ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	*Αυτοματοποιημένη Ανάλυση Binary Αρχείων: Ανάπτυξη Script για την Ανίχνευση Ευπαθειών* **Automated Binary File Analysis: Developing a Script for Vulnerability Detection**
Ονοματεπώνυμο Φοιτητή	*Μάλλιος Κωνσταντίνος*
Πατρώνυμο	*Τριαντάφυλλος*
Αριθμός Μητρώου	*Π20118*
Επιβλέπων	**Κοτζανικολάου Παναγιώτης,** *Καθηγητής*

Ημερομηνία Παράδοσης **Φεβρουάριος 2026**

Copyright ©

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα κ. Δημήτριο Κούτρα και καθηγητή μου κ. Παναγιώτη Κοτζανικολάου που με καθοδήγησαν με την εμπειρική κατάρτιση και τις εύστοχες παρατηρήσεις τους μέχρι την επιτυχή υλοποίηση της πτυχιακής μου εργασίας.

Θα ήθελα, επίσης, να ευχαριστήσω την οικογένεια μου αλλά και τους συναδέλφους μου , που ξεκινήσαμε το ακαδημαϊκό αυτό ταξίδι, για την ψυχολογική αλληλοϋποστήριξη όλων αυτών των χρόνων.

Περίληψη

Παρά το γεγονός ότι αποτελεί πρόβλημα εδώ και δεκαετίες, η εκμετάλλευση δυαδικών (binary exploitation) παραμένει σοβαρό ζήτημα στην ασφάλεια των υπολογιστών. Αυτό οφείλεται κυρίως στην ύπαρξη σφαλμάτων διαφθοράς μνήμης σε προγράμματα που είναι γραμμένα σε ανασφαλείς αλλά αναντικατάστατες γλώσσες προγραμματισμού, όπως η C και η C++. Το λογισμικό συχνά γράφεται για έναν συγκεκριμένο σκοπό και επαναχρησιμοποιείται για άλλους. Αυτό συμβαίνει συχνά με τη μορφή προ-μεταγλωττισμένων βιβλιοθηκών. Αυτή είναι σίγουρα η περίπτωση στα περιβάλλοντα ICS, όπου οι ενημερώσεις είναι σπάνιες και το κόστος διακοπής λειτουργίας είναι υψηλό. Η επαναχρησιμοποίηση κώδικα έχει σίγουρα τα οφέλη της, όπως η μείωση του χρόνου ανάπτυξης κώδικα ή, αν η βιβλιοθήκη που χρησιμοποιείται είναι ανοικτού κώδικα, το πλεονέκτημα του να ελέγχεται από πολλούς ανθρώπους για σφάλματα πριν από τη χρήση. Ωστόσο, η επαναχρησιμοποίηση κώδικα έχει και τα μειονεκτήματά της. Για παράδειγμα, μια βιβλιοθήκη κώδικα μπορεί να έχει αναπτυχθεί πριν από πολλά χρόνια και να εξακολουθεί να χρησιμοποιείται σε ένα συγκεκριμένο έργο, ενώ ενδέχεται να μην υποστηρίζεται πλέον και να χρησιμοποιείται απλώς επειδή «δουλεύει». Επιπλέον, άλλοι παλιοί και ξεχασμένοι τμηματικοί κώδικες μπορεί να βρουν τον δρόμο τους σε ένα έργο, καθώς το έργο επεξεργάζεται από διάφορους προγραμματιστές. Το πρόβλημα με αυτά τα παραδείγματα είναι ότι αυτά τα τμήματα κώδικα μπορεί, εν αγνοία των προγραμματιστών, να φέρουν ευπάθειες μαζί τους. Αυτές οι ευπάθειες μπορούν, με τη σειρά τους, να θέσουν σε κίνδυνο ολόκληρο το έργο. Η παρούσα εργασία εστιάζει στη στατική ανάλυση binary αρχείων, με στόχο την ανάπτυξη ενός αυτοματοποιημένου εργαλείου που θα διευκολύνει τον εντοπισμό ευπαθειών και την αξιολόγηση της ασφάλειας.

Λέξεις – κλειδιά: Δυαδικά Αρχεία, Στατική ανάλυση, Αυτοματοποιημένη Αξιολόγηση Ασφάλειας, Ανίχνευση Ευπαθειών, Εργαλεία Γραμμής Εντολών.

Abstract

Despite being a problem for decades, binary exploitation remains a serious issue in computer security. This is mainly due to the existence of memory corruption bugs in programs written in insecure but irreplaceable programming languages, such as C and C++. Software is often written for one specific purpose and reused for others. This often happens in the form of pre-compiled libraries. This is certainly the case in ICS environments, where updates are rare and the cost of downtime is high. Code reuse certainly has its benefits, such as reducing code development time or, if the library used is open source, the advantage of being checked by many people for errors before use. However, code reuse also has its disadvantages. For example, a code library may have been developed many years ago and is still used in a particular project, while it may no longer be supported and is used simply because it "works". In addition, other old and forgotten pieces of code may find their way into a project as the project is worked on by different developers. The problem with these examples is that these pieces of code may, unbeknownst to the developers, carry vulnerabilities with them. These vulnerabilities can, in turn, compromise the entire project. This paper focuses on static binary analysis, with the aim of developing an automated tool that will facilitate vulnerability detection and security assessment.

Key – words: Binary Files, Static Analysis, Automated Security Assessment, Vulnerability Detection, Command Line Tools.

Περιεχόμενα

Πίνακας Πινάκων.

Πίνακας Εικόνων

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή στην έννοια των Binary Αρχείων

1.1 Τι είναι ένα δυαδικό αρχείο;

Ένα δυαδικό αρχείο είναι ένα αρχείο του οποίου το περιεχόμενο βρίσκεται σε δυαδική μορφή, αποτελούμενο από μια σειρά διαδοχικών bytes, το καθένα από τα οποία έχει μήκος οκτώ bits. Το περιεχόμενο πρέπει να ερμηνεύεται από ένα πρόγραμμα ή έναν επεξεργαστή υλικού που κατανοεί εκ των προτέρων ακριβώς πώς είναι μορφοποιημένο το περιεχόμενο και πώς να διαβάσει τα δεδομένα. Τα δυαδικά αρχεία περιλαμβάνουν ένα ευρύ φάσμα τύπου αρχείων, όπως executables, βιβλιοθήκες, graphics, βάσεις δεδομένων, archives και πολλά άλλα.

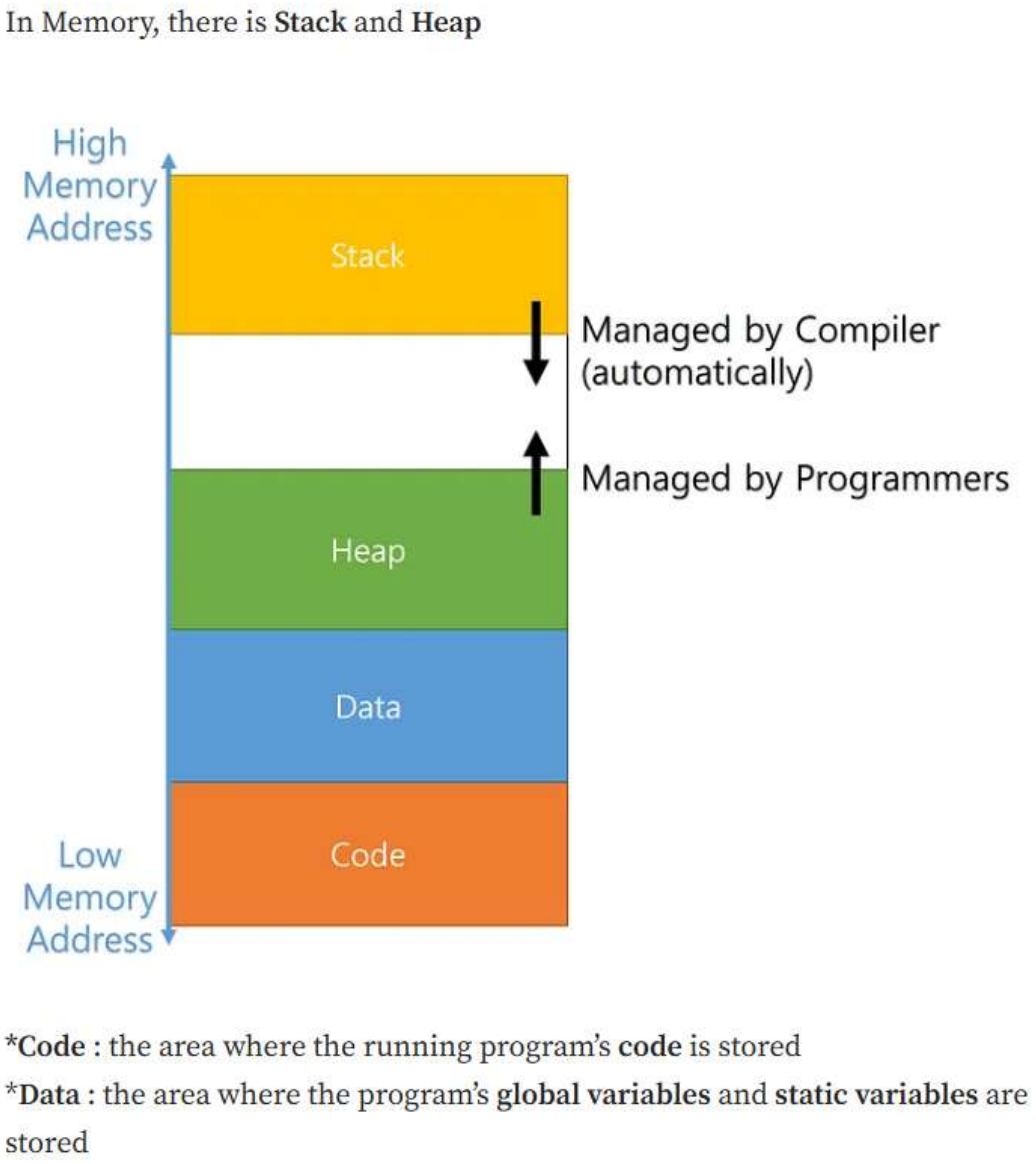
1.2 Πώς χρησιμοποιείται ένα δυαδικό αρχείο;

Τα binaries αρχεία δεν είναι αναγνώσιμα από τον άνθρωπο και απαιτούν ειδικό πρόγραμμα ή επεξεργαστή υλικού που γνωρίζει πώς να διαβάσει τα δεδομένα μέσα στο αρχείο. Μόνο τότε μπορούν να κατανοηθούν και να επεξεργαστούν σωστά οι εντολές που είναι κωδικοποιημένες στο δυαδικό περιεχόμενο.

