μεγάλο αριθμό από syn, fin και reset πακέτα[69] και ταυτόχρονα χαμηλό αριθμό από P και ack πακέτα. Στις ανώμαλες ροές κυκλοφορίας, παρατηρείται, επίσης, έλλειψη της ομαλής ροής των δεδομένων που περιλαμβάνονται στα πακέτα, το οποίο φαίνεται μέσω της καταγραφής του αριθμού των bytes που μεταφέρονται για το κάθε πακέτο.

Μια περίοδος λειτουργίας (session) ορίζεται ως η ροή πληροφοριών από τον πηγαίο συνδυασμό IP διεύθυνσης και αριθμού θύρας στον αντίστοιχο συνδυασμό προορισμού. Ομαδοποιώντας τις περιόδους λειτουργίας στα TCP/IP πρωτόκολλα μέσω συσταδοποίησης πολλαπλών μεταβλητών[70] και της τεχνικής Principal Components Analysis (PCA) για μείωση των δεδομένων, καθώς δεδομένα πολλαπλών μεταβλητών είναι δύσκολο να αναπαρασταθούν γραφικά, φάνηκε η διαφορετική κατανομή που υπάρχει μεταξύ ομαλών και μη-ομαλών sessions. Αυτό οφείλεται στο γεγονός ότι ένα από τα χαρακτηριστικά της ανώμαλης κυκλοφορίας είναι ο χαμηλός αριθμός πακέτων που μεταφέρονται από μια οποιαδήποτε πηγή σε κάποιο συνδυασμό IP διεύθυνσης και αριθμού θύρας (σε οποιοδήποτε session δηλαδή). Επίσης, η μέτρηση μόνο των επικεφαλίδων των πακέτων, σε σχέση με ολόκληρο το περιεχόμενό τους, το βοηθάει στη διαχείριση κίνησης υψηλής ταχύτητας σε πραγματικό χρόνο.

Για να αξιολογήσουν την τεχνική τους, οι συγγραφείς χρησιμοποίησαν τα 1999 MIT Lincoln Labs δεδομένα και, πιο συγκεκριμένα, τα δεδομένα για FTP, HTTP και SMTP, λόγω της συχνότητάς τους, μαζί με κάποιους άλλους τύπους (Telnet, ftp κτλ.). Τα δεδομένα αυτά περιλαμβάνουν ημερήσια tcpdump[71] αρχεία πολλών εβδομάδων για το 1998 και 1999, με κάθε αρχείο να περιέχει μέχρι και 1 εκατομμύριο TCP εγγραφές. Επειδή όμως το σύνολο αυτών των δεδομένων είναι προσομοιωμένο και όχι πραγματικό, οι συγγραφείς σημείωσαν ότι το εργαλείο τους μπορεί να έχει διαφορετική συμπεριφορά σε πραγματικό περιβάλλον. Χρησιμοποιήθηκε η απόσταση Mahalanobis για να μετρηθεί η απόσταση μεταξύ των παρακάτω επιθέσεων και των ομαλών συστάδων, με αποτέλεσμα, αφενός, οι TCP επιθέσεις Portsweep, Satan και Neptune να έχουν αρκετά μεγάλη απόσταση από τις ομαλές συστάδες και, κατά συνέπεια, να χαρακτηριστούν εύκολα ως ανωμαλίες, και, αφετέρου, το Mailbomb να ταιριάζει με κάποιες από τις παραγόμενες ομαλές συστάδες, χωρίς σημαντική διαφορά στην απόσταση.

---

[67] Τα συστήματα ανίχνευσης εισβολών δικτύου επικεντρώνονται στην επισκεψιμότητα του δικτύου και στοχεύουν σε εξωτερικές από το σύστημα επιθέσεις που προέρχονται από το δίκτυο.

[68] Στα δίκτυα υπολογιστών, μέτρηση της κυκλοφορίας δικτύου ονομάζεται η διαδικασία μέτρησης της ποσότητας και του είδους της κυκλοφορίας σε ένα συγκεκριμένο δίκτυο, κάτι που σχετίζεται άμεσα με την αποτελεσματική διαχείριση του εύρους ζώνης (bandwidth).

[69] Στα δίκτυα υπολογιστών, πακέτο ορίζεται ως η βασική μονάδα πληροφορίας που μεταφέρεται μέσα στα δίκτυα.

[70] Η συσταδοποίηση (cluster analysis) είναι μία τεχνική πολλαπλών μεταβλητών που χρησιμοποιείται για την εύρεση ομάδων σε παρατηρούμενα δεδομένα. Ο σκοπός είναι ο σχηματισμός ομάδων με τέτοιον τρόπο ώστε τα αντικείμενα της κάθε ομάδας να είναι παρόμοια μεταξύ τους και όσο το δυνατόν περισσότερο διαφορετικά από τα αντικείμενα των άλλων ομάδων.

[71] Το tcpdump είναι ένας κοινός αναλυτής πακέτων που εκτελείται από το command line. Επιτρέπει στο χρήστη την εμφάνιση των TCP/IP και άλλων πακέτων που μεταφέρονται ή εισέρχονται σε ένα δίκτυο, στο οποίο είναι συνδεδεμένος ο υπολογιστής.

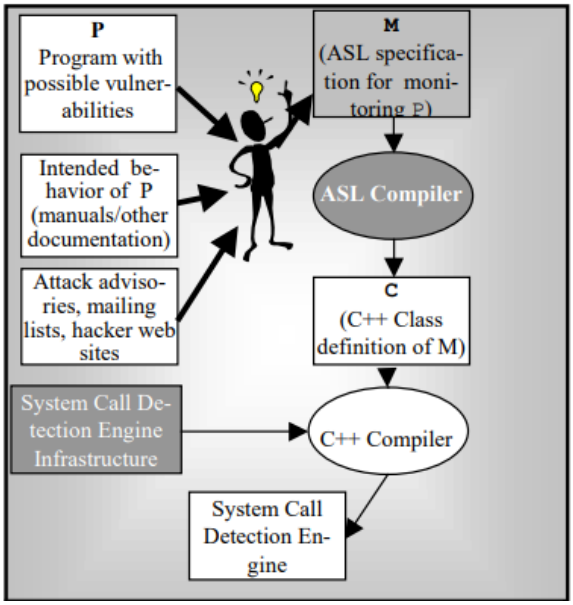
2.1.18. Πρόληψη Εισβολών Μέσω της Παρακολούθησης της Συμπεριφοράς των Διεργασιών

Οι R. Sekar, T. Bowen και M. Segal στο έργο τους [18] προσπαθούν να αντιμετωπίσουν κάποια από τα βασικά προβλήματα των συστημάτων που δέχονται επίθεση όπως:

- Ανίχνευση των επιθέσεων πριν αυτές προκαλέσουν ζημιά,
- Περιορισμός και ελαχιστοποίηση της ζημιάς μέσω της απομόνωσης των κακόβουλων στοιχείων σε πραγματικό χρόνο, και
- Εντοπισμός της προέλευσης της επίθεσης

Οι συγγραφείς, λοιπόν, αντιμετωπίζουν το θέμα της ανίχνευσης μέσω της παρακολούθησης των γεγονότων[72] σε πραγματικό χρόνο και της σύγκρισης αυτών με γεγονότα που θεωρούνται μη αποδεκτά. Αντίστοιχα, το πρόβλημα της απομόνωσης και του εντοπισμού διευθετείται μέσω κατάλληλα σχεδιασμένων προγραμμάτων, ονόματι reactions, που σκοπό έχουν να απομονώνουν τα πλέον μολυσμένα στοιχεία και να τα εμποδίζουν από το να μολύνουν ακόμα περισσότερα. Ταυτόχρονα, ένας τρόπος που βοηθούν στον εντοπισμό της προέλευσης της επίθεσης είναι με το να δίνουν την ψευδαίσθηση στον επιτιθέμενο ότι η επίθεση εκτελέστηκε με επιτυχία, οπότε η επίθεση να συνεχίζεται κανονικά, ενώ στην πραγματικότητα δεν προκαλεί καμία ζημιά. Αν και η προσέγγιση αυτή μπορεί να εφαρμοσθεί σε οποιοδήποτε σύγχρονο λειτουργικό σύστημα, εντούτοις υλοποιήθηκε συγκεκριμένα για το λειτουργικό σύστημα Linux.

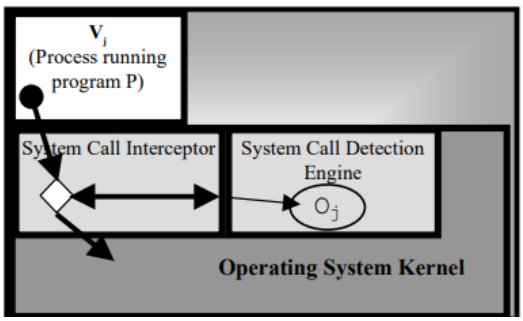
Το σύστημα μοντελοποιείται ως ένα κατανεμημένο σύστημα που αποτελείται από εξυπηρετητές (hosts), διασυνδεδεμένους μέσω δικτύου που είναι συνδεδεμένο στο Διαδίκτυο, από όπου και επιχειρούνται ασύρματα οι επιθέσεις. Ο τρόπος ανίχνευσης που προτείνεται βασίζεται στην εύρεση των αποδεκτών συμπεριφορών των διαδικασιών που εκτελούνται από τους υπολογιστές του συστήματος. Οι συμπεριφορές αυτές καθορίζονται ως λογικοί ισχυρισμοί στις ακολουθίες κλήσεων συστήματος, κάτι που γίνεται μέσω της υψηλού επιπέδου γλώσσας προδιαγραφών ASL (Auditing Specification Language). Οι προδιαγραφές M της ASL γλώσσας σχεδιάζονται από το διαχειριστή ασφαλείας του συστήματος και μεταγλωττίζονται σε βελτιστοποιημένα προγράμματα με σκοπό την αποτελεσματική ανίχνευση παρεκκλίσεων από την καθορισμένη συμπεριφορά. Οι προδιαγραφές απεικονίζονται μέσω της χρήσης μη-ντετερμινιστικού εκτεταμένου πεπερασμένου αυτομάτου (EFSA). Η ASL επίσης περιγράφει αυτόματες αμυντικές ενέργειες περιορισμού ή απομόνωσης της ζημιάς, σε περίπτωση που βρεθούν ανακολουθίες κατά τη διάρκεια εκτέλεσης. Κατά την offline φάση (Εικόνα 18), οι προδιαγραφές μεταγλωττίζονται σε C++ κλάση (μία κλάση για κάθε προδιαγραφή), η οποία μεταγλωττίζεται με τη σειρά της και συνδέεται με μία υποδομή που συλλαμβάνει κλήσεις συστήματος. Αυτό δημιουργεί ένα εκτελέσιμο το οποίο αναφέρεται ως μηχανή ανίχνευσης των κλήσεων συστήματος και το οποίο λειτουργεί σε επίπεδο πυρήνα.



Εικόνα 17 - Παραγωγή μηχανών ανίχνευσης (offline)

[72] Ως γεγονότα αναφέρονται εδώ οι κλήσεις συστήματος που καλούνται από την παρακολουθούμενη διαδικασία και στα οποία αποθηκεύονται το όνομα και οι παράμετροί τους.. Σε κάθε κλήση συστήματος αντιστοιχούν δύο γεγονότα και, πιο συγκεκριμένα, η είσοδος και η έξοδος από την κλήση συστήματος.

Κάθε κλήση που καλείται κατά τη διάρκεια εκτέλεσης (Εικόνα 19) συλλαμβάνεται από τον Interceptor και στέλνεται στη μηχανή ανίχνευσης και αναχαιτίζεται αμέσως πριν και αμέσως μετά την εκτέλεση των λειτουργιών (ή συμπεριφορών) του πυρήνα. Μία μηχανή ανίχνευσης συγκρίνει την κλήση συστήματος που καλείται με τις προδιαγραφές που έχουν αρχικά διαμορφωθεί από τη ASL γλώσσα του προγράμματος υπό εξέταση.



Εικόνα 18 - Εκτέλεση των μηχανών

Συνολικά, τα βασικά χαρακτηριστικά της προσέγγισης αυτής συνοψίζονται στα εξής:

- *Πρόληψη*: δίνει τη δυνατότητα στο πρόγραμμα να συνεχίσει την εκτέλεσή του, ακόμα και όταν είναι γνωστό ότι περιέχει αδυναμίες που μπορούν πιθανώς να τις εκμεταλλευτούν.
- *Προγραμματισιμότητα*: δίνει τη δυνατότητα στο διαχειριστή του συστήματος να δράσει άμεσα σε περίπτωση ανακάλυψης καινούριας αδυναμίας.
- *Αυτόματη αντίδραση*: οι δυνατότητες ανίχνευσης και αντίδρασης περιλαμβάνονται στην ίδια προδιαγραφή, προσφέροντας μία πιο αποδοτική λύση.
- *Εξαπάτηση*: μέσω της εξαπάτησης του επιτιθέμενου στο να πιστεύει ότι η απόπειρά του ήταν επιτυχής, είναι δυνατή η παρακολούθηση και καταγραφή της συμπεριφοράς του, καθώς και η καλύτερη κατανόηση των αδυναμιών του συστήματος.
- *Δυναμικώς ρυθμιζόμενη παρακολούθηση*: ο βαθμός λεπτομέρειας της παρακολούθησης μπορεί να αυξηθεί δυναμικά κατά τη διάρκεια εκτέλεσης με το που ανιχνευθούν λάθη ή ύποπτες δραστηριότητες.