1.3 Πως ορίζεται η εκμετάλλευση binary αρχείων;

Η εκμετάλλευση δυαδικών αρχείων (binary exploitation) είναι μία μέθοδος εντοπισμού και αξιοποίησης ευπαθειών σε προγράμματα υπολογιστών, με στόχο την τροποποίηση ή τη διακοπή των προβλεπόμενων λειτουργιών τους. Αυτές οι ευπάθειες μπορεί να οδηγήσουν σε παράκαμψη ελέγχου ταυτότητας, διαρροή πληροφοριών, ή ακόμα και σε συνθήκες remote code execution.

Ένα μεγάλο μέρος της εκμετάλλευσης δυαδικών αρχείων πραγματοποιείται στη στοίβα (stack), ενίοτε στη σωρό (heap), και μερικές φορές στον πυρήνα του συστήματος (kernel space). **Η στοίβα** είναι η περιοχή μνήμης όπου αποθηκεύονται προσωρινές μεταβλητές που δημιουργούνται από συναρτήσεις. **Η σωρός**, αντίθετα, είναι μια περιοχή μνήμης που μπορεί να δεσμευτεί δυναμικά κατά την εκτέλεση του προγράμματος. Η εκμετάλλευση αυτών των περιοχών μνήμης μπορεί να επιτρέψει σε έναν εισβολέα να παρακάμψει περιορισμούς και να εκτελέσει κακόβουλο κώδικα, τροποποιώντας την κανονική συμπεριφορά του προγράμματος.



Εικόνα 1. Η αρχιτεκτονική της Μνήμης.

Μεταξύ αυτών, οι ευπάθειες σε επίπεδο λογισμικού είναι από τις πιο γνωστές και μελετημένες. Περιλαμβάνουν **Stack-based buffer overflows** που προκύπτουν όταν ένα πρόγραμμα γράφει περισσότερα δεδομένα σε μια θέση στη μνήμη από όσα έχει προβλεφθεί, παραβιάζοντας γειτονικές περιοχές μνήμης και επιτρέποντας σε έναν εισβολέα να εκτελέσει κακόβουλο κώδικα. Επίσης, περιλαμβάνουν **Use-after-free** οι οποίες εμφανίζονται σε προγράμματα με κακή διαχείριση μνήμης στη σωρό. Ειδικότερα, συμβαίνουν όταν ο κώδικας προσπαθεί να χρησιμοποιήσει μια περιοχή μνήμης που έχει ήδη ελευθερωθεί, δίνοντας τη δυνατότητα σε έναν εισβολέα να εισάγει κακόβουλα δεδομένα στη μνήμη.

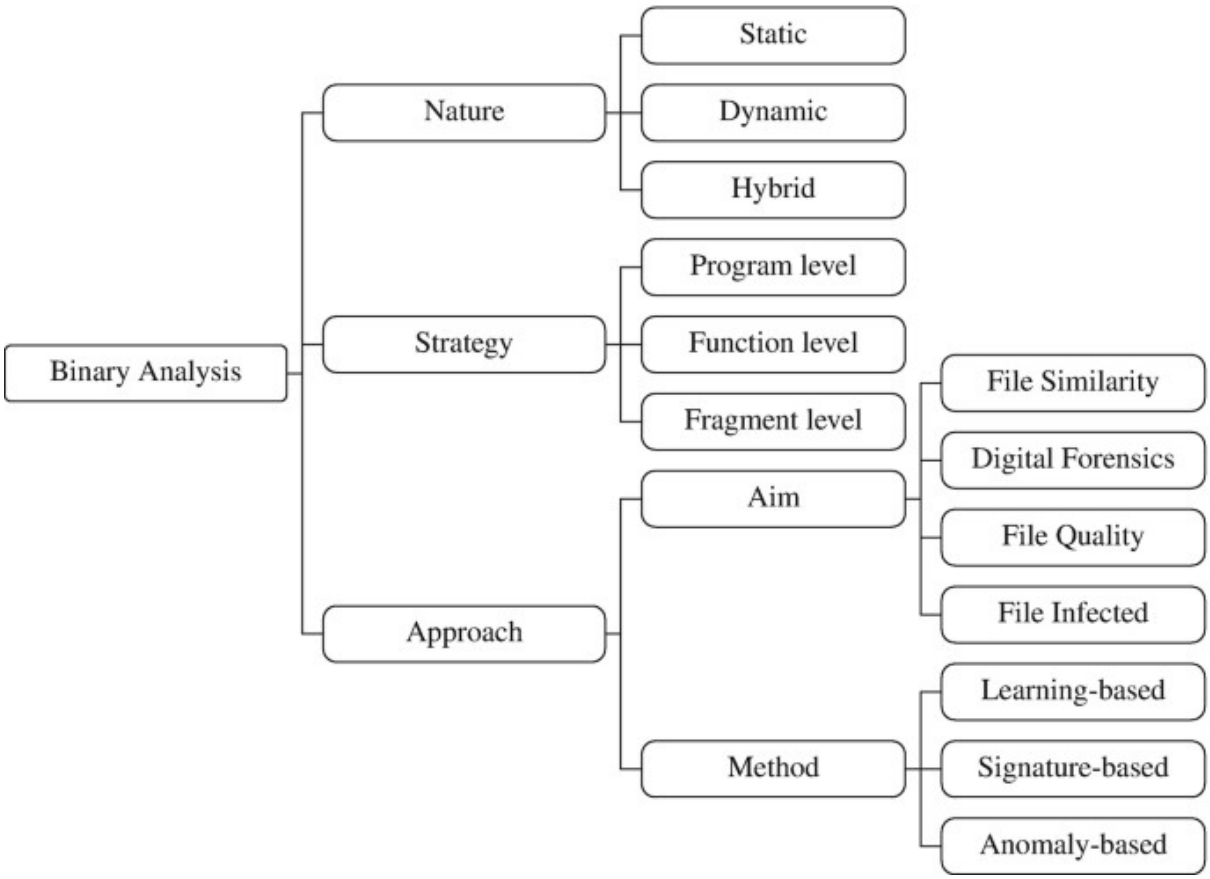
Αυτές οι αδυναμίες συνήθως επιτρέπουν σε έναν εισβολέα να αποκτήσει πλήρη έλεγχο ενός προγράμματος. Σε χειρότερα σενάρια, ο εισβολέας μπορεί να αποκτήσει δικαιώματα διαχειριστή ή υπερχρήστη, εάν το πρόγραμμα εκτελείται με τέτοια προνόμια. Αυτό μπορεί να οδηγήσει σε πλήρη παραβίαση του συστήματος. Αναλυτικότερη αναφορά θα γίνει και στις ευπάθειες library exploitation, format string, integer underflow, race condition, heap overflow καθώς καθιστούν την ύπαρξη τους σπουδαίο κίνδυνο στα υπολογιστικά μας συστήματα.

1.4 Πως δημιουργήθηκε η ανάγκη για ανάλυση binary αρχείων ;

Ο πηγαίος κώδικας δεν είναι πάντα διαθέσιμος για ανάλυση. Για παράδειγμα, ορισμένες εταιρείες αγοράζουν το firmware σε δυαδική μορφή για να το ενσωματώσουν στο hardware των προϊόντων τους. Ένα άλλο παράδειγμα είναι οι εταιρείες λογισμικού που αξιοποιούν κώδικα και βιβλιοθήκες τρίτων, όπως frameworks, GUI και βάσεις δεδομένων, για να εμπλουτίσουν τον κώδικά τους, και αυτές οι βιβλιοθήκες συχνά δεν περιέχουν πηγαίο κώδικα. Όποια και αν είναι η περίπτωση, εξακολουθεί να είναι σημαντικό οι καταναλωτές αυτών των δυαδικών αρχείων να κατανοούν το περιεχόμενό τους.

Λόγω της φύσης τους, το κακόβουλο λογισμικό απαιτεί εκτενή ανάλυση και αντίστροφη μηχανική. Ωστόσο, η ανάλυση δυαδικού κώδικα μπορεί επίσης να χρησιμοποιηθεί για την εύρεση προβλημάτων, συμπεριλαμβανομένων των αδυναμιών ασφαλείας που εκδηλώνονται μόνο σε επίπεδο δυαδικού κώδικα, ακόμη και αν έχετε διαθέσιμο τον πηγαίο κώδικα.

Η ανάλυση δυαδικών αρχείων (Binary Analysis) επικεντρώνεται στη εξέταση εκτελέσιμων binaries χωρίς να απαιτείται η εκτέλεσή τους. Εφαρμόζεται κυρίως σε περιπτώσεις όπου δεν υπάρχει πρόσβαση στον πηγαίο κώδικα και στοχεύει στην αποκάλυψη εσωτερικής πληροφορίας, όπως ευπάθειες, ενσωματωμένα κακόβουλα δεδομένα, παραποιήσεις, εξαρτήσεις και μη ορατές λειτουργίες. Η χρήση της είναι καθοριστική για την ασφάλεια στο software supply chain, καθώς επιτρέπει την αξιολόγηση λογισμικού τρίτων και τον εντοπισμό πιθανών κινδύνων πριν από τη διανομή ή την ενσωμάτωση. Η ανάλυση αυτή εντοπίζει κρυπτογράφηση ή συμπίεση, και επισημαίνει πιθανές αποκλίσεις από τα πρότυπα ασφαλείας. Χρησιμοποιείται ευρέως από ομάδες κυβερνοασφάλειας, software engineers και φορείς που απαιτείται να συμμορφώνονται με κανονισμούς και απαιτήσεις ασφαλείας.



Εικόνα 2. Στάδια ανάλυσης Δυαδικών Αρχείων.

Συνήθως, οι προγραμματιστές λογισμικού μπορούν να δημιουργήσουν οδηγούς για συσκευές υλικού βάσει της λεπτομερούς τεκμηρίωσής τους, αλλά πολλοί κατασκευαστές τρίτων παρέχουν μόνο ένα εκτελέσιμο αρχείο σε δυαδική μορφή, που αναφέρεται ως δυαδικό blob. Τα **BLOB** είναι χρήσιμα για την αποθήκευση μεγάλων δυαδικών δεδομένων στις βάσεις δεδομένων, κυρίως όταν πρόκειται για αρχεία πολυμέσων. Αν και προσφέρουν οικονομική αποθήκευση και απρόσκοπτη ενσωμάτωση με τα αντίγραφα ασφαλείας, τα μειονεκτήματά τους περιλαμβάνουν την ελάχιστη υποστήριξη σε όλες τις βάσεις δεδομένων και την ανεπαρκή απόδοση. Η ακριβής λειτουργία και η ποιότητα ασφαλείας των δυαδικών blobs είναι άγνωστες, πράγμα που σημαίνει ότι απαιτούν από τους χρήστες να εμπιστεύονται τους τρίτους προμηθευτές για την ασφάλεια του συστήματός τους. . Είναι απαραίτητα για να κατανοήσουμε αν και πώς να χρησιμοποιήσουμε τα BLOB για βέλτιστες λύσεις αποθήκευσης.