Δε δόθηκαν εμπειρικά στοιχεία στη δημοσίευση αυτή, ενώ οι συγγραφείς συνεχίζουν την έρευνα προς βελτιστοποίηση της γλώσσας προδιαγραφών τους και της ανάπτυξης κατάλληλου αλγορίθμου για τη μεταγλώττιση των προδιαγραφών σε ντετερμινιστικό πλέον εκτεταμένο πεπερασμένο αυτόματο. Τέλος, βρίσκονται στη διαδικασία ανάπτυξης μεσαίας προς μεγάλης κλίμακας πειραμάτων, με σκοπό να αξιολογήσουν τις επιπτώσεις απόδοσης της online μεθόδου παρακολούθησής τους.

2.1.19. Τεχνικές Εξόρυξης Δεδομένων για την Ανίχνευση Εισβολών

Οι Wenke Lee και Salvatore J. Stolfo στο έργο τους [19] προτείνουν τη χρήση δύο τεχνικών εξόρυξης δεδομένων[73]: κανόνων συσχέτισης[74] και συχνών επεισοδίων[75], για περιγραφή της συμπεριφοράς προγραμμάτων ή χρηστών, μέσω του υπολογισμού των μοτίβων εντός και μεταξύ αρχείων ελέγχου. Τα ανακαλυφθέντα μοτίβα μπορούν να καθοδηγήσουν τη διαδικασία συλλογής δεδομένων ελέγχου και να διευκολύνουν την επιλογή των χαρακτηριστικών. Για την αντιμετώπιση των προκλήσεων της αποτελεσματικής μάθησης και της ανίχνευσης σε πραγματικό χρόνο, οι συγγραφείς προτείνουν στο τέλος μία αρχιτεκτονική βασισμένη σε πράκτορες[76] για τα συστήματα ανίχνευσης εισβολών, όπου οι πράκτορες μάθησης κάνουν συνεχώς υπολογισμούς και παρέχουν τα ενημερωμένα μοντέλα ανίχνευσης στους πράκτορες ανίχνευσης.

Πιο αναλυτικά, οι συγγραφείς διενήργησαν πειράματα σε *sendmail* ίχνη δεδομένων, ομαλών και μη[77], για την κατασκευή των μοντέλων ταξινόμησης[78] των ακολουθιών κλήσεων συστήματος, προς ανίχνευση ανωμαλιών. Στα δεδομένα αυτά εφαρμόστηκαν τεχνικές μηχανικής μάθησης με σκοπό την παραγωγή ταξινομητών που θα μπορούσαν να διαχωρίσουν τις αδυναμίες του συστήματος από τις ομαλές εκτελέσεις. Τα δεδομένα της φάσης εκμάθησης (training phase) περιείχαν ομαλές και μη-ομαλές ακολουθίες. Σε περίπτωση ακριβής αντιστοιχίας μεταξύ της ακολουθίας (στα δεδομένα υπό εξέταση) και μιας ακολουθίας στην ομαλή λίστα, τότε αυτή χαρακτηριζόταν ως ομαλή, αλλιώς μη-ομαλή. Στα δεδομένα αυτά εφαρμόστηκε, επίσης, ένα πρόγραμμα εκμάθησης κανόνων, ονόματι RIPPER, οι κανόνες του οποίου, 200-280 στο σύνολο, χρησιμοποιήθηκαν για να προβλέψουν κατά πόσο μία ακολουθία είναι ομαλή ή όχι. Αν περισσότερες από κάποιο καθορισμένο-από-το-χρήστη-όριο προβλέψεις βρέθηκαν να είναι μη-ομαλές, τότε όλη η περιοχή θεωρήθηκε μη-ομαλή. Εν συνεχεία, εάν το ποσοστό των μη-ομαλών περιοχών ήταν μεγαλύτερο από ένα συγκεκριμένο όριο, για παράδειγμα 2%, τότε όλο το ίχνος θεωρήθηκε ότι είναι εισβολή.

Οι συγγραφείς διενήργησαν επίσης πειράματα χρησιμοποιώντας *tcpdump*[79] δεδομένα, τα οποία περιλάμβαναν μόνο TCP (Transmission Control Protocol) και UDP (User Datagram Protocol) πακέτα, για να μάθουν τι ήταν φυσιολογικό και τι όχι για τις εισερχόμενες συνδέσεις. Κάθε σύνδεση ανήκει σε μία από τις τρεις κατηγορίες: εξερχόμενη (*out-going*), εισερχόμενη (*in-going*) ή εσωτερική (*inter-LAN*), ενώ τα χαρακτηριστικά της είναι τα εξής: ώρα έναρξης, διάρκεια, εξυπηρετητές που λαμβάνουν μέρος, πύλες, διάφορα στατιστικά της σύνδεσης (όπως τα bytes που στέλνονται στην καθεμία κατεύθυνση κτλ.), πρωτόκολλο (TCP ή UDP) και ο χαρακτηρισμός ως ομαλής ή κάποιου λάθους σύνδεσης. Εφαρμόστηκε και εδώ το πρόγραμμα RIPPER για την κατηγοριοποίηση των μοτίβων σε ομαλά και μη, ανάλογα με τα χαρακτηριστικά της σύνδεσης. Σε περίπτωση που ο ταξινομητής προέβλεπε έναν προορισμό, με βάση τα χαρακτηριστικά της σύνδεσης, που είναι διαφορετικός από τον πραγματικό, σε κάποιο συγκεκριμένο χρονικό διάστημα, τότε η κατάσταση αυτή θεωρείτο ως εσφαλμένη κατηγοριοποίηση (misclassification), με το ποσοστό αυτών των

[73] Γενικά, η εξόρυξη δεδομένων αναφέρεται στη διαδικασία της (αυτόματης) εξαγωγής μοντέλων από μεγάλες αποθήκες δεδομένων.

[74] Στην επιστήμη της εξόρυξης δεδομένων, κανόνες συσχέτισης ονομάζονται if / then (αν / τότε) δηλώσεις (statements) που βοηθούν στην αποκάλυψη των σχέσεων μεταξύ φαινομενικά άσχετων μεταξύ τους δεδομένων σε μία σχεσιακή βάση δεδομένων ή άλλο αποθετήριο πληροφοριών.

[75] Στην επιστήμη της εξόρυξης δεδομένων, συχνά επεισόδια ονομάζονται συλλογές γεγονότων που συμβαίνουν συχνά μαζί. Πιο γενικά, τα επεισόδια είναι μερικώς ταξινομημένα σύνολα γεγονότων.

[76] Στην επιστήμη των υπολογιστών, πράκτορας θεωρείται ένα πρόγραμμα υπολογιστή που ενεργεί εκ μέρους κάποιου άλλου προγράμματος ή χρήστη.

[77] Τα μη ομαλά ίχνη περιλάμβαναν: 3 ίχνη από επιθέσεις *sscp* (*sunsendmailcp*), 2 ίχνη από επιθέσεις *syslog-remote,* 2 ίχνη από επιθέσεις *syslog-local,* 2 ίχνη από επιθέσεις *decode,* 1 ίχνος από επιθέσεις *sm5x και* 1 ίχνος από επιθέσεις *sm565a.*

[78] Κατά την ταξινόμηση, κάθε δεδομένο αντιστοιχίζεται σε μία από τις υπάρχουσες, προκαθορισμένες, κατηγορίες.

[79] Το tcpdump είναι ένας κοινός αναλυτής πακέτων που εκτελείται από το command line. Επιτρέπει στο χρήστη την εμφάνιση των TCP/IP και άλλων πακέτων που μεταφέρονται ή εισέρχονται σε ένα δίκτυο, στο οποίο είναι συνδεδεμένος ο υπολογιστής.

καταστάσεων να είναι πολύ χαμηλό στα κανονικά δεδομένα σύνδεσης και πολύ υψηλό σε δεδομένα εισβολής.

Κάθε ένας από τους ταξινομητές που περιγράφτηκαν παραπάνω μοντελοποιεί μία συγκεκριμένη πλευρά της συμπεριφοράς του συστήματος και, ως εκ τούτου, ονομάζεται ταξινομητής βάσης (ή απλού επιπέδου), παίρνοντας ως είσοδο δεδομένα ελέγχου (audit data)[80]. Οι συλλογικές προβλέψεις από πολλούς τέτοιους ταξινομητές, ο καθένας από τους οποίους μοντελοποιεί μία διαφορετική πλευρά του συστήματος υπό εξέταση, βοηθάει στη βελτίωση της αποτελεσματικότητας του συστήματος ανίχνευσης. Κάθε καταχώρηση στα δεδομένα εκμάθησης είναι μία συλλογή αυτών των προβλέψεων, που παρήχθησαν την ίδια χρονική στιγμή, από τους ταξινομητές βάσης. Κάθε χαρακτηριστικό της εκάστοτε καταχώρησης παίρνει τιμή 1 ή 0, ανάλογα με το αν η πρόβλεψη του ταξινομητή βάσης για την μοντελοποιημένη συμπεριφορά ήταν «ομαλή» ή «ανώμαλη» αντίστοιχα (αν, δηλαδή, υπήρχε αντιστοιχία με το μοντέλο ή όχι). Στο τέλος, εφαρμόζεται ένας αλγόριθμος εκμάθησης με σκοπό την παραγωγή του ταξινομητή μετά-δεδομένων, ο οποίος δέχεται σαν είσοδο τις προαναφερθείσες προβλέψεις από τους ταξινομητές βάσης και εξάγει μία τελική εκτίμηση.

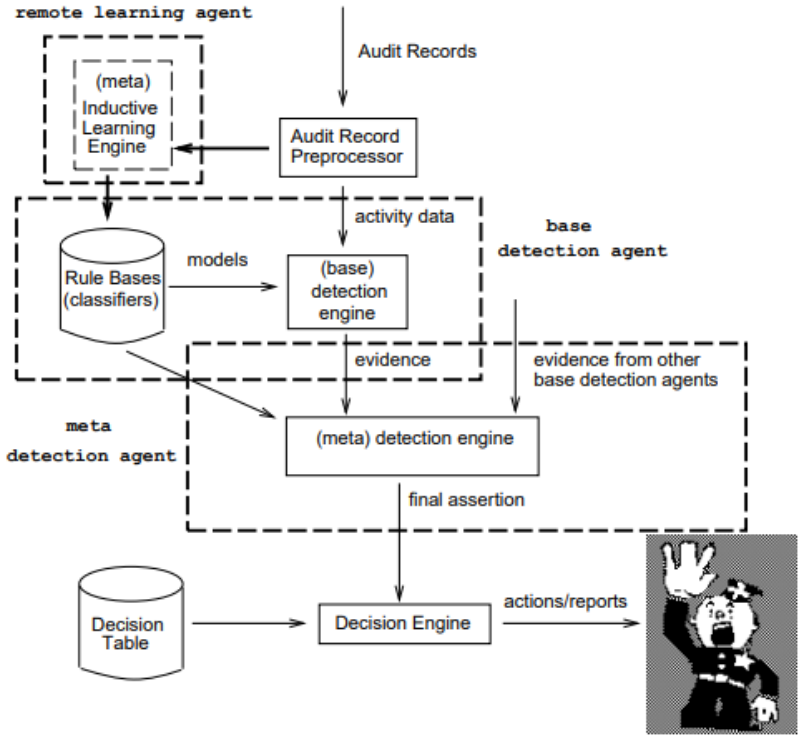
Για την κατασκευή ενός αποτελεσματικού ταξινομητή βάσης, χρειάζεται η συλλογή μίας επαρκούς ποσότητας από δεδομένα εκμάθησης καθώς και η επιλογή των σημαντικών χαρακτηριστικών. Για τους σκοπούς αυτούς, χρησιμοποιούνται αλγόριθμοι εξόρυξης δεδομένων, όπως των κανόνων συσχέτισης και των συχνών επεισοδίων. Ο πρώτος ψάχνει για μοτίβα εντός των δεδομένων ελέγχου, ενώ ο δεύτερος μεταξύ τους. Ο στόχος των κανόνων συσχέτισης είναι να αποκομίσουν σχέσεις πολλαπλών χαρακτηριστικών από έναν πίνακα βάσης δεδομένων, όπου κάθε γραμμή είναι ένα αρχείο ελέγχου και κάθε στήλη ένα χαρακτηριστικό του αρχείου. Τα συχνά επεισόδια είναι ένα σύνολο από γεγονότα που συμβαίνουν συχνά μέσα σε κάποιο συγκεκριμένο χρονικό διάστημα. Τα γεγονότα αυτά πρέπει να συμβούν μαζί με τουλάχιστον μία προκαθορισμένη ελάχιστη συχνότητα. Υπάρχει, επίσης, διαχωρισμός μεταξύ διαδοχικών (serial) και παράλληλων (parallel) επεισοδίων, όπου τα γεγονότα στα πρώτα πρέπει να συμβούν με κάποια συγκεκριμένη χρονική σειρά, ενώ στα δεύτερα δεν υπάρχει κάποιος αντίστοιχος περιορισμός.

Οι κανόνες συσχέτισης και τα συχνά επεισόδια μπορούν να αναφερθούν συλλογικά ως σύνολο κανόνων. Ένα πρόγραμμα εκτελείται πολλές φορές και με διαφορετικές ρυθμίσεις κάθε φορά και, για κάθε καινούρια εκτέλεση, υπολογίζεται το σύνολο κανόνων του, το οποίο συγχωνεύεται στο ήδη συνολικό (υπάρχον). Σε περίπτωση κανόνα που υπάρχει ήδη, αυξάνεται ο μετρητής του, ενώ κανόνες που δεν υπήρχαν πριν, προστίθεται στο σύνολο. Όταν δεν υπάρχουν πλέον καινούριοι κανόνες να προστεθούν, σταματάει η διαδικασία συλλογής δεδομένων, ενώ οι κανόνες με χαμηλό μετρητή, με βάση κάποιο όριο που καθορίζεται από το χρήστη, εξαλείφονται από το σύνολο. Στη συνέχεια, χρησιμοποιούνται πληροφορίες συσχέτισης για την επιλογή του υποσυνόλου των χαρακτηριστικών αυτών που κρίνονται ως σημαντικά για τις διαδικασίες ταξινόμησης. Τέλος, υπολογίζονται οι κανόνες συσχέτισης και τα συχνά επεισόδια στα καινούρια δεδομένα ελέγχου και συγκρίνονται με το υφιστάμενο σύνολο κανόνων.

Διενεργήθηκαν προκαταρκτικά πειράματα στα tcpdump δεδομένα για τις εισερχόμενες συνδέσεις, χρησιμοποιώντας τα προγράμματα κανόνων συσχέτισης και συχνών επεισοδίων για διάφορες ελάχιστες συχνότητες και χρονικά διαστήματα. Μέσω της χειρωνακτικής εξέτασης των ομαλών και μη-ομαλών δεδομένων, οι συγγραφείς συμπέραναν ότι η μέθοδός τους θα μπορούσε να χρησιμοποιηθεί για τον εντοπισμό μη-ομαλής δραστηριότητας.

[80] Στην προκειμένη περίπτωση, τα δεδομένα ελέγχου αναφέρονται σε γενικές ροές δεδομένων, οι οποίες έχουν κατάλληλα επεξεργαστεί για σκοπούς ανίχνευσης. Ένα παράδειγμα μίας τέτοιας ροής είναι τα δεδομένα σύνδεσης που εξάγονται από το ακατέργαστο αποτέλεσμα εξόδου του tcpdump.

Τέλος, οι συγγραφείς προτείνουν μία αρχιτεκτονική στήριξης του συστήματος βασισμένη σε δύο ειδών πράκτορες: τους πράκτορες εκμάθησης (learning agents) και τους πράκτορες ανίχνευσης (detection agents), όπως αναφέρθηκε και στην πρώτη παράγραφο. Οι πράκτορες εκμάθησης, που μπορεί να βρίσκονται εντός κάποιου εξυπηρετητή (server) λόγω της υπολογιστικής ισχύος του, είναι υπεύθυνοι για τους υπολογισμούς και τη συντήρηση των συνόλων των κανόνων για τα προγράμματα και τους χρήστες. Παράγουν τόσο τους ταξινομητές βάσης όσο και τους ταξινομητές μετά-δεδομένων. Οι πράκτορες ανίχνευσης είναι εξοπλισμένοι με το σύνολο κανόνων, το οποίο περιοδικά ενημερώνεται, που τους παρέχεται από τον αντίστοιχο πράκτορα εκμάθησης, και είναι επεκτάσιμοι. Οι μηχανές ανίχνευσής τους εκτελούν τον ταξινομητή που παίρνει ως είσοδο τα δεδομένα ελέγχου και εξάγει τις προβλέψεις για πιθανές εισβολές. Η βασική διαφορά μεταξύ των πρακτόρων ανίχνευσης βάσης (base detection agent) και των πρακτόρων ανίχνευσης μετά-δεδομένων (meta detection agent), οι οποίοι δε χρειάζεται να τρέχουν στον ίδιο εξυπηρετητή (host), είναι ότι οι πρώτοι χρησιμοποιούν σαν είσοδο προ-επεξεργασμένα δεδομένα ελέγχου ενώ οι δεύτεροι χρησιμοποιούν τις προβλέψεις που παράγονται από όλους τους πρώτους. Η αρχιτεκτονική αυτή παρουσιάζεται σχηματικά στη δίπλα εικόνα.



Εικόνα 19 - Αρχιτεκτονική του Συστήματος Ανίχνευσης Εισβολών βασισμένου στους Πράκτορες

2.1.20. Βελτιωμένη Μέθοδος Ανίχνευσης Εισβολών βάσει Σκιαγράφησης των Διαδικασιών

Οι Izuru Sato, Yoshinori Okazaki και Shigeki Goto στο έργο τους [20] προτείνουν μία μέθοδο ανίχνευσης εισβολών βασισμένη στη συχνότητα των κλήσεων συστήματος. Συγκεντρώνονται κλήσεις συστήματος από μία διαδικασία και, βάσει αυτών, κατασκευάζεται το προφίλ της διαδικασίας. Καθώς σε μία μη εξουσιοδοτημένη πρόσβαση μπορεί να υπάρχουν διαφορές στη σειρά και τη συχνότητα των κλήσεων, καταγράφονται οι κλήσεις συστήματος που καλούνται υπό κανονικές συνθήκες με σκοπό να συγκριθούν με τις αντίστοιχες κλήσεις συστήματος που καλούνται από το υπό εξέταση πρόγραμμα / διαδικασία.

Στη μέθοδο αυτή κατασκευάζεται το λεγόμενο προφίλ (*profile*) για την κάθε διαδικασία, βάσει της ακολουθίας και συχνότητας των κλήσεων συστήματος που καλούνται από αυτήν. Το περιβάλλον που χρησιμοποιήθηκε για την κατασκευή του κανονικού προφίλ (*normal profile*) αποτελείται από έναν κεντρικό υπολογιστή που χρησιμοποιεί το λειτουργικό σύστημα Solaris 2.5.1, μαζί με άλλους οκτώ, στο ίδιο δίκτυο, στο Waseda University, οι πέντε από τους οποίους χρησιμοποιούν Solaris 7 και οι υπόλοιποι FreeBSD 3.4-RELEASE λειτουργικό σύστημα. Στη συνέχεια, γίνεται καταγραφή της ακολουθίας των κλήσεων συστήματος σε κανονικές συνθήκες χρήσης, όπως αναφέρθηκε και στην εισαγωγή. Το σύνολο των ενεργειών αυτών καλείται προφίλ της διαδικασίας (*process profile*) και αποτελείται από τρεις τύπους προφίλ:

- Βασικό προφίλ (***Base profile***)
- SUID προφίλ (***SUID profile***)
- Προφίλ δαίμονα (***Daemon profile***)

Το βασικό προφίλ είναι μία συλλογή από κλήσεις συστήματος υπό κανονικές συνθήκες. Στο προφίλ αυτό καταγράφονται ο τύπος και ο αριθμός κατάταξης της κάθε κλήσης με βάση τη συχνότητα εμφάνισης[81]. Όσο πιο μικρός είναι ο αριθμός αυτός, τόσο πιο συχνά εμφανίζεται η κλήση αυτή στη διαδικασία και, αντίστοιχα, όσο πιο μεγάλος είναι, τόσο πιο σποραδικά εμφανίζεται η κλήση. Γύρω στα 30 λεπτά είναι αρκετά για την καταγραφή επαρκούς δείγματος από κλήσεις συστήματος, ενώ, το ίδιο το βασικό προφίλ μπορεί να χρησιμοποιηθεί από διαφορετικούς υπολογιστές, που όμως χρησιμοποιούν το ίδιο λειτουργικό σύστημα, χωρίς να χρειάζεται ενημερώσεις. Το βασικό χρησιμοποιείται, επίσης, στην κατασκευή των δύο επόμενων προφίλ.

Το SUID προφίλ είναι μια καταγραφή από κλήσεις συστήματος. Η καταγραφή ξεκινάει όταν εκτελείται το πρόγραμμα suid μέσω της κλήσης συστήματος execve(). Το προφίλ αυτό είναι μια ακολουθία από τις εμφανίσεις των κλήσεων συστήματος του προγράμματος υπό κανονικές συνθήκες. Κάθε κλήση μεταμορφώνεται σε έναν αριθμό, που υποδεικνύεται στο βασικό προφίλ. Στην περίπτωση διαδικασιών που δέχονται είσοδο από το χρήστη, όπως το *rlogin* πρόγραμμα που περιμένει το χρήστη να εισάγει κωδικό, το προφίλ τελειώνει όταν το πρόγραμμα περιμένει την είσοδο του χρήστη. Στην περίπτωση διαδικασιών που τερματίζουν αυτόματα, η κατασκευή του προφίλ σταματάει όταν καλείται η κλήση συστήματος *_exit()*. Όπως και τα βασικά προφίλ, τα SUID προφίλ μπορούν να μοιραστούν μεταξύ υπολογιστών που χρησιμοποιούν το ίδιο λειτουργικό σύστημα, ενώ δε χρειάζονται συχνές ενημερώσεις.

[81] Για την απόδοση αριθμών στις κλήσεις συστήματος, δοκιμάστηκαν και αξιολογήθηκαν τρεις τρόποι. Ο πρώτος ήταν η ανάθεση του πραγματικού αριθμού της κλήσης από το λειτουργικό σύστημα (π.χ. fork = 2 στο λειτουργικό σύστημα SunOS). Ο δεύτερος ήταν βάσει της συχνότητας εμφάνισης των κλήσεων στο πρόγραμμα, όπως τους έχει αποδοθεί από το βασικό προφίλ, και είναι αυτός που χρησιμοποιήθηκε τελικά, ενώ ο τρίτος ήταν η ανάθεση τυχαίων αριθμών.

Το προφίλ δαίμονα καταγράφει ακολουθίες κλήσεων συστήματος για τους δαίμονες[82] του προγράμματος, ενώ χρησιμοποιεί τις αδρανείς καταστάσεις για την οριοθέτηση των προφίλ. Η καταγραφή ξεκινάει όταν ο πελάτης (client) συνδεθεί στο δαίμονα.

Μόλις καθοριστεί το προφίλ της διαδικασίας, που περιλαμβάνει τα τρία επιμέρους προφίλ που είδαμε παραπάνω, πρέπει να μετρηθεί η απόσταση (*distance*) μεταξύ του προφίλ και των δεδομένων του δείγματος, δηλαδή των δεδομένων του υπό εξέταση προγράμματος κατά τη διάρκεια εκτέλεσης. Η τιμή της απόστασης καλείται σκορ (*score*) και εξυπηρετεί ως δείκτης της κακόβουλης συμπεριφοράς της διαδικασίας, όταν συγκρίνεται με το συνολικό αριθμό κλήσεων συστήματος του προγράμματος. Κατά την κατασκευή του προφίλ, καταγράφονται περίπου 80 με 90 κλήσεις συστήματος. Όταν, λοιπόν, το σκορ είναι μικρότερο του 25, που αντιστοιχεί περίπου στο ¼ του συνολικού αριθμού των κλήσεων, η διαδικασία βρίσκεται σε κανονική κατάσταση (*normal state*). Αν το σκορ είναι μεταξύ 25 και 50, δηλαδή μεγαλύτερο από το ¼ και μικρότερο από το ½ του συνολικού αριθμού των κλήσεων, η διαδικασία βρίσκεται σε κατάσταση προσοχής (*caution state*). Τέλος, αν το σκορ είναι μεγαλύτερο από 50, μεγαλύτερο από το ½ δηλαδή, τότε η διαδικασία θεωρείται ότι βρίσκεται σε κατάσταση ανίχνευσης (*detect state*).

Επιπλέον, γίνεται χρήση ενός βέλτιστου αλγορίθμου αντιστοίχισης μοτίβων, που ονομάζεται *DP matching*, με σκοπό τη μέτρηση των αποστάσεων αυτών και την εξάλειψη των ψευδώς θετικών συναγερμών[83]. Ο αλγόριθμος αυτός βρίσκει την καλύτερη αντιστοίχιση όταν τα συγκρινόμενα μοτίβα αποτελούνται από διαφορετικό αριθμό στοιχείων, χωρίς να μειώνει την ικανότητα ανίχνευσης εισβολών. Χρησιμοποιώντας τη μέθοδο των συγγραφέων σε συνδυασμό με τον DP matching αλγόριθμο, οι απόπειρες εισβολής στις εντολές *rlogin*[84], *eject* και *portscan*[85] ανιχνεύθηκαν με επιτυχία. Ωστόσο, η μέθοδος αυτή δεν κατάφερε να εντοπίσει εισβολές στο *Qpopper*[86]. Να σημειωθεί ότι σε όλες τις παραπάνω περιπτώσεις, εκτός της εντολής rlogin, δεν υπήρχαν μεγάλες διαφορές στις αποστάσεις με και χωρίς τη χρήση του DP matching αλγορίθμου και, ακόμα και χωρίς τη χρήση αυτού, η κατηγοριοποίηση παρέμεινε η ίδια. Τέλος, η μέθοδος, δοκιμάστηκε και σε σύνολο δεδομένων ανίχνευσης LPR από το Πανεπιστήμιο του Νέου Μεξικού, με και χωρίς τον DP matching αλγόριθμο, με επιτυχία και στις δύο περιπτώσεις, με μεγάλη, όμως, διαφορά στην απόσταση (σκορ ίσο με 138.30 χωρίς τον αλγόριθμο και 664.33 με τον αλγόριθμο).