Εικόνα 3. Binary Large Object

Σκοπός και Στόχοι της Παρούσας Εργασίας

Λόγω των αυξανόμενων αναγκών για κατανόηση και έλεγχο των binary αρχείων, η παρούσα εργασία στοχεύει στην ανάπτυξη ενός αυτοματοποιημένου εργαλείου στατικής ανάλυσης δυαδικών αρχείων. Συγκεκριμένα, προτείνεται δημιουργία ενός script που αξιοποιεί υπάρχοντα command-line tools και libraries, με σκοπό την γρήγορη και αξιόπιστη εξαγωγή πληροφοριών από binaries. Η εργασία δεν επιχειρεί να αντικαταστήσει άλλα ήδη υπάρχοντα εργαλεία, αλλά να προσφέρει μια εύκολη και κατανοητή λύση που να επιτρέπει την εξαγωγή κρίσιμων πληροφοριών από binaries, όπως μεταδεδομένα, εξαρτήσεις, γνωστές ευπάθειες, δομικά χαρακτηριστικά και βασικούς μηχανισμούς ασφάλειας, χωρίς την ανάγκη εκτέλεσής τους.

1.5 Αναφορές σε παρόμοιες έρευνες.

Στο άρθρο "Recognizing Functions in Binaries with Neural Networks" (USENIX Security Symposium), οι Shin, Williams και Song πρότειναν μια σύγχρονη προσέγγιση που ξεπερνά τους παραδοσιακούς περιορισμούς της ανάλυσης δυαδικών. Λόγω του stripping, το παραδοσιακό function recognition είναι εξαιρετικά δύσκολος. Οι ερευνητές εκπαίδευσαν ένα νευρωνικό δίκτυο (Recurrent Neural Network - RNN) για να αναγνωρίζει τα όρια συναρτήσεων και να επαναφέρει πληροφορίες που έχουν αφαιρεθεί, αναλύοντας ακολουθίες από instructions. (Eui Chul Richard Shin, 2015)

Το άρθρο "angr: A Next-Generation Binary Analysis Platform" από τους Shoshitaishvili et al., που παρουσιάστηκε στο Annual Computer Security Applications Conference (ACSAC), περιγράφει λεπτομερώς το angr. Αυτό το πλαίσιο συνδυάζει μια πληθώρα από τεχνικές ανάλυσης υπό μια ενιαία, εύχρηστη διεπαφή. Το angr εκτελεί ανάλυση ευπάθειας, automatic exploit generation, και deobfuscation κώδικα μέσω ισχυρής συμβολικής εκτέλεσης και δειγματοληψίας της χώρας των διαφόρων εκτελέσεων. (Wang & Shoshitaishvili, n.d.)

Οι Cha, Avgerinos, Rebert, και Brumley παρουσίασαν το Mayhem στο συμβόλαιο "Unleashing Mayhem on Binary Code" (IEEE S&P). Το Mayhem συνδυάζει συμβολική εκτέλεση με δειγματοληψία concolic execution για να ανακαλύψει αυτόματα και να επαληθεύσει εκμεταλλεύσιμες ευπάθειες σε δυαδικά αρχεία. Η ικανότητά του να παράγει working exploits για τις ευπάθειες που βρίσκει το καθιστά ένα από τα πιο προηγμένα συστήματα της εποχής του. (Cha, Avgerinos, Rebert, & Brumley, 2012)

Οι Dinaburg, Royal, Sharif, και Lee πρότειναν το Ether στο άρθρο "Ether: Malware Analysis via Hardware Virtualization Extensions" (ACM CCS). Το Ether χρησιμοποιεί επεκτάσεις εικονικοποίησης υλικού (Intel VT-x) για να παρακολουθεί την εκτέλεση ύποπτου λογισμικού από "κάτω" από το λειτουργικό σύστημα, με τρόπο αόρατο για το malware. Αυτή η προσέγγιση ήταν πρωτοποριακή στη δυναμική ανάλυση malware που αντιστέκεται στην ανίχνευση. (Artem Dinaburg, 2008)

Στο άρθρο "Leveraging ELF Metadata for Malware Detection", διάφοροι ερευνητές διερεύνησαν την αποτελεσματικότητα της χρήσης δημοφιλών μετρικών ως features για μοντέλα Machine Learning που προβλέπουν αν ένα ELF δυαδικό είναι κακόβουλο. (Leveraging ELF Metadata for Malware Detection, 2020)

Οι παραπάνω ερευνητικές εργασίες αποτελούν την αφετηρία της εξέλιξης της αυτοματοποιημένης ανάλυσης δυαδικών, από τα θεμέλια της συμβολικής εκτέλεσης και της δυαδικής instrumentation έως τις σύγχρονες τεχνικές μηχανικής μάθησης. Η παρούσα πτυχιακή εργαία, αν και χαμηλότερων δυνατοτήτων, εντάσσεται σε αυτό το πλαίσιο, προτείνοντας μια πρακτική και εύκολα επεκτάσιμη μέθοδο που αξιοποιεί υπάρχοντα command line tools για να παράγει γρήγορα και αξιόπιστα αποτελέσματα.

ΚΕΦΑΛΑΙΟ 2 Εισαγωγή στην Ανάλυση Δυαδικών αρχείων

Πρόλογος

Η δυαδική ανάλυση είναι η διαδικασία εξέτασης των ιδιοτήτων των δυαδικών αρχείων, συμπεριλαμβανομένων των εντολών που περιέχουν και των δεδομένων που είναι κωδικοποιημένα σε δυαδική μορφή, για να μάθουμε περισσότερα σχετικά με το περιεχόμενο του αρχείου ή τον σκοπό του προγράμματος.

Μπορούμε να κατηγοριοποιήσουμε την ανάλυση δυαδικού κώδικα σε δύο θεμελιώδεις ομάδες ανάλογα με το πότε πραγματοποιούμε την ανάλυση και σε άλλες δύο σημαντικές ομάδες με βάση το επίπεδο πολυπλοκότητας των μεθόδων που επιλέγουμε για να πραγματοποιήσουμε την ανάλυση.

Η στατική ανάλυση περιλαμβάνει την εξέταση του εκτελέσιμου δυαδικού αρχείου χωρίς να το εκτελέσουμε. Τα εργαλεία στατικής ανάλυσης μπορεί να εντοπίσουν σφάλματα ή ευπάθειες ασφαλείας απλώς διαβάζοντας ένα πρόγραμμα.

Η δυναμική ανάλυση περιλαμβάνει την παρακολούθηση του προγράμματος καθώς εκτελείται σε ένα πραγματικό ή σχεδόν πανομοιότυπο περιβάλλον, αν χρειαστεί, χρησιμοποιώντας πλήρη συστήματα ή εξομοίωση λειτουργίας χρήστη. Τα εργαλεία δυναμικής ανάλυσης ενσωματώνουν τον κώδικα ανάλυσης στο πρόγραμμα, ο οποίος αποθηκεύει πληροφορίες σχετικά με την εκτέλεση του προγράμματος ως μεταδεδομένα.

2.1 Τι είναι η Στατική Ανάλυση Δυαδικών αρχείων;

Η στατική ανάλυση είναι μια σημαντική μέθοδος για την αξιολόγηση της ασφάλειας του λογισμικού, καθώς επιτρέπει τον εντοπισμό σφαλμάτων και ευπαθειών χωρίς την ανάγκη εκτέλεσης του κώδικα. Αντί να εξετάζεται η συμπεριφορά του προγράμματος κατά την εκτέλεση, αναλύεται ο πηγαίος κώδικας, το bytecode ή τα binaries, με στόχο την ανίχνευση πιθανών σφαλμάτων, ευπαθειών ή αποκλίσεων από τις βέλτιστες πρακτικές προγραμματισμού. Η σημασία της στατικής ανάλυσης έγκειται κυρίως στη δυνατότητα εντοπισμού ευπαθειών πριν την εκτέλεση του λογισμικού, γεγονός που επιτρέπει την έγκαιρη διόρθωση των προβλημάτων, μειώνοντας το κόστος και τον χρόνο επισκευής. Λειτουργεί ως ένας μηχανισμός "δεύτερης ματιάς", ο οποίος ανιχνεύει αυτά τα λάθη πριν φτάσουν στο στάδιο της παραγωγής, μειώνοντας την πιθανότητα εμφάνισης κρίσιμων προβλημάτων.

Η στατική ανάλυση εφαρμόζεται συνήθως κατά τη διάρκεια του κύκλου ανάπτυξης λογισμικού (SDLC), διασφαλίζοντας ότι τα προβλήματα αντιμετωπίζονται από νωρίς, πριν γίνουν πιο δύσκολα και δαπανηρά στη διόρθωσή τους. Ο κύκλος ζωής ανάπτυξης λογισμικού για τα δυαδικά αρχεία περιλαμβάνει μια δομημένη διαδικασία για τη δημιουργία, τη δοκιμή και την ανάπτυξη λογισμικού, εξασφαλίζοντας ποιότητα και αποδοτικότητα. Αυτή η διαδικασία περιλαμβάνει συνήθως τον σχεδιασμό, την ανάλυση απαιτήσεων, τον σχεδιασμό, την κωδικοποίηση, τη δοκιμή, την ανάπτυξη και τη συντήρηση. Η σύνταξη και η συσκευασία είναι επίσης κρίσιμα βήματα, καθώς μετατρέπουν τον πηγαίο κώδικα σε εκτελέσιμα δυαδικά αρχεία. Ουσιαστικά, η ασφάλεια SDLC είναι μια ολοκληρωμένη προσέγγιση στην ανάπτυξη λογισμικού που δίνει προτεραιότητα στην ασφάλεια σε κάθε στάδιο, με αποτέλεσμα πιο ασφαλή, αξιόπιστα και οικονομικά αποδοτικά προϊόντα λογισμικού.

Εικόνα 4. Κύκλος ανάπτυξης λογισμικού.

Η στατική ανάλυση επιτρέπει τη χρήση εξειδικευμένων εργαλείων που αυτοματοποιούν τη διαδικασία ανίχνευσης ευπαθειών. Εργαλεία όπως το SonarQube, το Checkmarx και το Fortify αναλύουν τον πηγαίο κώδικα και παρέχουν λεπτομερείς αναφορές για πιθανά σφάλματα και παραβιάσεις ασφαλείας. Τα εργαλεία αυτά βοηθούν επίσης στον εντοπισμό γνωστών κενών ασφαλείας, στην ανάλυση της δομής και της ποιότητας του κώδικα και στον έλεγχο συμμόρφωσης με διεθνή πρότυπα ασφαλείας, όπως το OWASP Top 10 και το SANS CWE Top 25.