[82] Ως δαίμονας (daemon) αναφέρεται ένα πρόγραμμα υπολογιστή που εκτελείται ως διαδικασία παρασκηνίου (background process), στην οποία ο χρήστης δεν έχει άμεσο έλεγχο. Ένας δαίμονας περιμένει για μία κλήση από τις εξωτερικές διαδικασίες για να αρχίσει τη δράση του.

[83] Χωρίς τη χρήση του αλγορίθμου αυτού, οι αξιολογήσεις του προγράμματος rlogin και της εντολής finger υπό κανονική χρήση έδωσαν πολύ μεγαλύτερα σκορ από το κανονικό, με αποτέλεσμα να καταταχθούν και οι δύο εσφαλμένα στην κατάσταση ανίχνευσης (detect state).

[84] Η εντολή rlogin επιτρέπει στο χρήστη να συνδεθεί μέσω δικτύου σε άλλον server, χρησιμοποιώντας την TCP πύλη δικτύου 513.

[85] Η διαδικασία portscan στέλνει μηνύματα σε διευθύνσεις πυλών σε πολλούς servers για να βρει κάποια ενεργή πύλη, με σκοπό τον καθορισμό των διαθέσιμων υπηρεσιών σε κάποιο απομακρυσμένο μηχάνημα. Παρόλο που δεν αποτελεί μη εξουσιοδοτημένη πρόσβαση, συνήθως ακολουθείται από απομακρυσμένες επιθέσεις.

[86] Το Qpopper, μία γνωστή εφαρμογή του POP – Post Office Protocol – το οποίο αποτελεί πρωτόκολλο διαχείρισης των e-mails, ήταν ένας δωρεάν server ανοικτού κώδικα, ο οποίος πλέον δε συντηρείται και του οποίου η τελευταία έκδοση κυκλοφόρησε το 2011 (έκδοση 4.1.0). Αποτελούσε την κοινή επιλογή για τους παρόχους internet, για επιχειρήσεις, σχολεία και άλλους οργανισμούς.

2.1.21. Παρακολούθηση Κρίσιμων για την Ασφάλεια Προγραμμάτων σε Κατανεμημένα Συστήματα

Οι Calvin Ko, Manfred Ruschitzka και Karl Levitt στο έργο τους [21] προτείνουν μία μέθοδο βάσει προδιαγραφών (specification-based method[87]) για τον εντοπισμό εκμετάλλευσης των αδυναμιών ενός κρίσιμου για την ασφάλεια προγράμματος (security-critical program), όπως είναι τα privileged[88] προγράμματα. Η μέθοδος αυτή χρησιμοποιεί προδιαγραφές ασφαλείας που περιγράφουν την επιθυμητή συμπεριφορά των προγραμμάτων, όσον αφορά το κομμάτι της ασφάλειας, και ανιχνεύουν τα ίχνη ελέγχου για τις λειτουργίες που παραβιάζουν τις προδιαγραφές αυτές. Πλευρές της συμπεριφοράς των προγραμμάτων σχετικές με την ασφάλεια είναι οι ακόλουθες:

- Προσβάσεις σε αντικείμενα του συστήματος (π.χ. αρχεία)
- Ακολουθία των λειτουργιών που εκτελούνται από το πρόγραμμα
- Σωστός συγχρονισμός σε κατανεμημένα συστήματα[89]
- Race conditions[90]

Οι συγγραφείς ανέπτυξαν ένα επίσημο πλαίσιο για τον καθορισμό της σχετικής-με-την-ασφάλεια συμπεριφοράς των προγραμμάτων, όπου βασίστηκε ο σχεδιασμός και η υλοποίηση ενός συστήματος ανίχνευσης εισβολών (DPEM) σε πραγματικό χρόνο για ένα κατανεμημένο σύστημα. Δημιούργησαν, επίσης, πολιτικές ασφαλείας για 15 setuid[91] Unix προγράμματα, συμπεριλαμβανομένων των *fingerd*, *rdist*, *sendmail*, *binmail*, *passwd* και *vi*. Πιο συγκεκριμένα, το πρόγραμμα *rdist*[92] μπορεί να επιτρέψει σε έναν πιθανό εισβολέα να αποκτήσει πρόσβαση επιπέδου root, δίνει δηλαδή τη δυνατότητα σε οποιονδήποτε χρήστη χωρίς δικαιώματα (non-privileged user) να αλλάξει τα δικαιώματα λειτουργίας οποιουδήποτε αρχείου στο σύστημα. Το σύστημά τους κάνει χρήση των ιχνών (*traces*), χρονικά διατεταγμένων, δηλαδή, ακολουθιών από γεγονότα[93] εκτέλεσης, και ανιχνεύει επιθέσεις που προκαλούνται από καταγεγραμμένα προγράμματα, συμπεριλαμβανομένων παραβιάσεων ασφαλείας που προκαλούνται από λανθασμένο συγχρονισμό στα κατανεμημένα προγράμματα. Τα ίχνη λαμβάνονται από τις καταγραφές ελέγχου (audit trails) σε πραγματικό χρόνο.

---

[87] Η ανίχνευση βάσει προδιαγραφών (specification-based detection) βασίζεται στις προδιαγραφές του προγράμματος οι οποίες περιγράφουν την επιθυμητή συμπεριφορά κρίσιμων για την ασφάλεια προγραμμάτων. Η παρακολούθηση της εκτέλεσης προγραμμάτων περιλαμβάνει την ανίχνευση παρεκκλίσεων της συμπεριφοράς τους από τις προδιαγραφές αυτές και όχι την παρακολούθηση εμφάνισης συγκεκριμένων μοτίβων επίθεσης.

[88] Στην επιστήμη των υπολογιστών, ως προνόμιο/δικαίωμα (privilege) ορίζεται η εξουσιοδότηση που έχει ο χρήστης σε ένα σύστημα για να εκτελέσει μία ενέργεια. Παραδείγματα διάφορων δικαιωμάτων περιλαμβάνουν την ικανότητα δημιουργίας ενός αρχείου σε ένα φάκελο, τη δυνατότητα ανάγνωσης ή διαγραφής ενός αρχείου, τη δυνατότητα πρόσβασης σε μία συσκευή κτλ.

[89] Ένα κατανεμημένο σύστημα (distributed system) αποτελείται από έναν αριθμό από εξυπηρετητές (hosts) που είναι συνδεδεμένοι μέσω δικτύου.

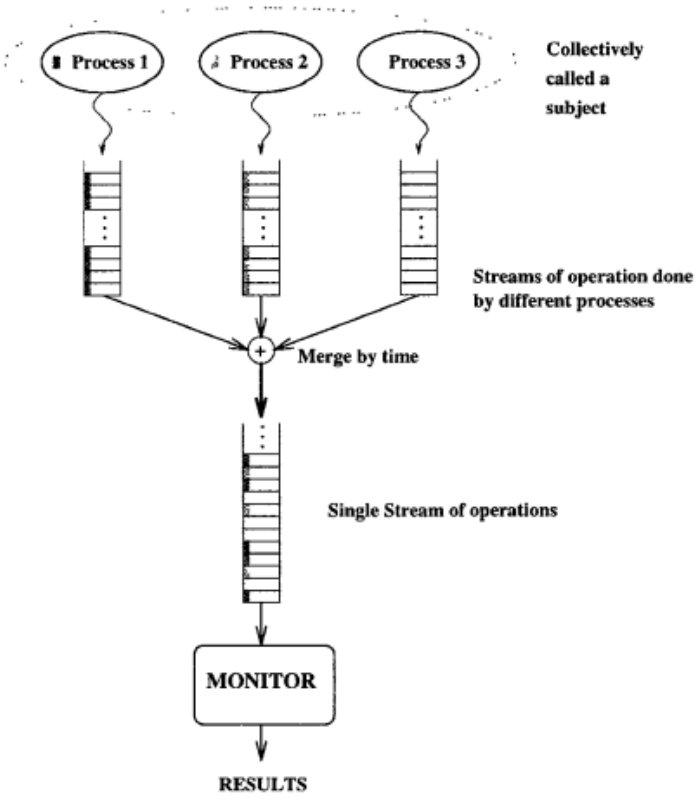
[90] Οι συνθήκες αυτές είναι ειδική κατηγορία των προβλημάτων συγχρονισμού και αναφέρονται στη συμπεριφορά ενός προγράμματος όπου το αποτέλεσμα εξαρτάται από την ακολουθία ή το συγχρονισμό άλλων γεγονότων που δεν μπορούν να ελεγχθούν. Όταν τα γεγονότα δε συμβαίνουν με τη σειρά που σκόπευε ο δημιουργός του προγράμματος, τότε αυτό αποτελεί σφάλμα του προγράμματος (bug).

[91] Το setuid είναι δείκτης δικαιωμάτων πρόσβασης στα λειτουργικά συστήματα Unix που επιτρέπει στους χρήστες να τρέξουν συγκεκριμένα προγράμματα με υψηλότερου επιπέδου προνόμια. Για παράδειγμα, μπορεί ο χρήστης να εκτελέσει ένα πρόγραμμα με δικαιώματα πρόσβασης ισοδύναμα με αυτά του κατόχου του προγράμματος.

[92] Το πρόγραμμα *rdist* (Remote File Distribution Program) είναι ένα βοηθητικό πρόγραμμα των Unix για τη διατήρηση πανομοιότυπων αντιγράφων των αρχείων σε πολλούς κεντρικούς υπολογιστές (hosts).

[93] Ένα τέτοιο γεγονός υποδηλώνει την εκτέλεση μιας λειτουργίας στο σύστημα και αποδίδεται στη διαδικασία που εκτελεί τη λειτουργία αυτή.

Η κάθε προδιαγραφή περιγράφει έγκυρες ακολουθίες λειτουργιών της εκτέλεσης ενός ή περισσοτέρων προγραμμάτων, οι οποίες συλλογικά ονομάζονται (παρακολουθούμενο) υποκείμενο (*monitored subject*). Στο επίπεδο του χρήστη, το υποκείμενο μπορεί να είναι μία ή περισσότερες εκτελέσεις του προγράμματος, ένας ή περισσότεροι χρήστες, κτλ. Στο επίπεδο του συστήματος, το υποκείμενο μπορεί να είναι μία ή περισσότερες διαδικασίες[94]. Η παρακολούθηση του υποκειμένου έγκειται στη συντακτική ανάλυση της ακολουθίας των λειτουργιών που εκτελούνται από αυτό. Στην Εικόνα 21 αναπαρίσταται η περίπτωση εκτέλεσης προγράμματος που αποτελείται από τρεις διεργασίες, η καθεμία από τις οποίες παράγει ένα ίχνος. Τα τρία ίχνη συγχωνεύονται για να σχηματίσουν ένα μοναδικό ίχνος, το οποίο θα λειτουργήσει ως είσοδο για το σύστημα παρακολούθησης (monitor).



Εικόνα 20 - Το Μοντέλο Παρακολούθησης

Μία ακολουθία λειτουργιών του υποκειμένου που δε συμμορφώνεται με την προδιαγραφή θεωρείται παραβίαση ασφαλείας. Καθώς η προδιαγραφή αποφασίζει κατά πόσο είναι έγκυρο το ίχνος εκτέλεσης ενός υποκειμένου, ονομάζεται πολιτική ίχνους (*trace policy*).

Οι έγκυρες ακολουθίες λειτουργιών των υποκειμένων καθορίζονται βάσει γραμματικών, των οποίων τα αλφάβητα αποτελούνται από λειτουργίες συστήματος. Οι συγγραφείς δημιούργησαν μία γραμματική παράλληλου περιβάλλοντος (*parallel environment grammar – PE grammar*) για τον καθορισμό των πολιτικών ίχνους και για να αντιμετωπίσουν προβλήματα συγχρονισμού στα κατανεμημένα προγράμματα. Η γραμματική αυτή περιλαμβάνει τέσσερα στοιχεία: το αλφάβητο, ένα σύνολο υπέρ-κανόνων (*hyperrules*-που είναι παραμετρικές εκδόσεις των κανόνων παραγωγής της γλώσσας), ένα σύνολο περιβαλλοντικών μεταβλητών και μία αρχική έκφραση (*start expression*).

Το σύστημα ανίχνευσης εισβολών DPEM (Distributed Program Execution Monitor)[95] παρακολουθεί τις εκτελέσεις των προγραμμάτων σε ένα κατανεμημένο σύστημα για την ανίχνευση αντιφατικής συμπεριφοράς με τις αντίστοιχες πολιτικές ίχνους. Χρησιμοποιώντας τις αντίστοιχες πολιτικές ίχνους, το πρωτότυπο DPEM δοκιμάστηκε[96] σε τρεις διαφορετικούς τύπους εισβολών, οι οποίες εκμεταλλεύονται αδυναμίες στα προγράμματα *rdist*, *sendmail* και *binmail*, και κατάφερε να εντοπίσει και τις τρεις σε 0,1 δευτερόλεπτο περίπου. Συγκεκριμένα για το πρόγραμμα *rdist*, το σήμα της παραβίασης παρουσιάστηκε περίπου 0,06 δευτερόλεπτα μετά την πραγματική επίθεση, ενώ

[94] Διαδικασίες (processes) είναι οι βασικές οντότητες οι οποίες εκτελούν λειτουργίες στα αντικείμενα του συστήματος (π.χ. σε 0

[95] Οι συγγραφείς ανέπτυξαν το πρωτότυπο σύστημα DPEM σε γλώσσα C. Το πρωτότυπο τρέχει σε λειτουργικό σύστημα Solaris 2.4 και χρησιμοποιεί υπηρεσίες ελέγχου του υποσυστήματος ελέγχου Sun BSM.