2.2 Πλεονεκτήματα της Στατικής Ανάλυσης

Η στατική ανάλυση αποτελεί μια κρίσιμη τεχνική για τον έλεγχο και την αξιολόγηση της ποιότητας και της ασφάλειας του λογισμικού πριν από την εκτέλεσή του. Παρακάτω παρουσιάζονται τα βασικά της πλεονεκτήματα με λεπτομερή ανάλυση.

Ασφάλεια χωρίς εκτέλεση

Η στατική ανάλυση εξετάζει τον πηγαίο ή μεταγλωττισμένο κώδικα του λογισμικού χωρίς να χρειάζεται η εκτέλεσή του. Αυτό παρέχει τη δυνατότητα εντοπισμού προβλημάτων ασφαλείας και σφαλμάτων προγραμματισμού σε πρώιμο στάδιο, προτού το λογισμικό εισέλθει σε παραγωγικό περιβάλλον. Για παράδειγμα, μπορεί να ανιχνεύσει μη ασφαλείς πρακτικές, όπως η απροστάτευτη διαχείριση μνήμης ή η μη ασφαλής πρόσβαση σε μεταβλητές.

Ολοκληρωμένη κάλυψη

Σε αντίθεση με τη δυναμική ανάλυση, η οποία εξαρτάται από τα σενάρια δοκιμών, η στατική ανάλυση εξετάζει ολόκληρο τον κώδικα, συμπεριλαμβανομένων των τμημάτων που ενδέχεται να μην εκτελούνται σε τυπικές συνθήκες λειτουργίας. Αυτό είναι ιδιαίτερα χρήσιμο για την ανίχνευση «νεκρού κώδικα» (dead code) και συμβάλλει στην ενίσχυση της συνολικής ασφάλειας και ποιότητας του λογισμικού.

Μείωση κόστους

Ο εντοπισμός και η διόρθωση σφαλμάτων κατά τα πρώτα στάδια της ανάπτυξης λογισμικού είναι σημαντικά φθηνότερος από την επίλυσή τους μετά την κυκλοφορία του λογισμικού στην παραγωγή. Η έγκαιρη αντιμετώπιση των σφαλμάτων μειώνει την πιθανότητα διακοπής υπηρεσιών, αποζημιώσεων ή ζητημάτων συμμόρφωσης με κανονισμούς ασφαλείας.

Βελτίωση διαφάνειας

Η στατική ανάλυση προσφέρει μια ολοκληρωμένη εικόνα της κατάστασης του κώδικα, της ποιότητάς του και των ενδεχόμενων προβλημάτων του. Αυτό επιτρέπει στους προγραμματιστές να τεκμηριώνουν καλύτερα τις αλλαγές τους, να παρακολουθούν την εξέλιξη του λογισμικού και να διατηρούν υψηλά επίπεδα ποιότητας κώδικα. Επιπλέον, ενισχύει τη συμμόρφωση με πρότυπα προγραμματισμού και ασφάλειας.

2.3 Περιορισμοί της Στατικής Ανάλυσης

Παρά τα σημαντικά πλεονεκτήματά της, η στατική ανάλυση δεν είναι πανάκεια και συνοδεύεται από συγκεκριμένους περιορισμούς.

Ψευδώς θετικά και ψευδώς αρνητικά

Τα εργαλεία στατικής ανάλυσης συχνά ανιχνεύουν προβλήματα που στην πραγματικότητα δεν αποτελούν ευπάθειες (ψευδώς θετικά) ή αποτυγχάνουν να εντοπίσουν κρίσιμα σφάλματα (ψευδώς αρνητικά). Αυτό μπορεί να οδηγήσει σε σύγχυση των προγραμματιστών και να μειώσει την αποδοτικότητα της διαδικασίας ελέγχου.

Έλλειψη δυναμικής ανάλυσης

Δεδομένου ότι η στατική ανάλυση δεν εξετάζει το λογισμικό κατά την εκτέλεσή του, δεν μπορεί να εντοπίσει προβλήματα που σχετίζονται με την πραγματική αλληλεπίδραση του λογισμικού με το περιβάλλον του ή με άλλες εφαρμογές. Για παράδειγμα, δεν μπορεί να ανιχνεύσει σφάλματα που εμφανίζονται μόνο σε συγκεκριμένες συνθήκες εκτέλεσης, όπως διαγωνιστικά προβλήματα (race conditions) ή προβλήματα διαχείρισης μνήμης που εμφανίζονται κατά τη λειτουργία του συστήματος.

Απαίτηση παραμετροποίησης

Η αποτελεσματική χρήση εργαλείων στατικής ανάλυσης απαιτεί σωστή ρύθμιση και παραμετροποίηση. Η διαδικασία αυτή μπορεί να είναι χρονοβόρα και να απαιτεί εξειδικευμένη γνώση από την ομάδα ανάπτυξης ή τον υπεύθυνο ασφαλείας. Επιπλέον, η επιλογή του κατάλληλου εργαλείου και η κατανόηση των αποτελεσμάτων της ανάλυσης αποτελούν προκλήσεις που πρέπει

να αντιμετωπιστούν. Η συνδυαστική χρήση στατικής και δυναμικής ανάλυσης μπορεί να ενισχύσει την ασφάλεια και την ποιότητα του λογισμικού, επιτυγχάνοντας μια ισορροπημένη προσέγγιση στον έλεγχο λογισμικού.

2.4 Διαφορές στατικής και δυναμικής ανάλυσης binary αρχείων

Τόσο η στατική ανάλυση όσο και η ανάλυση χρόνου εκτέλεσης είναι βασικές τεχνικές στον κύκλο ζωής ανάπτυξης λογισμικού. Η στατική ανάλυση προσφέρει το πλεονέκτημα της έγκαιρης ανίχνευσης και της ολοκληρωμένης κάλυψης χωρίς εκτέλεση, αλλά δεν αντιμετωπίζει ζητήματα που αφορούν συγκεκριμένα τον χρόνο εκτέλεσης. Από την άλλη πλευρά, η ανάλυση χρόνου εκτέλεσης παρέχει πληροφορίες για την πραγματική συμπεριφορά και τις δυναμικές αλληλεπιδράσεις, αλλά περιορίζεται από την εξάρτησή της από την εκτέλεση και συγκεκριμένα σενάρια δοκιμών. Συνδυάζοντας και τις δύο προσεγγίσεις, οι προγραμματιστές μπορούν να κατανοήσουν καλύτερα την ανθεκτικότητα, την απόδοση και την ασφάλεια της εφαρμογής. Πολύ συχνά, ο συνδυασμός των πλεονεκτημάτων της στατικής και της δυναμικής ανάλυσης σε μια υβριδική προσέγγιση είναι η βέλτιστη λύση.

Dynamic Analysis:

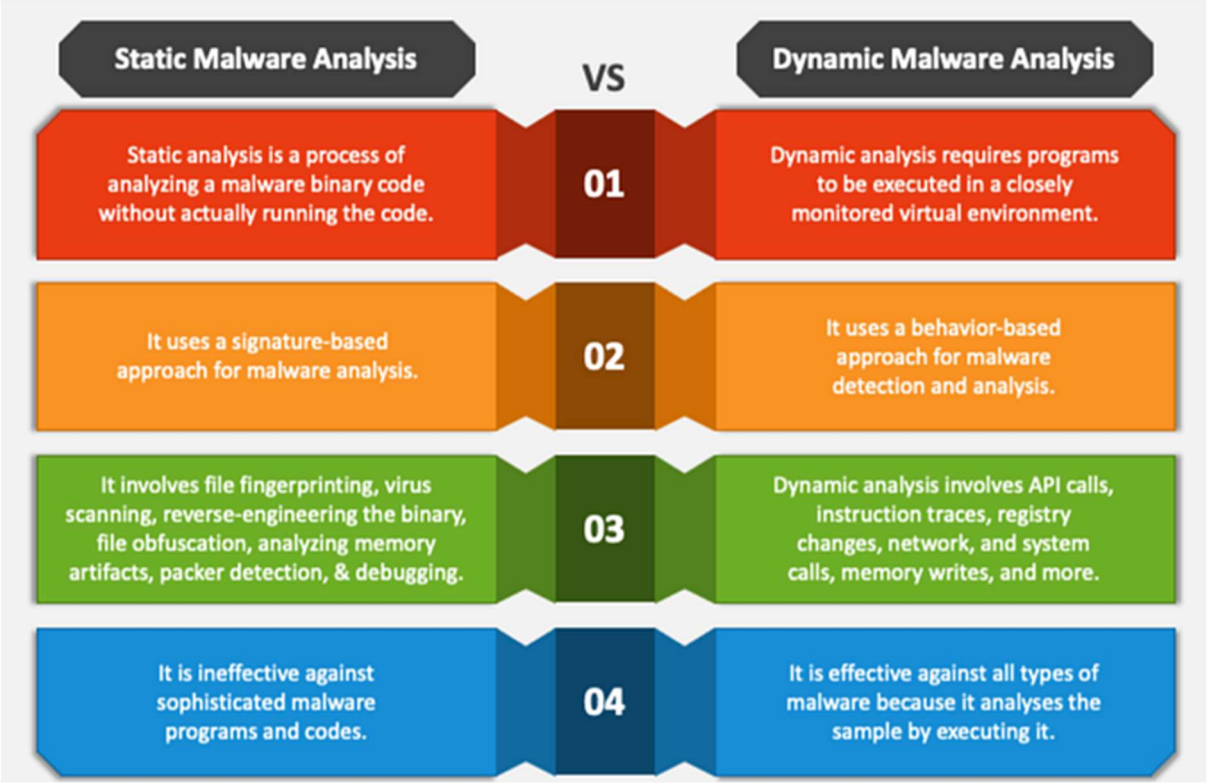
Με τη δυναμική ανάλυση, ένα ύποπτο αρχείο ενεργοποιείται σε ένα virtual machine και αναλύεται για να διαπιστωθεί τι κάνει. Το αρχείο βαθμολογείται με βάση το τι κάνει κατά την εκτέλεση, αντί να βασίζεται σε signatures για την αναγνώριση απειλών. Αυτό επιτρέπει στη δυναμική ανάλυση να αναγνωρίζει απειλές που δεν μοιάζουν με τίποτα που έχει δει ποτέ στο παρελθόν. Για τα πιο ακριβή αποτελέσματα, το sample πρέπει να έχει πλήρη πρόσβαση στο διαδίκτυο, όπως θα είχε ένα μέσο τερματικό σε ένα εταιρικό δίκτυο, καθώς οι απειλές συχνά απαιτούν εντολές και έλεγχο για να αποκαλυφθούν πλήρως. Ως μηχανισμός πρόληψης, η ανάλυση κακόβουλου λογισμικού μπορεί να απαγορεύσει την πρόσβαση στο διαδίκτυο και να προσποιηθεί response calls για να ξεγελάσει την απειλή ώστε να αποκαλυφθεί, αλλά αυτό μπορεί να είναι αναξιόπιστο και δεν αποτελεί πραγματική αντικατάσταση της πρόσβασης στο διαδίκτυο. Μπορεί να χρειαστούν αρκετά λεπτά για να ξεκινήσει ένα virtual machine, να ανοίξει το αρχείο σε αυτήν να δούμε τι κάνει, να κλείσουμε το machine και να αναλύσουμε τα αποτελέσματα. Ενώ η δυναμική ανάλυση είναι η πιο δαπανηρή και χρονοβόρα μέθοδος, είναι επίσης το μόνο εργαλείο που μπορεί να ανιχνεύσει αποτελεσματικά άγνωστες ή zero-day απειλές.

Static Analysis:

Σε αντίθεση με τη δυναμική ανάλυση, η στατική ανάλυση εξετάζει το περιεχόμενο ενός συγκεκριμένου αρχείου όπως υπάρχει στον δίσκο, και όχι όπως θα εκτελεστεί από τον τελικό χρήστη. Αναλύει τα δεδομένα, εξάγει patterns, attributes και artifacts, και επισημαίνει ανωμαλίες. Η στατική ανάλυση είναι ανθεκτική στα προβλήματα που παρουσιάζει η δυναμική ανάλυση. Είναι εξαιρετικά αποτελεσματική , διαρκεί μόνο ένα κλάσμα του δευτερολέπτου και είναι πολύ πιο οικονομική. Μπορεί επίσης να λειτουργήσει για οποιοδήποτε αρχείο, καθώς δεν υπάρχουν συγκεκριμένες απαιτήσεις, περιβάλλοντα που πρέπει να προσαρμοστούν ή εξερχόμενες επικοινωνίες που απαιτούνται από το αρχείο για να πραγματοποιηθεί η ανάλυση. Τα συμπιεσμένα

αρχεία έχουν ως αποτέλεσμα την απώλεια ορατότητας. Η στατική ανάλυση μπορεί να παρακαμφθεί σχετικά εύκολα εάν το αρχείο είναι συμπιεσμένο. Ενώ τα συμπιεσμένα αρχεία λειτουργούν καλά στη δυναμική ανάλυση, το πόσο μπορούμε να αναλύσουμε το πραγματικό αρχείο και να δούμε εξάγουμε όσες περισσότερες πληροφορίες γίνεται από αυτό, δηλαδή η ορατότητά του, χάνεται κατά τη στατική ανάλυση, καθώς η repacking του sample μετατρέπει ολόκληρο το αρχείο σε noise. Αυτό που μπορεί να εξαχθεί στατικά είναι σχεδόν τίποτα.

Η στατική και η δυναμική ανάλυση κώδικα είναι πιο αποτελεσματικές όταν χρησιμοποιούνται μαζί ως συμπληρωματικές τεχνικές. Η στατική ανάλυση κώδικα υπερέχει στον εντοπισμό σφαλμάτων κωδικοποίησης, compliance violations και ελλείψεων ασφάλειας στον πηγαίο κώδικα, ενώ η δυναμική ανάλυση κώδικα αποκαλύπτει σφάλματα κατά τη διάρκεια χρόνου εκτέλεσης, memory leaks και bottlenecks που εμφανίζονται μόνο κατά την εκτέλεση. Συνδυάζοντας τη στατική και τη δυναμική ανάλυση κώδικα, οι ομάδες software επιτυγχάνουν ολοκληρωμένη κάλυψη, διασφαλίζοντας ότι τόσο η δομή του κώδικα όσο και runtime behavior εξετάζονται απόλυτα.



Εικόνα 5. Διαφορές Στατικής και Δυναμικής Ανάλυσης.

Τι άλλο υπάρχει;

Πέρα από αυτές τις βασικές μεθόδους, πρόσθετες τεχνικές δοκιμών ασφάλειας εφαρμογών, όπως οι interactive application security testing (IAST) και οι dynamic application security testing (DAST), μπορούν να ενισχύσουν περαιτέρω την ασφάλεια ενός έργου. Τα εργαλεία IAST αναλύουν τον κώδικα σε πραγματικό χρόνο κατά την εκτέλεσή του, παρέχοντας λεπτομερείς πληροφορίες για τις ευπάθειες και τις αλληλεπιδράσεις του συστήματος. Τα εργαλεία DAST προσομοιώνουν επιθέσεις

σε εφαρμογές που εκτελούνται για να εντοπίσουν κινδύνους ασφάλειας που ενδέχεται να μην είναι ορατοί μόνο μέσω της στατικής ανάλυσης. Αξιοποιώντας αυτές τις συμπληρωματικές τεχνικές, οι ομάδες software μπορούν να εντοπίζουν πιο αποτελεσματικά τις ευπάθειες, τα σφάλματα κωδικοποίησης, τις παραβιάσεις συμμόρφωσης και τους κινδύνους ασφάλειας, με αποτέλεσμα πιο ασφαλείς και ανθεκτικές εφαρμογές

2.5 Μέθοδοι στατικής ανάλυσης

Στο παρόν κεφάλαιο, θα παρουσιαστούν δύο θεμελιώδεις μέθοδοι στατικής ανάλυσης που χρησιμοποιούνται ευρέως για την ανάλυση δυαδικών αρχείων και την ανίχνευση κακόβουλου λογισμικού. Η πρώτη είναι η ανάλυση strings, μια τεχνική εξαγωγής αναγνώσιμων ακολουθιών χαρακτήρων από δυαδικά αρχεία, που μπορεί να αποκαλύψει κρίσιμες πληροφορίες σχετικά με τη λειτουργικότητα του αρχείου. Η δεύτερη είναι η ανάλυση εντροπίας, μια μέθοδος μέτρησης της τυχαιότητας των δεδομένων σε ένα δυαδικό αρχείο, η οποία μπορεί να υποδείξει την παρουσία συμπιεσμένων ή κρυπτογραφημένων τμημάτων. Η κατανόηση αυτών των μεθόδων είναι απαραίτητη για την αποτελεσματική ανάλυση και τον εντοπισμό δυνητικών απειλών σε εκτελέσιμα αρχεία και άλλα δυαδικά δεδομένα.

2.5.1 Strings Analysis

Σε αντίθεση με τα απλά αρχεία κειμένου, τα οποία αποθηκεύουν δεδομένα ως χαρακτήρες οργανωμένους σε μηδέν ή περισσότερες γραμμές, τα δυαδικά αρχεία αποθηκεύουν δεδομένα πιο αποδοτικά, ως ακολουθία bytes. Αυτά τα bytes συνήθως ερμηνεύονται ως κάτι διαφορετικό από κειμενικούς χαρακτήρες. Τα εκτελέσιμα αρχεία συχνά αναφέρονται ως δυαδικά, αν και στην πραγματικότητα τα δυαδικά αρχεία μπορούν να περιέχουν οποιοδήποτε τύπο περιεχομένου.

Ορισμένα δυαδικά αρχεία περιέχουν σημαντικά metadata, τα οποία παρέχουν πληροφορίες για τον τρόπο ερμηνείας των δεδομένων που βρίσκονται μέσα στο αρχείο. Παραδείγματα τέτοιων metadata είναι οι υπογραφές αρχείων (file signatures) οι οποίοι μπορούν να αναγνωρίσουν τη μορφή του αρχείου, παρόμοια με τις ετικέτες στα προϊόντα ενός σούπερ μάρκετ. Επίσης, τα εκτελέσιμα αρχεία περιέχουν δυαδικό κώδικα που κάνει τον υπολογιστή να εκτελεί συγκεκριμένες εργασίες σύμφωνα με τις οδηγίες που περιλαμβάνονται στον κώδικα. Αυτός ο τρόπος αποθήκευσης καθιστά τα δυαδικά αρχεία πολύ ευέλικτα, επιτρέποντάς τους να υποστηρίζουν ένα ευρύ φάσμα λειτουργιών και δεδομένων, από εικόνες και αρχεία πολυμέσων μέχρι εκτελέσιμα προγράμματα.

Η ανάλυση strings είναι μια θεμελιώδης τεχνική στον τομέα της στατικής ανάλυσης κακόβουλου λογισμικού και γενικότερα στην ανάλυση δυαδικών αρχείων. Στην ουσία της, αφορά την εξαγωγή αναγνώσιμων ακολουθιών χαρακτήρων από δυαδικά αρχεία. Τα strings, δηλαδή ακολουθίες χαρακτήρων που είναι αποθηκευμένες με encoding όπως το ASCII ή το UTF-8, μπορούν να αποκαλύψουν πολλές πληροφορίες σχετικά με τον σκοπό και τη λειτουργικότητα ενός αρχείου. Για παράδειγμα, μπορούν να εμφανίζουν URLs, file paths, error messages ή ακόμα και ύποπτες εντολές.

Ως πρώτο βήμα στη στατική ανάλυση, η ανάλυση strings παρέχει γρήγορα χρήσιμες πληροφορίες χωρίς να χρειάζεται να εκτελεστεί το κακόβουλο λογισμικό. Το εργαλείο strings του UNIX είναι ένα χαρακτηριστικό παράδειγμα εφαρμογής γραμμής εντολών που εξάγει εκτυπώσιμους χαρακτήρες από οποιοδήποτε αρχείο. Η εκτέλεση της εντολής strings binaryfile μπορεί να αποκαλύψει αναγνώσιμες συμβολοσειρές κειμένου σε αρχεία που διαφορετικά φαίνονται ακατανόητα.

Ωστόσο, το εργαλείο strings έχει περιορισμούς. Δεν προσφέρει μεγάλη προσαρμοστικότητα και, σε πολλές περιπτώσεις, απαιτούνται πιο προηγμένες μέθοδοι ανάλυσης. Παρόλα αυτά, είναι ένα εξαιρετικό σημείο εκκίνησης. Ένα χαρακτηριστικό παράδειγμα είναι η εκτέλεση της εντολής strings malware.bin | grep "http", η οποία μπορεί να αποκαλύψει URLs που πιθανώς οδηγούν σε command-and-control servers, παρέχοντας κρίσιμες πληροφορίες για την ανάλυση ενός κακόβουλου λογισμικού.