[96] Τα πειράματα διενεργήθηκαν σε υπολογιστή Sun SPARCstation 5 με 32MB μνήμη και λειτουργικό σύστημα Solaris 2.4, ενώ οι αρχικές εκδόσεις των *rdist* και *sendmail* αντικαταστάθηκαν από τις SUN 4.3 εκδόσεις, καθώς οι αδυναμίες τους έχουν αφαιρεθεί από το Solaris 2.4. Οι εισβολές πραγματοποιήθηκαν χρησιμοποιώντας συνδυασμό από Perl scripts και C προγράμματα.

παρόμοια καθυστέρηση (≈0,05 δευτερόλεπτα) παρουσιάστηκε και στην περίπτωση των προγραμμάτων *vi* και *passwd*.

Η προσέγγιση αυτή μπορεί να χρησιμοποιηθεί για την ανίχνευση τόσο γνωστών όσο και άγνωστων μέχρι στιγμής επιθέσεων. Μπορεί, επίσης, να εντοπίσει επιθέσεις που εκμεταλλεύονται άγνωστες μέχρι τώρα αδυναμίες σε κρίσιμα για την ασφάλεια προγράμματα. Ένας από τους περιορισμούς του συστήματος είναι το γεγονός ότι μία πολιτική ίχνους ασχολείται πρωτίστως με λειτουργίες, όπως, για παράδειγμα, οι προσβάσεις σε αρχεία που μπορεί να εκτελέσει το υποκείμενο. Ως εκ τούτου, δεν λαμβάνει υπόψη του τις συγκεκριμένες τιμές των δεδομένων που το υποκείμενο διαβάζει ή γράφει, κάτι που θα οδηγούσε σε σημαντική αύξηση του υπολογιστικού κόστους. Τέλος, πέρα από την παρακολούθηση των προγραμμάτων με δικαιώματα (privileged programs), η προσέγγισή τους αυτή μπορεί να χρησιμοποιηθεί και για την παρακολούθηση τμημάτων και υπηρεσιών δικτύου σχετικών με την ασφάλεια, συμπεριλαμβανομένων και των DNS (Domain Name Services), συστημάτων αρχείων δικτύου και δρομολογητών (routers).

2.2. Κατηγοριοποίηση Τεχνικών

Βάσει των κατηγοριών και των προσεγγίσεων που αναλύθηκαν, οι προτεινόμενες τεχνικές μπορούν να ταξινομηθούν ως εξής:

Anomaly-based techniques
Dynamic

- ***Εντοπισμός Κακόβουλων Προγραμμάτων με τη Χρήση Αυτομάτων Πεπερασμένων Καταστάσεων (FSA) (3.8)***: Κάθε κόμβος στο αυτόματο αντιστοιχεί στην απόλυτη τιμή που έχει ο μετρητής προγράμματος τη στιγμή της κλήσης και οι μεταβάσεις από τον έναν κόμβο στον άλλον γίνονται μέσω των κλήσεων συστήματος. Πολλαπλές εκτελέσεις του αυτομάτου δίνουν τη φυσιολογική συμπεριφορά στον αλγόριθμο εκπαίδευσης. Μη έγκυρη μετάβαση από την τρέχουσα κατάσταση του αυτομάτου στην πιο πρόσφατη κλήση θεωρείται ανωμαλία. Χρησιμοποιείται μετρητής για τη σταδιακή συσσώρευση των ανωμαλιών ανάλογα με το βάρος του και, όταν ξεπεραστεί ένα συγκεκριμένο όριο, θεωρείται ότι υπάρχει εισβολή.
- ***Ανίχνευση Ανωμαλιών Βάσει Προδιαγραφών: Μια Καινούρια Προσέγγιση για τον Εντοπισμό Επιθέσεων Δικτύου (3.11)***[97]: Χρησιμοποιείται εκτεταμένο πεπερασμένο αυτόματο για τη μοντελοποίηση της συμπεριφοράς ενός δικτύου. Δημιουργούνται αντίγραφα των καταστάσεων και μεταβλητών του αυτομάτου για τις διαφορετικές μεταβάσεις, οι οποίες γίνονται μέσω της αποστολής/αποδοχής δικτυακών πακέτων. Μέσω χειροκίνητα ανεπτυγμένων προδιαγραφών και, στη συνέχεια, τεχνικών μηχανικής μάθησης, δημιουργείται το προφίλ κανονικής συμπεριφοράς του προγράμματος σε ομαλές συνθήκες. Σε περίπτωση σημαντικής απόκλισης μεταξύ των στατικών ιδιοτήτων δύο προφίλ, όπως η συχνότητα εμφάνισης συγκεκριμένων μεταβάσεων και η πιο συχνά εμφανιζόμενη τιμή μιας μεταβλητής, επισημαίνεται ανωμαλία.
- ***Ανίχνευση Ανωμαλιών Ωφέλιμων Φορτίων (PAYL) (3.12)***: Το εργαλείο υπολογίζει το αναμενόμενο ωφέλιμο φορτίο μίας εφαρμογής ή υπηρεσίας δικτύου και χρησιμοποιείται κυρίως έναντι σκουληκιών (worms). Κατά την εκπαίδευση, δημιουργείται μια κατανομή συχνοτήτων των bytes του φορτίου, η οποία οδηγεί στην ανάπτυξη κεντροειδούς μοντέλου για κάθε εύρος μήκους, πύλη και κατεύθυνση ροής, μέσω n-gram ανάλυσης. Το κάθε μοντέλο αποθηκεύει τη μέση συχνότητα των bytes και την τυπική απόκλιση της συχνότητας του κάθε byte. Μοντέλα με κοντινές μέσες συχνότητες κατανομής συγχωνεύονται. Κατά την ανίχνευση, το εργαλείο υπολογίζει τις κατανομές των εισερχόμενων φορτίων και χρησιμοποιεί τη Mahalonobis απόσταση για να τις συγκρίνει με αυτές των μοντέλων. Αν ξεπεραστεί κάποιο όριο στην απόσταση, το φορτίο θεωρείται κακόβουλο.
- ***Ανίχνευση Εισβολών Χρησιμοποιώντας Μοτίβα Παρακολούθησης Ίχνους Μεταβλητού Μήκους (3.16)***: Κατά την εκπαίδευση, γίνεται καταγραφή των γεγονότων ελέγχου, τα οποία δείχνουν την ακολουθία των δραστηριοτήτων σε οποιαδήποτε χρονική στιγμή, για τις διάφορες εκτελέσεις μιας διαδικασίας. Τα γεγονότα μεταφράζονται σε ακολουθίες χαρακτήρων και παράγουν τον πίνακα κανόνων. Γίνεται συγκέντρωση των διαδοχικών εμφανίσεων του ίδιου χαρακτήρα και αφαίρεση των διπλότυπων. Οι ακολουθίες αυτές παράγουν τον πίνακα μοτίβων. Κατά την ανίχνευση, παράγονται τα γεγονότα από τη διαδικασία σε πραγματικό χρόνο και σημειώνονται αυτά για τα οποία δεν υπάρχει αντίστοιχη εγγραφή στον πίνακα. Με βάση το πόσο καλή αντιστοίχιση μπορεί να γίνει, ανεξαρτήτως μήκους ακολουθίας, αποφασίζεται η συμπεριφορά του προγράμματος.
- ***Δικτυακή Ανάλυση των Μη-Ομαλών Γεγονότων (NATE) (3.17)***: Το NATE επικεντρώνεται στην ανίχνευση της ανώμαλης κυκλοφορίας στο δίκτυο. Οι ροές πληροφοριών από το συνδυασμό προέλευσης IP διεύθυνσης – αριθμού θύρας στον αντίστοιχο συνδυασμό προορισμού ορίζονται ως sessions. Μέσω της χρήσης cluster analysis, ομαδοποιούνται τα sessions στα TCP/IP

[97] Η συγκεκριμένη μέθοδος, όπως επισημάνθηκε και κατά την ανάλυσή της, είναι συνδυασμός τεχνικής βάσει ανωμαλιών και βάσει προδιαγραφών, οπότε και επιλέχθηκε να κατανεμηθεί στην κύρια κατηγορία των τεχνικών βάσει ανωμαλιών (anomaly-based techniques).

πρωτόκολλα και φαίνεται η διαφορετική κατανομή μεταξύ των ομαλών και μη ομαλών sessions. Η μέτρηση μόνο των επικεφαλίδων των πακέτων και όχι ολόκληρου του περιεχομένου τους, το βοηθάει στη διαχείριση κίνησης υψηλής ταχύτητας σε πραγματικό χρόνο.

- ***Τεχνικές Εξόρυξης Δεδομένων για την Ανίχνευση Εισβολών (3.19)***: Χρησιμοποιείται συρόμενο παράθυρο σε ομαλές και μη ομαλές ακολουθίες κλήσεων συστήματος για να παραχθούν μοναδικές ακολουθίες. Παράγονται οι λεγόμενοι ταξινομητές βάσης για το διαχωρισμό ακολουθιών κλήσεων συστήματος σε ομαλές και μη, οι οποίοι παίρνουν ως είσοδο δεδομένα ελέγχου. Κάθε χαρακτηριστικό της εκάστοτε καταχώρησης παίρνει τιμή 1 ή 0 ανάλογα με το αν υπήρχε αντιστοιχία με το μοντέλο ή όχι. Επίσης, εφαρμόζεται ένας αλγόριθμος εκμάθησης για την παραγωγή των ταξινομητών μετά-δεδομένων , που δέχονται σαν είσοδο τις εκτιμήσεις των ταξινομητών βάσης και εξάγουν μία τελική εκτίμηση. Για την κατασκευή των ταξινομητών βάσης χρησιμοποιούνται αλγόριθμοι εξόρυξης δεδομένων, όπως των κανόνων συσχέτισης και των συχνών επεισοδίων, οι οποίοι αναφέρονται συλλογικά ως σύνολο κανόνων. Ο πρώτος προσπαθεί να αποκομίσει σχέσεις πολλαπλών χαρακτηριστικών από έναν πίνακα βάσης δεδομένων, όπου κάθε γραμμή είναι ένα αρχείο ελέγχου και κάθε στήλη ένα χαρακτηριστικό του αρχείου. Ο δεύτερος είναι ένα σύνολο από γεγονότα που συμβαίνουν μαζί με τουλάχιστον μία προκαθορισμένη ελάχιστη συχνότητα σε κάποιο συγκεκριμένο χρονικό διάστημα. Το πρόγραμμα εκτελείται πολλές φορές και με διαφορετικές ρυθμίσεις κάθε φορά και υπολογίζεται το σύνολο κανόνων του για κάθε εκτέλεση, το οποίο συγχωνεύεται με το ήδη υπάρχον. Όταν δεν υπάρχουν πλέον καινούριοι κανόνες να προστεθούν, σταματάει η διαδικασία συλλογής δεδομένων και κανόνες με χαμηλή συχνότητα εμφάνισης εξαλείφονται από το σύνολο. Αντίστοιχα, υπολογίζεται το σύνολο κανόνων και στα δεδομένα υπό εξέταση και συγκρίνονται με το υφιστάμενο σύνολο κανόνων.
- ***Βελτιωμένη Μέθοδος Ανίχνευσης Εισβολών βάσει Σκιαγράφησης των Διαδικασιών (3.20)***: Κατά τη μέθοδο αυτή, κατασκευάζεται το προφίλ της διαδικασίας υπό εξέταση βάσει της ακολουθίας και της συχνότητας των κλήσεων συστήματος που καλούνται από αυτήν σε κανονικές συνθήκες χρήσης. Πιο συγκεκριμένα, το προφίλ αυτό αποτελείται από τρία επιμέρους προφίλ. Στο βασικό προφίλ καταγράφονται ο τύπος και ο αριθμός κατάταξης της κάθε κλήσης με βάση τη συχνότητα εμφάνισης. Το SUID προφίλ είναι μία ακολουθία από τις εμφανίσεις των κλήσεων συστήματος υπό κανονικές συνθήκες, με την κάθε κλήση να μεταφράζεται σε έναν αριθμό ο οποίος υποδεικνύεται στο βασικό προφίλ. Το προφίλ δαίμονα καταγράφει ακολουθίες κλήσεων συστήματος για τους δαίμονες του προγράμματος. Μόλις καθοριστεί το προφίλ της διαδικασίας μέσω των τριών παραπάνω, πρέπει να μετρηθεί η απόσταση μεταξύ αυτού και των δεδομένων του δείγματος κατά τη διάρκεια εκτέλεσης. Αν η τιμή της απόστασης ξεπεράσει κάποιο συγκεκριμένο όριο, τότε η εξεταζόμενη διαδικασία θεωρείται πιθανώς κακόβουλη. Τέλος, χρησιμοποιείται ο *DP matching* αλγόριθμος αντιστοίχισης μοτίβων για την μέτρηση των αποστάσεων και την εξάλειψη των ψευδών συναγερμών.