Η ανάλυση των strings έχει εφαρμογές και πέρα από τον τομέα της ασφάλειας. Μπορεί να χρησιμοποιηθεί στον τομέα της forensic analysis για την ανίχνευση πληροφοριών από κατεστραμμένα αρχεία, στο reverse engineering για την κατανόηση της εσωτερικής λειτουργίας ενός προγράμματος, καθώς και στο debugging για τον εντοπισμό σφαλμάτων και την επίλυση προβλημάτων.

2.5.2 Ανάλυση της Εντροπίας

Τι είναι η Εντροπία;

Η έννοια της εντροπίας, όπως εισήχθη από τον Claude Shannon, αναφέρεται στο μέτρο της αβεβαιότητας ή του τυχαίου χαρακτήρα μιας πληροφορίας. Στα πλαίσια των δυαδικών αρχείων, η εντροπία χρησιμοποιείται για να αξιολογήσει την τυχαιότητα ή την προβλεψιμότητα των δεδομένων που περιέχονται σε αυτά. Ένα αρχείο με υψηλή εντροπία παρουσιάζει μεγάλη ποικιλία byte τιμών, υποδηλώνοντας ότι τα δεδομένα είναι περισσότερο τυχαία ή συμπιεσμένα, ενώ ένα αρχείο με χαμηλή εντροπία παρουσιάζει επαναλαμβανόμενα μοτίβα, υποδηλώνοντας λιγότερη τυχαιότητα.

Η ανίχνευση περιοχών με υψηλή εντροπία σε δυαδικά αρχεία είναι σημαντική, καθώς μπορεί να υποδηλώνει την παρουσία συμπιεσμένων ή κρυπτογραφημένων τμημάτων. Οι τεχνικές συμπίεσης και κρυπτογράφησης αυξάνουν την εντροπία των δεδομένων, καθιστώντας τα πιο τυχαία και δυσκολότερα στην ανάλυση. Στην ανάλυση κακόβουλου λογισμικού, οι περιοχές με υψηλή εντροπία μπορεί να κρύβουν κακόβουλο κώδικα που έχει κρυπτογραφηθεί ή συμπιεστεί για να αποφύγει την ανίχνευση. Μια μελέτη με τίτλο "Wavelet decomposition of software entropy reveals symptoms of malicious code" αναφέρει ότι οι αλλαγές στην εντροπία ενός εκτελέσιμου αρχείου μπορούν να αποκαλύψουν κρυμμένο κακόβουλο κώδικα, ειδικά όταν αυτός είναι κρυπτογραφημένος ή συμπιεσμένος.

Υπολογισμός Εντροπίας

Ο υπολογισμός της εντροπίας σε ένα δυαδικό αρχείο περιλαμβάνει τη μέτρηση της κατανομής των byte τιμών σε ολόκληρο το αρχείο ή σε συγκεκριμένα τμήματά του. Η εντροπία υπολογίζεται χρησιμοποιώντας τον τύπο της εντροπίας του Shannon:

H(X)=−∑i=1np(xi)log⁡2p(xi)H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)H(X)=−∑i=1np(xi)log2p(xi)

Εξίσωση 1

όπου p(xi)p(x_i)p(xi) είναι η πιθανότητα εμφάνισης της τιμής byte xix_ixi στο σύνολο των δεδομένων. Η εντροπία μετριέται σε bits ανά byte και κυμαίνεται από 0 (όταν το αρχείο περιέχει μόνο μία τιμή byte) έως 8 (όταν όλες οι δυνατές τιμές byte εμφανίζονται με ίση πιθανότητα).

Η ερμηνεία των αποτελεσμάτων της ανάλυσης εντροπίας απαιτεί προσοχή. Υψηλή εντροπία μπορεί να υποδηλώνει συμπιεσμένα ή κρυπτογραφημένα δεδομένα, αλλά δεν αποτελεί από μόνη της απόδειξη κακόβουλης δραστηριότητας. Αντίθετα, περιοχές με χαμηλή εντροπία μπορεί να περιέχουν μη κρυπτογραφημένα ή μη συμπιεσμένα δεδομένα, όπως κείμενο ή σταθερές τιμές. Σύμφωνα με την πατέντα "System and method for determining data entropy to identify malware", η ανάλυση της εντροπίας μπορεί να βοηθήσει στον εντοπισμό κακόβουλου λογισμικού, ειδικά όταν συνδυάζεται με άλλες τεχνικές ανάλυσης .

Συνοψίζοντας, η ανάλυση εντροπίας είναι ένα ισχυρό εργαλείο για την κατανόηση της δομής και του περιεχομένου των δυαδικών αρχείων. Μέσω του υπολογισμού και της ερμηνείας της εντροπίας, οι αναλυτές μπορούν να εντοπίσουν περιοχές ενδιαφέροντος, όπως συμπιεσμένα ή κρυπτογραφημένα τμήματα, και να εστιάσουν περαιτέρω την ανάλυσή τους σε πιθανά κακόβουλα ή ύποπτα δεδομένα.

2.6 Μέθοδοι που θα χρησιμοποιηθούν στην παρούσα υλοποίηση.

Πρωταρχική επιλογή, της παρούσας πτυχιακής, αποτελεί η Στατική Ανάλυση, η οποία εφαρμόζεται μέσω του αυτοματοποιημένου script για την εξέταση των αρχείω απομονωμένων από την εκτέλεση. Ειδικότερα η μέθοδος String Analysis, όπως αναφέρθηκε στο κεφάλαιο 2.5.1, ενσωματώνεται στην υλοποίηση ως το βασικό εργαλείο εξαγωγής μεταδεδομένων. Μέσω αυτής, το script αυτοματοποιεί τον εντοπισμό κρήσιμων πληροφοριών οι οποίες υποδεικνύουν τις λειτπουργικές δυνατότητες του λογισμικού. Επιπροσθέτως, η εφαρμογή υπολογισμού Εντροπίας σε sections επιτρέπει στο script να αναγνωρίζει τμήματα τα οποία φέρουν ενδείξεις συμπίεσης ή κτρυπτογράφησης. Αυτή η μέθοδος αποκαλύπτει αν το δυαδικό αρχείο έχει υποστεί αλλοιώσεις με σκοπό την την αποφυγή της στατικής ανίχνευσης. Έτσι, στο πλαίσιο της παρούσας πτυχιακής εργασίας και για την ανάπτυξη του αυτοματοποιημένου εργαλείου που παρουσιάζεται στη συνέχεια, επιλέχθηκαν συγκεκριμένες μεθοδολογίες από τις προαναφερθείσες, οι οποίες κρίθηκαν ως οι πλέον κατάλληλες για την υλοποίηση μιας στατικής ανάλυσης υψηλής αξιοπιστίας.

ΚΕΦΑΛΑΙΟ 3 Τεχνολογίες και Εργαλεία Στατικής Ανάλυσης

3.1 Το εργαλείο 'strings':

Εξαγωγή Κειμένου για Ψηφιακή Εγκληματολογία
Υπάρχει ένα εργαλείο γραμμής εντολών που ονομάζεται 'strings' και βοηθά στην εξαγωγή σειρών κειμένου από οποιοδήποτε αρχείο. Το 'strings' είναι ένα από τα πιο σημαντικά εργαλεία που πρέπει να γνωρίζει κάθε επαγγελματίας ψηφιακής εγκληματολογίας.

Τι είναι οι ASCII και UNICODE συμβολοσειρές;

Βλέπουμε κάθε αρχείο και εφαρμογή στον υπολογιστή μας σε μια γλώσσα της επιλογής μας – Αγγλικά, Ισπανικά ή οποιαδήποτε άλλη γλώσσα. Ωστόσο, όλα αυτά τα αρχεία και οι εφαρμογές είναι κατανοητά από τον υπολογιστή μόνο σε δυαδική μορφή. Για να γεφυρωθεί η διαφορά μεταξύ της γλώσσας που είναι κατανοητή από τον άνθρωπο και της γλώσσας του υπολογιστή, επιχειρούμε να ερμηνεύσουμε μια δυαδική τιμή ως δεκαεξαδική. Για παράδειγμα, η δυαδική τιμή 01001101 σε δεκαεξαδικό είναι 4D. Κάθε τέσσερα bits σε μια δυαδική τιμή αντιπροσωπεύονται από έναν αριθμό σε δεκαεξαδικό. Έτσι, το 0100 σε δυαδική μορφή αντιστοιχεί στο 4 σε δεκαεξαδικό και το 1101 στο D.

Πώς ο υπολογιστής γνωρίζει ότι το 01001101 αντιστοιχεί στο 4D;

Αυτό οφείλεται σε έναν προκαθορισμένο χάρτη μεταξύ δυαδικών και δεκαεξαδικών τιμών για συγκεκριμένες τιμές. Συνήθως, αυτό περιλαμβάνει τα κεφαλαία και πεζά γράμματα του αγγλικού αλφαβήτου, τους αριθμούς 0-9, και ειδικούς χαρακτήρες όπως $, &, *. Το ASCII και το Unicode είναι κάποια σύνολα χαρακτήρων που ορίζουν αυτές τις αντιστοιχίσεις.

Το Unicode περιγράφει την αντιστοίχιση για τα αλφάβητα στις περισσότερες γλώσσες του κόσμου (Αγγλικά, Ελληνικά, Λατινικά), μαθηματικά σύμβολα, emoji, κ.λπ. Το ASCII περιγράφει την αντιστοίχιση μόνο για τα κεφαλαία και πεζά γράμματα της αγγλικής γλώσσας, τους αριθμούς 0-9, και ειδικούς χαρακτήρες.

Τι κάνει το εργαλείο 'strings';

Δεδομένου ενός αρχείου (αρχείο κειμένου, εικόνας ή δυαδικό), το 'strings' μπορεί να εξάγει ASCII και Unicode συμβολοσειρές από αυτό. Το εκτελέσιμο binary του 'strings' είναι διαθέσιμο για λειτουργικά συστήματα Windows, Linux και Mac. Από προεπιλογή, το εργαλείο εξάγει μόνο τις ASCII συμβολοσειρές, χρησιμοποιώντας την εντολή:
strings <file-to-extract-strings-from>

Για να εξαγάγετε επίσης τις Unicode συμβολοσειρές από ένα αρχείο, πρέπει να χρησιμοποιηθούν ειδικές επιλογές στη γραμμή εντολών. Στα Linux, η επιλογή -el χρησιμοποιείται για την εμφάνιση των Unicode συμβολοσειρών:
strings -el <file-to-extract-strings-from>

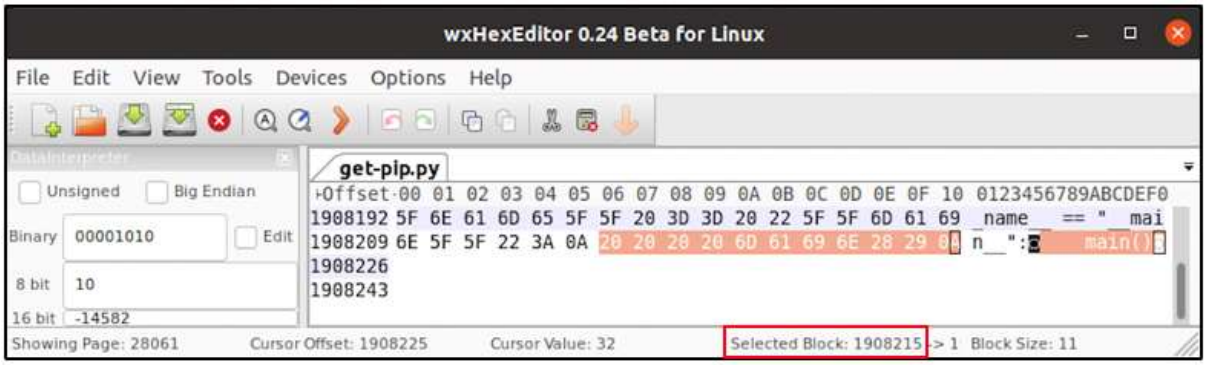
Σε ορισμένες περιπτώσεις, μπορεί να είναι σημαντικό να βρείτε την ακριβή θέση μιας συγκεκριμένης συμβολοσειράς μέσα σε ένα αρχείο. Στα Linux, η επιλογή -td μπορεί να χρησιμοποιηθεί για να εμφανιστεί η μετατόπιση στην οποία βρέθηκε μια συμβολοσειρά:
strings -td <file-to-extract-strings-from>

Παράδειγμα εξόδου: Όταν η εντολή εκτελέστηκε σε ένα αρχείο get-pip.py, βρέθηκε ότι η συμβολοσειρά _main() (με κάποια κενά πριν από αυτή) εμφανίστηκε στη θέση 1908215. Μερικές συμβολοσειρές που αναγνωρίζονται θα είναι κατανοητές, ενώ άλλες όχι.

```
1907925 LB!YYiwa+Wo&aUaCuNm0Rj{Q6aWAK2mlUeH&SOfaHqKi005&I001EX0000000000005+c>1Y-JaA|Na
1908005 Uv_0~WN&gWcV%K_Zewp`X>Mn8FLY&dbaO6nc~DCM0u%!j0000801jt2Qrian)};gh0Mifv04e|g0000
1908085 0000000HlH8ZWaJ=X>c!Jc4cm4Z*nhpWnyJ+V{c?>ZfA2ZcwcpMWpZC+WoBt^Wn?aJc~DCQ1^@s60D1
1908165 v>0cUmq0M2t30000
1908188 if __name__ == "__main__":
1908215     main()
```

Εικόνα 6. Αποτέλεσμα εφαρμογής του εργαλείου Strings.

Όταν το αρχείο ανοίχθηκε στον hex editor, η συμβολοσειρά βρέθηκε στη μετατόπιση (offset) 1908215. Το παρακάτω στιγμιότυπο το απεικονίζει.



Εικόνα 7. Αποτέλεσμα εφαρμογής του εργαλείου Strings 2.

3.2 Το εργαλείο 'Radare2':

Το Radare2 (γνωστό και ως r2) είναι ένα πλήρες πλαίσιο για reverse engineering και ανάλυση δυαδικών αρχείων· αποτελείται από μια σειρά μικρών βοηθητικών εργαλείων που μπορούν να χρησιμοποιηθούν είτε μαζί είτε ανεξάρτητα από τη γραμμή εντολών. Είναι κατασκευασμένο γύρω από έναν disassembler για λογισμικό υπολογιστών, ο οποίος δημιουργεί πηγαίο κώδικα assembly από εκτελέσιμο μηχανικό κώδικα, και υποστηρίζει μια ποικιλία εκτελέσιμων μορφών για διαφορετικές αρχιτεκτονικές επεξεργαστών και λειτουργικά συστήματα.

Ιστορία

Το Radare2 δημιουργήθηκε τον Φεβρουάριο του 2006, με στόχο την παροχή μιας δωρεάν και απλής γραμμής εντολών για έναν δεκαεξαδικό επεξεργαστή που να υποστηρίζει αντισταθμίσεις 64-bit για αναζητήσεις και ανάκτηση δεδομένων από σκληρούς δίσκους για σκοπούς εγκληματολογικής έρευνας. Έκτοτε, το έργο αναπτύχθηκε με αλλαγμένο στόχο την παροχή ενός ολοκληρωμένου πλαισίου για ανάλυση δυαδικών αρχείων, ενώ ακολουθεί πολλές από τις αρχές της φιλοσοφίας Unix.

Το 2009 αποφασίστηκε να ξαναγραφεί εξ ολοκλήρου, ώστε να ξεπεραστούν οι περιορισμοί του αρχικού σχεδιασμού. Από τότε, το έργο συνέχισε να αναπτύσσεται, προσελκύοντας πολλούς μόνιμους προγραμματιστές. Το 2016, πραγματοποιήθηκε το πρώτο συνέδριο r2con στη

Βαρκελώνη, με περισσότερους από 100 συμμετέχοντες και παρουσιάσεις για διάφορα χαρακτηριστικά και βελτιώσεις του πλαισίου. Το Radare2 έχει παρουσιαστεί σε πολλές σημαντικές παρουσιάσεις ασφαλείας, όπως recon, hack.lu, και 33c3.

Χαρακτηριστικά και Χρήση

Το Radare2 έχει απότομη καμπύλη εκμάθησης, καθώς τα κύρια εκτελέσιμα αρχεία του λειτουργούν μέσω γραμμής εντολών και δεν διαθέτει γραφικό περιβάλλον χρήστη (GUI) από μόνο του. Αρχικά κατασκευασμένο γύρω από έναν δεκαεξαδικό επεξεργαστή, τώρα διαθέτει πολλαπλά εργαλεία και χαρακτηριστικά, καθώς και συνδέσεις για πολλές γλώσσες προγραμματισμού. Ωστόσο, έχει πλέον ένα WebUI, ενώ το επίσημο GUI του Radare2 είναι το Iaito.

Στατική Ανάλυση

Το Radare2 μπορεί να συναρμολογήσει και να αποσυναρμολογήσει πολλά προγράμματα λογισμικού, κυρίως εκτελέσιμα, και να εκτελέσει binary diffing με γραφήματα, να εξάγει πληροφορίες, όπως σύμβολα μετατοπίσεων, και διάφορους άλλους τύπους δεδομένων. Χρησιμοποιεί μια NoSQL βάση δεδομένων με όνομα sdb για την καταγραφή πληροφοριών ανάλυσης που μπορεί να εξαχθούν από το Radare2 ή να προστεθούν χειροκίνητα από τον χρήστη.

Δυναμική Ανάλυση

Το Radare2 διαθέτει ενσωματωμένο debugger, χαμηλότερου επιπέδου από το GDB. Μπορεί επίσης να συνδεθεί με το GDB και το WineDBG για αποσφαλμάτωση Windows binaries σε άλλα συστήματα, και να χρησιμοποιηθεί ως kernel debugger με το VMWare.

Εκμετάλλευση Λογισμικού

Το Radare2 διαθέτει δυνατότητες, όπως μηχανή αναζήτησης ROP gadgets και ανίχνευση περιορισμών (mitigation detection), που το καθιστούν χρήσιμο για τους δημιουργούς exploits. Επιπλέον, υποστηρίζει πολλές μορφές αρχείων, γεγονός που το καθιστά δημοφιλές μεταξύ των ομάδων CTF και άλλων επαγγελματιών ασφαλείας.

GUI

Το έργο Iaito αναπτύχθηκε ως το πρώτο αφιερωμένο GUI για το Radare2. Στη συνέχεια, εξελίχθηκε σε Cutter, το οποίο αποτέλεσε δεύτερο GUI για το Radare2. Όταν το Cutter διαχωρίστηκε από το έργο Radare2 στα τέλη του 2020, το Iaito επανασχεδιάστηκε και παραμένει το επίσημο GUI του Radare2, συντηρούμενο από τα μέλη του έργου Radare2.

3.3 Το εργαλείο ‘Objdump’:

Όταν χειριζόμαστε αρχεία αντικειμένων (object files) χωρίς πρόσβαση στον πηγαίο κώδικα, είναι κρίσιμο να εξάγουμε όσο το δυνατόν περισσότερες πληροφορίες από αυτά τα αρχεία, ειδικά για σκοπούς αποσφαλμάτωσης, αντίστροφης μηχανικής ή ανάλυσης συστημάτων. Η εντολή objdump στο Linux διαδραματίζει σημαντικό ρόλο σε αυτά τα σενάρια, παρέχοντας ένα ισχυρό σύνολο εργαλείων για την εξέταση του περιεχομένου των αρχείων αντικειμένων. Παρακάτω θα μάθουμε για τη σύνταξη και τα παραδείγματα της εντολής objdump, προσφέροντας μια ευέλικτη λύση γραμμής εντολών για προγραμματιστές, αναλυτές ασφάλειας και διαχειριστές συστημάτων.

Τι είναι η εντολή objdump;

Η εντολή objdump στο Linux είναι ένα εργαλείο που σας επιτρέπει να εμφανίζετε πληροφορίες σχετικά με αρχεία αντικειμένων. Χρησιμοποιείται συνήθως για αποσφαλμάτωση και αντίστροφη μηχανική, προσφέροντας πληροφορίες για τη δομή και το περιεχόμενο των μεταγλωττισμένων αρχείων. Η objdump μπορεί να διαχειριστεί μια ευρεία γκάμα εκτελέσιμων μορφών, όπως τα αρχεία ELF (Executable and Linkable Format), και παρέχει επιλογές για αποσυναρμολόγηση κώδικα, επιθεώρηση κεφαλίδων και πολλά άλλα.