Static

- ***Εξαγωγή των API Κλήσεων των Windows (3.2)***: Σκοπός της τεχνικής αυτής είναι η αυτοματοποιημένη εξαγωγή και ανάλυση της συμπεριφοράς των API κλήσεων από κακόβουλα λογισμικά (με τη μορφή δυαδικών εκτελέσιμων αρχείων), σε απομονωμένο περιβάλλον, τα οποία χρησιμοποιούν τεχνικές απόκρυψης του κώδικά τους (obfuscation techniques). Αρχικά, γίνεται εξαγωγή του κρυπτογραφημένου κώδικα του κάθε αρχείου μέσω του ανιχνευτή PEiD, εφόσον υπάρχει. Έπειτα, χρησιμοποιείται το εργαλείο IDA Pro για την ανάλυση του δυαδικού κώδικα του κάθε εκτελέσιμου, που οδηγεί στην παραγωγή ενός .idb αρχείου για το κάθε ένα από αυτά. Με βάση τα προηγούμενα αποθηκεύεται για το κάθε εκτελέσιμο πληροφορίες σχετικές με το δυαδικό του κώδικα, μεταξύ των οποίων και τις ακολουθίες των API κλήσεών του. Στη συνέχεια, οι ακολουθίες αυτές συγκρίνονται και αντιστοιχίζονται με τις αντίστοιχες ακολουθίες API κλήσεων του MSDN (Microsoft Developer Network) και αποφασίζεται η κακόβουλη ή μη συμπεριφορά του κώδικα. Τέλος, βάσει συγκεκριμένων χαρακτηριστικών των API κλήσεων (όπως η συχνότητά εμφάνισης της καθεμίας, παρατηρούμενες ακολουθίες και οι ενέργειες που εκτελούνται αμέσως πριν ή αμέσως μετά μία κλήση), γίνεται περαιτέρω ανάλυση της κακόβουλης συμπεριφοράς και

κατηγοριοποίησή της στις ακόλουθες έξι κύριες κατηγορίες: αναζήτηση αρχείων προς μόλυνση, αντιγραφή/διαγραφή αρχείων, ανάκτηση πληροφοριών του αρχείου, μετακίνηση αρχείων, ανάγνωση/εγγραφή αρχείων και αλλαγή στα χαρακτηριστικά του αρχείου.

Specification-based techniques
Dynamic

- ***Ασφαλής Εκτέλεση Προγράμματος μέσω Δυναμικής Παρακολούθησης της Ροής Πληροφοριών (3.13)***: Η μέθοδος βασίζεται στην ιδέα ότι τα δεδομένα χωρίζονται σε αυθεντικά και ψευδή, ανάλογα με το αν το δεδομένο είναι υπό τον έλεγχο του προγράμματος ή όχι. Καθορίζεται μία πολιτική ασφαλείας από το λειτουργικό σύστημα, η οποία αναθέτει το ψηφίο (bit) 0 ή 1 αντίστοιχα, ανάλογα με το αν τα δεδομένα είναι αυθεντικά ή ψευδή. Η πολιτική ασφαλείας που ακολουθείται, θεωρεί όλα τα I/O κανάλια εν δυνάμει ψευδή, ενώ χρησιμοποιεί τις εξαρτήσεις *Copy*, *Computation*, *Load-address* και *Store-address*. Αν βρεθεί ότι ο έλεγχος του προγράμματος εξαρτάται από ψευδή δεδομένα, ο επεξεργαστής επιστρέφει εξαίρεση και το λειτουργικό σύστημα τερματίζει τη διαδικασία. Για παράδειγμα, μία συνάρτηση χρησιμοποιεί την *fgets* για να διαβάσει από ένα αρχείο εισόδου, το οποίο θεωρείται εξ' ορισμού ψευδές. Το string εισόδου αντιγράφεται και πάει στον buffer. Οι επεξεργασμένες τιμές θεωρούνται ψευδές μέσω της *Copy dependency*. Τέλος, όταν η συνάρτηση επιστρέφει, χρησιμοποιεί την *ret* οδηγία (instruction), η οποία σχετίζεται με τους καταχωρητές, και, αφού ο pointer είναι ψευδής, ο επεξεργαστής επιστρέφει εξαίρεση.
- ***Εντοπισμός Επιθέσεων στις Εφαρμογές Χρησιμοποιώντας Δυναμική Ανάλυση Ροής Πληροφοριών (3.14)*:** Παρουσιάζεται το εργαλείο DIFA, το οποίο σχεδιάστηκε ειδικά για Java εφαρμογές. Χρησιμοποιεί τη γλώσσα μηχανής της JVM (bytecodes). Όταν η JVM φορτώνει το .class αρχείο, παίρνει μία ροή από bytecodes για κάθε μέθοδο στην κλάση. Οι ροές αυτές αποτελούν ακολουθίες οδηγιών προς τη JVM. Εκτελούνται όταν καλεστεί η μέθοδός τους κατά τη διάρκεια εκτέλεσης του προγράμματος και μπορούν να συγκριθούν με γνωστές πολιτικές ροών. Ορίζονται τρεις οντότητες για τις πολιτικές αυτές: Sources, SensitiveSources και sink methods. Ένα παράδειγμα πιθανής διαρροής πληροφοριών είναι το ακόλουθο: χρησιμοποιείται ένας scheduler συναντήσεων στο Πεντάγωνο που παραγγέλνει αυτόματα φαγητό από κάποια ώρα και μετά, χρησιμοποιώντας ένα fax server και χρεώνεται στον πιο υψηλόβαθμο υπάλληλο . Τα Sources αναφέρονται σε όλα τα αντικείμενα υπό παρακολούθηση. Τα SensitiveSources αναφέρονται στα ευαίσθητα αντικείμενα της πρώτης κατηγορίας, δηλαδή στα χαρακτηριστικά που καθορίζουν τους υπαλλήλους υψηλού κινδύνου, όπως το όνομα, το ID και ο αριθμός λογαριασμού τους. Τα Sinks αναφέρονται στις μεθόδους που σχετίζονται με την αποστολή πληροφοριών μέσω fax. Ο παραλήπτης του fax μπορεί να αναγνωρίζει κάποιο μοτίβο στους λογαριασμούς εξόδων και άρα να φτάνει στο συμπέρασμα ότι στη συνάντηση λαμβάνουν μέρος υψηλόβαθμοι υπάλληλοι και άρα να προγραμματίζονται κάποιες ειδικές αποστολές.
- ***Πρόληψη Εισβολών Μέσω της Παρακολούθησης της Συμπεριφοράς των Διεργασιών (3.18)***: Η μέθοδος αυτή υλοποιήθηκε σε λειτουργικό Linux και αφορά συστήματα υπολογιστών που είναι διασυνδεδεμένα μέσω ενός δικτύου. Το δίκτυο είναι συνδεδεμένο στο Διαδίκτυο, απ' όπου και προέρχονται οι πιθανές απομακρυσμένες επιθέσεις. Η μέθοδος επικεντρώνεται σε τρία ζητήματα: 1) στην ανίχνευση επιθέσεων πριν την πρόκληση ζημιάς, 2) στον περιορισμό πιθανής ζημιάς και 3) στον εντοπισμό της προέλευσης της επίθεσης. Για κάθε πρόγραμμα αναπτύσσεται μία προδιαγραφή από το διαχειριστή ασφαλείας του συστήματος, ο οποίος γνωρίζει την επιθυμητή συμπεριφορά του προγράμματος και πιθανές αδυναμίες αυτού. Η κάθε προδιαγραφή χρησιμοποιεί την υψηλού επιπέδου γλώσσα προδιαγραφών ASL. Μέσω των προδιαγραφών αυτών καθορίζονται οι ομαλές και μη ομαλές συμπεριφορές των διαδικασιών που εκτελούνται από τους υπολογιστές του συστήματος, μέσω των ακολουθιών των κλήσεων συστήματος. Κατά την εκτέλεση μιας διαδικασίας από το πρόγραμμα, χρησιμοποιείται μία αρχικοποίηση της αντίστοιχης προδιαγραφής για την παρακολούθηση των κλήσεων συστήματος που καλούνται κατά την διαδικασία. Σε περίπτωση σημαντικής απόκλισης από την επιθυμητή συμπεριφορά, η διαδικασία τερματίζεται ή συνεχίζει να εκτελείται σε απομονωμένο περιβάλλον, με σκοπό την περαιτέρω ανάλυσή της στη συνέχεια. Για παράδειγμα, η προδιαγραφή για το δαίμονα *finger*, ο οποίος είναι υπεύθυνος για την παροχή πληροφοριών σχετικά με το χρήστη, όπως το πλήρες

ονοματεπώνυμο και η e-mail διεύθυνσή του, δεν του επιτρέπει την εκτέλεση αρχείων, την εγγραφή σε αρχείο, τη δημιουργία σύνδεσης στο δίκτυο, ενώ του δίνει δικαιώματα ανάγνωσης μόνο σε συγκεκριμένα αρχεία. Σε περίπτωση εντοπισμού συμπεριφοράς που αντιβαίνει σε αυτή την προδιαγραφή, η αντίστοιχη κλήση συστήματος δεν εκτελείται και επιστρέφεται μήνυμα λάθους ή τερματίζεται η διαδικασία.

- ***Παρακολούθηση Κρίσιμων για την Ασφάλεια Προγραμμάτων σε Κατανεμημένα Συστήματα (3.21)***: Η τεχνική αυτή επικεντρώνεται στην ανίχνευση εκμετάλλευσης αδυναμιών κρίσιμων για την ασφάλεια προγραμμάτων. Χρησιμοποιεί προδιαγραφές ασφαλείας που περιγράφουν την επιθυμητή συμπεριφορά των προγραμμάτων σχετικά με την ασφάλεια. Κάνει χρήση χρονικά διατεταγμένων ακολουθιών από γεγονότα εκτέλεσης, που ονομάζονται ίχνη και λαμβάνονται σε πραγματικό χρόνο. Η κάθε προδιαγραφή περιγράφει έγκυρες ακολουθίες λειτουργιών της εκτέλεσης ενός ή περισσοτέρων προγραμμάτων, που συλλογικά ονομάζονται παρακολουθούμενο υποκείμενο. Καθώς η προδιαγραφή αποφασίζει κατά πόσο είναι έγκυρο το ίχνος εκτέλεσης του υποκειμένου, ονομάζεται πολιτική ίχνους. Η παρακολούθηση του υποκειμένου έγκειται στην συντακτική ανάλυση της ακολουθίας των λειτουργιών που εκτελούνται από αυτό. Μία ακολουθία λειτουργιών του υποκειμένου που δε συμμορφώνεται με την προδιαγραφή θεωρείται παραβίαση ασφαλείας. Οι έγκυρες ακολουθίες λειτουργιών καθορίζονται βάσει γραμματικών, των οποίων τα αλφάβητα αποτελούνται από λειτουργίες συστήματος.

Static

- ***Στατικός Εντοπισμός Κακόβουλου Λογισμικού σε Εκτελέσιμα Προγράμματα (3.4)***: Το εργαλείο που προτείνεται δέχεται σαν είσοδο ένα εκτελέσιμο αρχείο δυαδικού κώδικα και μία πολιτική ασφαλείας και αποφασίζει για την ύπαρξη ή μη κακόβουλου κώδικα. Η πολιτική ασφαλείας είναι ένα σύνολο κανόνων που χαρακτηρίζει ως αποδεκτή ή μη την εκτέλεση ενός προγράμματος. Αρχικά το εκτελέσιμο μεταφράζεται σε γλώσσα υψηλού επιπέδου, μέσω του εργαλείου IDA32 Pro, και, έπειτα, αναλύεται για να παραχθεί το συντακτικό του δένδρο. Από αυτό παράγονται, στη συνέχεια, τα κατευθυνόμενα γραφήματα ροής ελέγχου και ροής δεδομένων. Κάθε κόμβος του πρώτου αντιστοιχεί σε μια δήλωση (statement) του προγράμματος, ενώ του δεύτερου αντιστοιχεί σε μία λειτουργία του προγράμματος. Από το συνδυασμό αυτών των δύο, δημιουργείται το API γράφημα, το οποίο περιλαμβάνει μόνο τις κλήσεις προς το API και αγνοεί τις υπόλοιπες εντολές. Στη συνέχεια, το εργαλείο επιτρέπει στον χρήστη να αποφασίσει ποιες από αυτές τις κλήσεις θεωρούνται κρισιμότερες σε θέματα ασφαλείας, αναλόγως την εφαρμογή, και έτσι παράγεται το κρίσιμο-API γράφημα, το οποίο ελέγχει εάν τηρούνται οι πολιτικές ασφαλείας. Οι τελευταίες εκπροσωπούνται από αυτόματα και εισάγονται χειροκίνητα από το χρήστη στο εργαλείο. Οι μεταβάσεις σε αυτά τα αυτόματα συμβαίνουν όταν εκτελείται κάποια ενέργεια από το σύστημα, ενώ η είσοδος στην κακή κατάσταση του αυτομάτου, θεωρείται παραβίαση της πολιτικής ασφαλείας. Για παράδειγμα, θεωρούμε ένα αυτόματο, όπου η πολιτική ασφαλείας που ορίζει δεν επιτρέπει στα προγράμματα να διαβάζουν από ένα αρχείο και, στη συνέχεια, να στέλνουν δεδομένα στο δίκτυο. Όποτε, αν ένα πρόγραμμα προσπαθεί να εκτελέσει τις εντολές *OpenFile*, *ReadFile* και *Send*, το αυτόματο θα μεταβεί στην κακή κατάσταση, που υποδηλώνει ότι παραβιάστηκε η πολιτική ασφαλείας του.

Hybrid

- ***Ανίχνευση Εισαγόμενου, Δυναμικά Παραγόμενου και Αποκρυμμένου Κακόβουλου Κώδικα (3.6)***: Η προσέγγιση DOME χρησιμοποιείται για την ανίχνευση κακόβουλου κώδικα, ο οποίος αλληλοεπιδρά άμεσα με το λειτουργικό σύστημα μέσω Win32 API κλήσεων, σε εκτελέσιμα προγράμματα. Ο κακόβουλος κώδικας μπορεί να εισαχθεί στο χώρο διευθύνσεων ή να δημιουργηθεί από μια διεργασία κατά την εκτέλεση του προγράμματος, ενώ μπορεί να είναι και εξαρχής κρυμμένος στον κώδικα πίσω από πολύπλοκους υπολογισμούς. Το DOME χρησιμοποιεί στατική ανάλυση για την αναγνώριση των οδηγιών που κάνουν κλήσεις προς το API σε ομαλές συνθήκες και αποθηκεύει τα ονόματα, τις εικονικές και τις διευθύνσεις επιστροφής τους (return addresses), οι οποίες θα βρίσκονται στην κορυφή της στοίβας κλήσεων όταν γίνονται οι κλήσεις. Η στοίβα κλήσεων κρατά πληροφορίες σχετικά με τις ενεργές διαδικασίες του προγράμματος. Στη

δεύτερη φάση, παρακολουθεί τις API κλήσεις που γίνονται από το εκτελέσιμο υπό εξέταση, σε πραγματικό χρόνο. Αναγνωρίζει τις οδηγίες που είναι υπεύθυνες για την κλήση και τη διεύθυνσή της μέσα στο εκτελέσιμο. Ύστερα συγκρίνει το όνομα και τη διεύθυνση επιστροφής της με αυτά που έχουν αναγνωριστεί κατά τη στατική ανάλυση. Οι περιπτώσεις αναντιστοιχίας επισημαίνονται ως απόπειρα εισβολής, στην οποία περίπτωση, το σύστημα μπλοκάρει την κλήση. Η ανίχνευση λαμβάνει μέρος πριν ο κακόβουλος κώδικας μπορέσει να εισέλθει στα αρχεία του λειτουργικού συστήματος και να προκαλέσει ζημιά. Σε περίπτωση ενημέρωσης του εκτελέσιμου, το βήμα της στατικής ανάλυσης πρέπει να γίνει ξανά από την αρχή.

- ***Ανίχνευση Εισβολών Μέσω Στατικής Ανάλυσης (3.10)***: Η προσέγγιση προϋποθέτει ότι το πρόγραμμα υπό εξέταση δεν έχει αρχικά γραφτεί για κακόβουλο σκοπό και ότι τα ίχνη των κλήσεων συστήματος του προγράμματος κατά την εκτέλεσή του συνάδουν με τον πηγαίο του κώδικα. Επικεντρώνεται στην εξάλειψη των ψευδών συναγερμών και υποθέτει ότι δεν υπάρχουν σφάλματα μνήμης στον πηγαίο κώδικα (π.χ. NULL-pointer references). Αρχικά, μέσω στατικής ανάλυσης του πηγαίου κώδικα της εφαρμογής, υπολογίζεται η αναμενόμενη συμπεριφορά της. Στη συνέχεια, παρακολουθείται το πρόγραμμα κατά την εκτέλεσή του και συγκρίνονται οι ακολουθίες κλήσεων συστήματος σε σχέση με αυτές που βρέθηκαν κατά την προηγούμενη φάση. Η κεντρική ιδέα είναι ότι θεωρούμε ως S το σύνολο των κλήσεων συστήματος που μπορεί να κάνει η εφαρμογή και S* το σύνολο των αποδεκτών ακολουθιών από κλήσεις, που έχουν παραχθεί με τη χρήση κανονικής γλώσσας. Σε περίπτωση ύπαρξης κλήσης εκτός αυτού του συνόλου κατά την εκτέλεση της εφαρμογής, εμποδίζεται η εκτέλεση της κλήσης, σταματάει η εφαρμογή και σημαίνεται συναγερμός. Πάνω σε αυτήν την ιδέα αναπτύσσεται το λεγόμενο callgraph model (μοντέλο γραφήματος-κλήσεων), όπου αναλύεται ο πηγαίος κώδικας της εφαρμογής και εξάγεται το διάγραμμα ροής ελέγχου, το οποίο αντιπροσωπεύει τις πιθανές ακολουθίες των κλήσεων συστήματός του, και αναπαρίσταται μέσω μη-ντετερμινιστικού πεπερασμένου αυτομάτου. Βελτίωση του παραπάνω αποτελεί το abstract stack model (αφηρημένο μοντέλο στοίβας), το οποίο κάνει χρήση μη ντετερμινιστικού αυτομάτου στοίβας και το οποίο εξαλείφει το πρόβλημα ύπαρξης ανέφικτων διαδρομών, όπως την περίπτωση κλήσης που επιστρέφει σε διαφορετική διεύθυνση από τη διεύθυνση επιστροφής της, που υπήρχε στο προηγούμενο μοντέλο. Τέλος, περιγράφεται το λεγόμενο digraph model (διγραφικό μοντέλο), το οποίο κατασκευάζει λίστα με όλες τις πιθανές ακολουθίες 2 διαδοχικών κλήσεων συστήματος, ξεκινώντας από οποιοδήποτε σημείο της εφαρμογής.

Signature-based techniques
Dynamic

- ***Χρήση της Μηχανικής Μάθησης και των API των Windows (3.1)***: Η συγκεκριμένη μέθοδος χρησιμοποιεί μία συλλογή από κακόβουλα και μη προγράμματα και καταγράφει τις ακολουθίες των Windows API κλήσεων της κάθε διεργασίας τους μέσω ενός ανιχνευτή API κλήσεων. Στη συνέχεια τις μετατρέπει σε μία ακολουθία ακεραίων, μέσω της API index βάσης δεδομένων, η οποία αντιστοιχίζει κάθε API κλήση σε ένα μοναδικό ακέραιο. Έπειτα, η παραγόμενη ακολουθία αποθηκεύεται στη βάση υπογραφών. Τέλος, χρησιμοποιείται ο προτεινόμενος αλγόριθμος μηχανικής μάθησης Association mining, ο οποίος λαμβάνει ως είσοδο την βάση υπογραφών και προσπαθεί να αναδείξει τις ενδιαφέρουσες σχέσεις που ενυπάρχουν σε αυτή, ως κανόνες κατηγοριοποίησης. Κατά την επόμενη φάση (Online φάση - αξιολόγησης), το δείγμα εκτελέσιμων μετατρέπεται στις αντίστοιχες υπογραφές (μέσω της ίδιας διαδικασίας) και αξιολογούνται παράθυρα μήκους 4 (4-grams – 3ης τάξης Μαρκοβιανή αλυσίδα). Τέλος, υπολογίζεται ο μέσος συντελεστής εμπιστοσύνης όλων των 4-grams παραθύρων και, με βάση αυτό, ταξινομείται η διεργασία σε ομαλή ή κακόβουλη αντίστοιχα.
- ***Συμπεριφορική Προσέγγιση για τον Εντοπισμό Σκουληκιών Υπολογιστή (Worms) (3.9)***: Η προσέγγιση αυτή στοχεύει στον εντοπισμό σκουληκιών (worms), μέσω της χρήσης συμπεριφορικών υπογραφών. Ενώ συνήθως οι υπογραφές ορίζονται ως κανονικές εκφράσεις στον κακόβουλο κώδικα, εντούτοις, στην προκειμένη περίπτωση, ορίζονται ως οι έμφυτες συμπεριφορές των σκουληκιών που εξαπλώνονται (propagating worms). Οι τρεις παρακάτω υπογραφές ονομάζονται υπογραφές βάσης και το βασικό τους χαρακτηριστικό είναι η

παρακολούθηση των ροών δεδομένων που εισέρχονται ή εξέρχονται από ένα κόμβο (κεντρικό υπολογιστή, router κτλ.). Η πρώτη από αυτές τις υπογραφές (*Server Changes to Client*) είναι όταν ένας διακομιστής (server) αλλάζει σε πελάτη (client). Από τη στιγμή που ένα σκουλήκι μολύνει ένα server, πρέπει να ενεργήσει ως πελάτης για να εξαπλωθεί σε κάποιον άλλον server με σκοπό να τον μολύνει και εκείνον μέσω της ίδιας αδυναμίας. Η δεύτερη (*alpha-in/alpha-out*) υποδεικνύει ότι μία συγκεκριμένη συμπεριφορά που συνέβη μεταξύ δύο κεντρικών υπολογιστών α και β, στη συνέχεια συνέβη μεταξύ των β και γ κεντρικών υπολογιστών. Δείχνει δηλαδή ότι τα σκουλήκια μεταφέρουν παρόμοια πακέτα μεταξύ μολυσμένων υπολογιστών. Η τρίτη (*fanout*) αποτελεί ειδική κατηγορία των επαγωγικών υπογραφών και συγκεκριμένα της σχέσης απογόνου (descendant relation), όπου το κάθε σκουλήκι πρέπει να μολύνει περισσότερους από έναν hosts κάθε φορά, αλλιώς δεν επιτυγχάνεται η εκθετική εξάπλωσή του. Πιο συγκεκριμένα, η fanout λαμβάνει υπόψιν της μόνο απογόνους βάθους ίσο με ένα και σε περίπτωση που βρεθεί host με περισσότερους απογόνους, τον μπλοκάρει.

Static

- ***Στατικός Αναλυτής Κακόβουλων Εκτελέσιμων (Static Analyzer of Vicious Executables) (SAVE) (3.5)*:** Η συγκεκριμένη προσέγγιση επικεντρώνεται στην ανίχνευση κακόβολου λογισμικού που έχει περάσει από τεχνικές πολυμορφισμού ή μεταμορφισμού, οι οποίες έχουν σκοπό την απόκρυψη της πραγματικής φύσης του λογισμικού. Η μέθοδος βασίζεται στην υπόθεση ότι όλες οι πιθανές εκδόσεις ενός malware μοιράζονται μία κοινή βασική υπογραφή. Θεωρείται, δηλαδή, ότι η αρχική έκδοση ενός συγκεκριμένου κακόβουλου λογισμικού *M* περιλαμβάνει ένα κακόβουλο σύνολο *S* από κλήσεις συστήματος. Τότε, οποιαδήποτε παραλλαγή του *M*, ύστερα από χρήση τεχνικών πολυμορφισμού ή μεταμορφισμού, θα περιλαμβάνει ένα αντίστοιχο κακόβουλο σύνολο από κλήσεις συστήματος *S'* και ζητείται να βρεθεί η ομοιότητα μεταξύ των δύο συνόλων. Η μέθοδος δέχεται ως είσοδο δυαδικά εκτελέσιμα αρχεία, τα οποία αναλύει στις ακολουθίες των API κλήσεών τους. Κάθε API κλήση αντιστοιχίζεται σε ένα μοναδικό ακέραιο, οπότε η ακολουθία των API κλήσεων παίρνει τελικά τη μορφή ακολουθίας ακεραίων. Στη συνέχεια, χρησιμοποιούνται τρεις γνωστές συναρτήσεις για τη μέτρηση της ομοιότητας της υπό εξέταση ακολουθίας ακεραίων με κάθε μία ακολουθία γνωστών κακόβουλων λογισμικών και υπολογίζεται ο μέσος όρος αυτών. Αν αυτός ο αριθμός ομοιότητας ξεπεράσει το 90% για κάποια από τις υπογραφές της βάσης, τότε το υπό εξέταση εκτελέσιμο θεωρείται παραλλαγή του αντίστοιχου κακόβουλου λογισμικού.
- ***Στατικός Αναλυτής Εκτελέσιμων Αρχείων (Static Analyzer for Executables) (SAFE) (3.15)***: Το εργαλείο αυτό επικεντρώνεται κυρίως στην ανίχνευση ιών σε εκτελέσιμα αρχεία, ειδικότερα μετά από χρήση τεχνικών απόκρυψης του κώδικα (obfuscation techniques), όπως η εισαγωγή εντολών που δεν χρησιμοποιούνται (dead-code insertion) και η αντικατάσταση οδηγιών με άλλες ισοδύναμες. Γίνεται η υπόθεση ότι, ακόμα και μετά τη χρήση των τεχνικών αυτών, δε γίνεται να αποκρυφθεί τελείως η κακόβουλη συμπεριφορά του ιού. Αρχικά, γίνεται μία αφηρημένη αναπαράσταση του κακόβουλου κώδικα (στην προκειμένη περίπτωση του ιού για τον οποίο προσπαθούν να ανιχνευθούν πιθανές παραλλαγές), μέσω της κατασκευής του αντίστοιχου αυτομάτου. Στη συνέχεια, το SAFE δημιουργεί ένα γράφημα ροής ελέγχου για την κάθε διαδικασία του εκτελέσιμου υπό εξέταση, στο οποίο αποτυπώνονται όλα τα δυνατά μονοπάτια που μπορεί να ακολουθήσει το πρόγραμμα κατά την εκτέλεσή του. Έπειτα, χρησιμοποιώντας ένα σύνολο από γενικευμένα μοτίβα, χαρακτηρίζει το κάθε κόμβο του γραφήματος με το σύνολο των μοτίβων που αντιστοιχίζονται σε αυτόν . Έτσι δημιουργείται το γράφημα ροής ελέγχου με σχολιασμούς, ένα για την κάθε διαδικασία του εκτελέσιμου. Το γράφημα αυτό δείχνει πού βρίσκεται το κάθε γενικευμένο μοτίβο στο εκτελέσιμο. Τέλος, υπολογίζεται η τομή μεταξύ της γλώσσας του κάθε γραφήματος με σχολιασμούς για την εκάστοτε διαδικασία του εκτελέσιμου με την αντίστοιχη γλώσσα των γενικευμένων μοτίβων (τα οποία χρησιμοποιούνται ως αλφάβητο για το αυτόματο). Αν η τομή αυτή βρεθεί να είναι μη-κενή, τότε το εκτελέσιμο θεωρείται ότι μεταφέρει παραλλαγή του ιού.

Hybrid

- ***Κατηγοριοποίηση Κακόβουλου Λογισμικού Μέσω Δομημένης Ροής Ελέγχου (3.3)***: Η μέθοδος αυτή επικεντρώνεται στην ανίχνευση παραλλαγών κακόβουλων λογισμικών. Καθώς στα κακόβουλα αυτά λογισμικά μπορεί να έχει γίνει χρήση τεχνικών μεταμόρφωσης του κώδικά τους (packing), ώστε να είναι πιο δύσκολος ο εντοπισμός τους, το σύστημα χρησιμοποιεί ανάλυση εντροπίας για να τα αναγνωρίσει, καθώς συνεχόμενα κομμάτια κώδικα με σχετικά ψηλό βαθμό εντροπίας συνήθως υποδεικνύουν την ύπαρξη packed κώδικα. Σε αυτά χρησιμοποιείται εξομοιωτής για την προσομοίωση της εκτέλεσης του κρυμμένου κώδικα σε ένα ασφαλές και απομονωμένο περιβάλλον. Η διαδικασία της προσομοίωσης επαναλαμβάνεται μέχρις ότου να εξαχθεί όλος ο packed κώδικας. Στη συνέχεια, το σύστημα προχωράει στην στατική ανάλυση του πλέον “unpacked” κώδικα. Πιο συγκεκριμένα, αναγνωρίζονται οι διαδικασίες (procedures) του κώδικα και παράγεται το γράφημα ροής ελέγχου για κάθε μία από αυτές. Μέσω της τεχνικής δόμησης (structuring), παράγονται οι αντίστοιχες δομημένες ροές ελέγχου υψηλού επιπέδου από τα προαναφερθέντα γραφήματα και αναπαρίστανται με τη μορφή συμβολοσειρών (υπογραφές), μία για κάθε διαδικασία μαζί με το αντίστοιχο βάρος που της αποδίδεται. Η κάθε μία από αυτές τις υπογραφές συγκρίνεται με τις υπογραφές γνωστών κακόβουλων λογισμικών που είναι αποθηκευμένες σε μία βάση δεδομένων, μέσω της edit distance μετρικής. Όταν ο λόγος ομοιότητας μεταξύ τους γίνει μεγαλύτερος ή ίσος του 0,90, τότε θεωρούμε ότι υπάρχει αντιστοίχιση των υπογραφών. Η σύγκριση γίνεται για όλες τις υπογραφές του κώδικα υπό εξέταση. Στη συνέχεια, για το κάθε malware της βάσης για το οποίο βρέθηκαν υπογραφές που να αντιστοιχίζονται στη δική του, συγκεντρώνονται οι λόγοι ομοιότητας των υπογραφών αυτών αναλογικά με τα βάρη τους. Αν η ομοιότητα μεταξύ του εξεταζόμενου κώδικα και κάποιου από τα malwares της βάσης δεδομένων ξεπεράσει το 0,60, τότε θεωρείται ότι είναι παραλλαγή αυτού του malware και οι υπογραφές του προστίθενται στη βάση δεδομένων.
- ***Εργαλείο για την Ανάλυση και Ανίχνευση Κινητού Κακόβουλου Κώδικα (3.7)***: Η τεχνική αυτή επικεντρώνεται στην ανίχνευση αυτό-κρυπτογραφημένων και πολυμορφικών ιών και σκουληκιών υπολογιστή. Αρχικά, χρησιμοποιείται προσομοιωτής για την ανάλυση της συμπεριφοράς του ιού κατά τη διάρκεια εκτέλεσής του. Στην περίπτωση του αυτό-κρυπτογραφημένου ιού, ο ιός θα αποκρυπτογραφήσει τον κώδικά του κατά την προσομοίωση, ο οποίος θα χρησιμοποιηθεί, ύστερα, κατά την στατική ανάλυση. Χρησιμοποιείται επίσης εξομοιωτής εκτέλεσης του λειτουργικού συστήματος, με σκοπό την αποθήκευση αναγκαίων πληροφοριών, όπως οι καταχωρητές και οι περιβαλλοντικές μεταβλητές. Στη συνέχεια, για να καθοριστεί η κακόβουλη συμπεριφορά του ιού, χρησιμοποιείται ανάλυση της ροής ελέγχου του κώδικα για να καθοριστούν οι API κλήσεις του. Έπειτα, το εργαλείο ελέγχει κατά πόσο αναγνωρίζονται συγκεκριμένοι τύποι συμπεριφοράς μέσα στον κώδικα, βάσει συγκεκριμένων πολιτικών. Οι πολιτικές αυτές περιγράφουν διάφορα σενάρια στο επίπεδο των API κλήσεων και ορίζονται με τη μορφή αυτομάτων, όπου οι μεταβάσεις από τη μία κατάσταση στην επόμενη γίνεται μέσω των API κλήσεων υπό συγκεκριμένες συνθήκες. Όταν το πρόγραμμα προς εξέταση φτάσει σε μία τελική κατάσταση «άρνησης» (“denial”), τότε θεωρείται κακόβουλο. Παραδείγματα αυτών των πολιτικών είναι ο έλεγχος για το κατά πόσο το πρόγραμμα δημιουργεί αντίγραφο του εαυτού του κατά την εκτέλεση, ο έλεγχος για πιθανές τροποποιήσεις σε αρχεία και φακέλους στα οποία δεν υπάρχουν δικαιώματα εγγραφής κτλ.

3. Συμπεράσματα

Στην παρούσα εργασία, παρουσιάστηκαν διάφορες τεχνικές, από όλες τις κατηγορίες και προσεγγίσεις, σχετικά με τον εντοπισμό κακόβουλου λογισμικού. Κάθε μία από αυτές επικεντρώνεται σε συγκεκριμένες πτυχές του εκάστοτε κακόβουλου λογισμικού, τις οποίες και χρησιμοποιεί προς όφελός της για τον εντοπισμό του. Πιο συγκεκριμένα, οι τεχνικές εντοπισμού που παρουσιάστηκαν ανήκουν στις κατηγορίες της εύρεσης βάσει ανωμαλιών (anomaly-based) και βάσει υπογραφών (signature-based). Οι πρώτες ορίζουν τι καθιστά την κανονική συμπεριφορά (νόρμα) ενός προγράμματος και εκπαιδεύουν το σύστημα να εντοπίζει τις αποκλίσεις από αυτήν, ενώ οι δεύτερες, με βάση ένα σύνολο από υπογραφές (συνήθως ακολουθίες από bytes) γνωστών κακόβουλων λογισμικών, συγκρίνουν αυτές με των αντίστοιχων προγραμμάτων προς εξέταση. Υποκατηγορία της πρώτης είναι και ο εντοπισμός βάσει προδιαγραφών (specification-based), η οποία εξετάζει πολιτικές που συνήθως καθορίζονται από τον χρήστη και βάσει του εάν ακολουθούνται ή όχι, κρίνονται κακόβουλα ή μη.

Κάθε μία από τις παραπάνω κατηγορίες μπορεί να εφαρμοστεί είτε μέσω στατικής προσέγγισης, είτε μέσω δυναμικής, είτε μέσω συνδυασμού των δύο (υβριδική προσέγγιση). Ο διαχωρισμός μεταξύ αυτών αφορά στον τρόπο με τον οποίο διακρίνονται τα ομαλά από τα κακόβουλα λογισμικά. Η πρώτη εντοπίζει τις δομικές ιδιότητες του προγράμματος, αναλύοντας τον πηγαίο κώδικά του, πριν αυτό εκτελεστεί ενώ η δεύτερη παρακολουθεί τη συμπεριφορά του προγράμματος σε πραγματικό χρόνο.

Η κάθε μία από αυτές τις τεχνικές μπορεί να θεωρηθεί ουσιώδης μόνο στην περιοχή δράσης της και όχι έναντι κάθε τύπου κακόβουλου λογισμικού, καθώς καμία από αυτές δεν είναι από μόνη της αρκετή. Αντιθέτως, όπως πολλοί από τους συγγραφείς σημειώνουν, οι τεχνικές που προτείνουν είναι πιο αποτελεσματικές όταν δουλεύουν συμπληρωματικά μαζί με άλλες τεχνικές που έχουν διαφορετικό πεδίο δράσης.

Τέλος, πιθανές συγκρίσεις μεταξύ των διαφόρων τεχνικών ιδίου πεδίου δεν είναι πάντα αξιόπιστες, καθώς δεν είναι πάντα ξεκάθαρο το κατά πόσο οι παράμετροι και οι ιδιότητες του δείγματος που χρησιμοποιείται για τις συγκρίσεις είναι βελτιστοποιημένες για την εκάστοτε τεχνική.

4. Βιβλιογραφία

[1] R. M. Chandrasekar Ravi, «Malware Detection using Windows API Sequence and Machine Learning,» *International Journal of Computer Applications (0975-8887),* τόμ. 43, αρ. 17, pp. 12-16, April 2012.

[2] S. V. P. W. Mamoun Alazab, «Towards understanding Malware Behaviour by the Extraction of API Calls,» *2010 Second Cybercrime and Trustworthy Computing Workshop,* July 2010.

[3] Y. X. Silvio Cesare, «Classification of Malware Using Structured Control Flow,» σε *Proc. 8th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010)*, Brisbane, Australia, 2010.

[4] M. D. J. D. M. M. E. Y. L. N. T. J. Bergeron, «Static Detection of Malicious Code in Executable Programs,» *Int. J. of Req. Eng,* 2001 .

[5] P. C. κ. S. M. A. H. Sung, «Static Analyzer of Vicious Executables (SAVE),» *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04) ,* 2004.

[6] R. I. K. S. M. L. R. K. C. J.C. Rabek, «Detection of Injected, Dynamically Generated, and Obfuscated Malicious Code,» *Proceedings of the 2003 ACM workshop on Rapid malcode,* 2003.

[7] T. I. T. S. T. I. Akira Mori, «A Tool for Analyzing and Detecting Malicious Mobile Code,» *Proceedings of the 28th international conference on Software engineering,* pp. 831-834, 2006.

[8] M. B. D. D. P. B. R. Sekar, «A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors,» *IEEE Symposium on Security and Privacy,* 2001.

[9] J. G. A. K. S. A. κ. S. D. T. Daniel R. Ellis, «A Behavioral Approach to Worm Detection,» *WORM '04 Proceedings of the 2004 ACM workshop on Rapid malcode,* pp. 43-53, 29 Οκτωβρίου 2004.

[10] D. D. David Wagner, «Intrusion Detection via Static Analysis,» *SP '01 Proceedings of the 2001 IEEE Symposium on Security and Privacy,* p. 156, 14-16 Μαΐου 2001.

[11] A. G. J. F. T. S. A. T. H. Y. S. Z. R. Sekar, «Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions,» *CCS '02 Proceedings of the 9th ACM conference on Computer and communications security,* pp. 265-274, 18-22 Νοεμβρίου 2002.

[12] S. J. S. Ke Wang, «Anomalous Payload-based Network Intrusion Detection,» *RAID 2004: Recent Advances in Intrusion Detection,* pp. 203-222, 2004.

[13] J. L. S. D. G. Edward Suh, «Secure Program Execution via Dynamic Information Flow Tracking,» *ASPLOS XI Proceedings of the 11th international conference on Architectural support for programming languages and operating systems,* pp. 85-96, 07-13 Οκτωβρίου 2004.

[14] A. P. Wes Masri, «Using Dynamic Information Flow Analysis to Detect,» *SESS '05 Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications,* pp. 1-7, 15-16 Μαίου 2005.

[15] S. J. Mihai Christodorescu, «Static Analysis of Executables to Detect Malicious Pattern,» *12th USENIX Security Symposium,* 2003.

[16] M. D. H. D. Andreas Wespi, «Intrusion Detection Using Variable-Length Audit Trail Patterns,» *RAID '00 Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection,* pp. 110-129, 02-04 Οκτωβρίου 2000.

[17] J. A.-F. Carol Taylor, «NATE - Network Analysis of Anomalous Traffic Events, A Low-Cost Approach,» *NSPW '01 Proceedings of the 2001 workshop on New security paradigms,* pp. 89-96, 10-13 Σεπτεμβρίου 2001.

[18] T. B. M. S. R. Sekar, «On Preventing Intrusions by Process Behavior Monitoring,» *ID'99 Proceedings of the 1st conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1,* p. 4, 09-12 Απριλίου 1999.

[19] S. J. S. Wenke Lee, «Data Mining Approaches for Intrusion Detection,» *SSYM'98 Proceedings of the 7th conference on USENIX Security Symposium - Volume 7,* p. 6, 26-29 Ιανουαρίου 1998.

[20] Y. O. S. G. Izuru Sato, «An Improved Intrusion Detecting Method Based on Process Profiling,» *IPSJ Journal,* τόμ. 43, αρ. 11, pp. 3316-3326, Νοέμβριος 2002.

[21] M. R. K. L. Calvin Ko, «Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach,» *In Proceedings of 1997 IEEE Symposium on Security and Privacy,* 4-7 Μαΐου 1997.