Συνήθεις χρήσεις της εντολής objdump:

Χρησιμοποιείται για τους εξής σκοπούς:

- Ανάκτηση κεφαλίδας αρχείου
- Λήψη μετατόπισης (offset) του αρχείου
- Εξαγωγή του bfdname
- Ανάλυση ονομάτων με το demangle
- Αποσφαλμάτωση αρχείου
- Αποσυναρμολόγηση αρχείου
- Ανάκτηση κεφαλίδων αρχείων

Σύνταξη:

bash

CopyEdit

objdump <option(s)> <file(s)>

3.4 Το εργαλείο 'Checksec':

Το Checksec είναι ένα shell script που μπορεί να χρησιμοποιηθεί για να ελέγξει τις ιδιότητες των δυαδικών αρχείων στο Linux. Αυτό μπορεί να χρησιμοποιηθεί για να ελέγξει διάφορες τεχνικές μετριασμού όπως το PIE, RELRO, NoExecute, Stack Canaries, ASLR και άλλες. Το Checksec μπορεί να ελέγξει τον πυρήνα του Linux για να δει αν ορισμένα χαρακτηριστικά ασφαλείας είναι ενεργοποιημένα. Όταν μεταγλωττίζετε τον πηγαίο κώδικα σε δυαδικό, υπάρχουν αρκετές δυνατότητες ασφαλείας που μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν κατά τη διάρκεια της μεταγλώττισης, ωστόσο, οι περισσότερες από αυτές τις δυνατότητες δεν είναι ενεργοποιημένες από προεπιλογή κατά τη μεταγλώττιση εφαρμογών και πρέπει να ρυθμιστούν κατά τη διάρκεια της μεταγλώττισης. Τα καλά νέα είναι ότι πολλά από τα κοινά upstream έργα έχουν υιοθετήσει αυτές τις ρυθμίσεις με την πάροδο του χρόνου για να γίνουν οι προεπιλεγμένες για τις εφαρμογές τους.

Ιστορία

Το 2009 ο Tobias Klein κυκλοφόρησε την πρώτη έκδοση του Checksec.sh, την οποία συντήρησε μέχρι τα τέλη του 2011. Κατά τη διάρκεια αυτής της περιόδου άρχισα να χρησιμοποιώ το checksec και συνέβαλα μερικές διορθώσεις που ενσωματώθηκαν στην τελευταία έκδοση που διατηρούσε ο Tobias. (v1.5). Περίπου 2 χρόνια αργότερα, ο Robin David αντέγραψε το checksec στο Github και έκανε κάποιες μικρές ενημερώσεις κατά την αρχική δέσμευση. Τον Φεβρουάριο του 2014, έκανα fork το έργο και άρχισα να κάνω κάποιες ενημερώσεις στο έργο. Από τότε είμαι ο συντηρητής του έργου με συνεισφορές και χαρακτηριστικά που προστέθηκαν από πολλούς ανθρώπους τα χρόνια.

Τεχνολογίες

Το εργαλείο αυτό διαθέτει ποικίλες τεχνολογίες όπως η τεχνολογία **No-Execute** η οποία χρησιμοποιείται σε επεξεργαστές για να χωρίζει περιοχές μνήμης σε 2 τμήματα: αποθήκευση κώδικα και αποθήκευση δεδομένων. Αυτό διαχωρίζει το τι πρέπει να εκτελείται ως κώδικας προγράμματος και τι πρέπει να διαβάζεται ως δεδομένα προγράμματος. Τα δεδομένα του προγράμματος αλλάζουν για να αφαιρεθούν τα εκτελέσιμα δικαιώματα από εκείνο το τμήμα της μνήμης, ώστε κανένα δεδομένο χρήστη να μην μπορεί να εκτελεστεί ως κώδικας εφαρμογής. Αυτή η τεχνολογία είναι ενσωματωμένη στον ίδιο τον επεξεργαστή και στα περισσότερα κοινά συστήματα είναι ενεργοποιημένη από προεπιλογή. Αυτό μπορεί να φανεί στις παραπάνω εικόνες. Μπορούμε να ξαναμεταγλωττίσουμε κάθε πρόγραμμα για να απενεργοποιήσουμε το bit NX. Πρέπει να σημειωθεί ότι αυτό απενεργοποιείται μόνο για σκοπούς επίδειξης και κανονικά δεν θα θέλαμε να το απενεργοποιήσουμε.

Η Τυχαία Διάταξη Χώρου Διευθύνσεων(Address Space Layout Randomization) είναι μια τεχνική ασφαλείας που χρησιμοποιείται για να βοηθήσει στην πρόληψη ή τη μείωση των πιθανοτήτων εκμετάλλευσης εφαρμογών. Η τυχαία διάταξη χώρου διευθύνσεων αποτρέπει έναν επιτιθέμενο από το να πηδήξει αξιόπιστα σε μια συγκεκριμένη τοποθεσία ή λειτουργία στη μνήμη. Η ASLR τυχαία διατάσσει τον χώρο διευθύνσεων σε διάφορες περιοχές της διαδικασίας. Ενώ υπάρχουν εκμεταλλεύσεις που μπορούν να παρακάμψουν τις προστασίες που παρέχονται από το ASLR, έχει αποδειχθεί αποτελεσματικό στη μείωση των πιθανοτήτων πολλών κοινών εκμεταλλεύσεων και περιλαμβάνεται σε Linux, Windows, macOS, BSD και άλλα. Είναι επίσης ενεργοποιημένο από προεπιλογή σε iOS και Android. Η ομάδα PaX δημοσίευσε τον πρώτο σχεδιασμό του ASLR το 2001, καθώς και τον σχεδιασμό για το noexec.
Για τα μεμονωμένα δυαδικά αρχεία, δεν υπάρχουν σημαίες που πρέπει να ενεργοποιηθούν, αλλά μπορούμε να ελέγξουμε τον πυρήνα για να δούμε αν το ASLR είναι ενεργοποιημένο. Στην παρακάτω εικόνα, μπορούμε να δούμε ότι το ASLR του Vanilla Kernel είναι ρυθμισμένο σε "Πλήρες", οπότε μπορούμε να επιβεβαιώσουμε ότι το ASLR είναι ενεργοποιημένο.
Αν δεν είναι ενεργοποιημένο, μπορεί να είναι με το «sysctl».
0 = Το ASLR είναι απενεργοποιημένο
1 = Τυχαία κατανομή ASLR της στοίβας, της σελίδας εικονικού δυναμικού κοινόχρηστου αντικειμένου (VDSO) και των περιοχών κοινής μνήμης
2 = Τυχαία κατανομή ASLR της στοίβας, της σελίδας VDSO, των περιοχών κοινής μνήμης και των τμημάτων δεδομένων

Συνοψίζοντας, τα χαρακτηριστικά ασφαλείας των δυαδικών αρχείων που προστίθενται κατά τη διάρκεια της μεταγλώττισης συχνά παραβλέπονται από τους περισσότερους, αλλά μπορούν να αποτελέσουν κρίσιμο μέρος της πολυεπίπεδης άμυνας και της βελτίωσης της συνολικής στάσης ασφαλείας των εφαρμογών σας. Ενώ υπάρχουν ορισμένες επιφυλάξεις ανάλογα με το πού και πώς χρησιμοποιούνται τα δυαδικά αρχεία, αν μπορείτε να διαχειριστείτε τις επιπλέον απαιτήσεις, τότε οι τελικές εικόνες θα παρέχουν επιπλέον προστασίες γύρω από κοινές επιθέσεις.

3.5 Το εργαλείο ‘Ghidra’:

Το Ghidra είναι ένα εργαλείο αντίστροφης μηχανικής που αναπτύχθηκε από την NSA και κυκλοφόρησε το 2019. Αυτό έχει αποδειχθεί ιδιαίτερα δημοφιλές στους αναλυτές κακόβουλου λογισμικού, καθώς είναι αυτό που ονομάζεται εργαλείο αποσυναρμολόγησης. Αυτό επιτρέπει σε έναν αναλυτή κακόβουλου λογισμικού να εξετάσει τη λειτουργικότητα ενός δείγματος κακόβουλου λογισμικού χωρίς να το εκτελέσει, κάτι που είναι εξαιρετικά χρήσιμο καθώς ο αναλυτής μπορεί να εξετάσει τον κώδικα του κακόβουλου λογισμικού και να χαρτογραφήσει τι ακριβώς κάνει.

Εικόνα 8. Το εργαλείο GHIDRA.

Ένα εργαλείο αποσυναρμολόγησης όπως το Ghidra δεν εκτελεί τον κώδικα, αλλά χαρτογραφεί τον κωδικό συναρμολόγησης του κακόβουλου λογισμικού και επιτρέπει στον χρήστη να προχωρά και να επιστρέφει στον κώδικα χωρίς να επηρεάζει το σύστημα αρχείων της συσκευής ανάλυσης. Αυτό καθιστά το Ghidra ένα ιδανικό εργαλείο για την αναγνώριση και την χαρτογράφηση λειτουργιών που μπορεί να είναι περαιτέρω ενδιαφέρουσες για έναν αναλυτή κακόβουλου λογισμικού.

Ιστορία

Η ύπαρξη του Ghidra αποκαλύφθηκε αρχικά στο κοινό μέσω του Vault 7 τον Μάρτιο του 2017, αλλά το λογισμικό παρέμεινε μη διαθέσιμο μέχρι την αποχαρτογράφησή του και την επίσημη κυκλοφορία του δύο χρόνια αργότερα. Ορισμένα σχόλια στον πηγαίο κώδικα υποδεικνύουν ότι υπήρχε ήδη από το 1999. Τον Ιούνιο του 2019, το coreboot άρχισε να χρησιμοποιεί το Ghidra για τις προσπάθειές του στην αντίστροφη μηχανική σε προβλήματα που αφορούν συγκεκριμένο υλικολογισμικό, μετά την ανοιχτή έκδοση του λογισμικού Ghidra.

Η Ghidra μπορεί να χρησιμοποιηθεί, επίσημα, ως αποσφαλματωτής από την έκδοση 10.0 του Ghidra. Ο αποσφαλματωτής του Ghidra υποστηρίζει την αποσφαλμάτωση προγραμμάτων Windows σε λειτουργία χρήστη μέσω του WinDbg, και προγραμμάτων Linux μέσω του GDB.

3.6 Το εργαλείο ‘IDA PRO’:

Το IDA Pro είναι ένα εργαλείο ανάλυσης δυαδικού κώδικα. Είναι ικανό να δημιουργεί χάρτες εκτέλεσης λογισμικού για να δείξει τις δυαδικές εντολές που εκτελούνται πραγματικά από τον επεξεργαστή σε μια συμβολική αναπαράσταση που ονομάζεται γλώσσα συναρμολόγησης. Αυτή η διαδικασία αποσυναρμολόγησης επιτρέπει στους ειδικούς λογισμικού να αναλύουν προγράμματα που υποψιάζονται ότι είναι κακόβουλα από τη φύση τους, όπως το spyware ή το malware. Ωστόσο, η γλώσσα συναρμολόγησης είναι δύσκολο να διαβαστεί και να κατανοηθεί. Γι' αυτόν τον λόγο, έχουν εφαρμοστεί προηγμένες τεχνικές στο IDA Pro για να κάνουν αυτόν τον πολύπλοκο κώδικα πιο ευανάγνωστο. Σε ορισμένες περιπτώσεις, είναι δυνατόν να επαναφέρουμε το δυαδικό πρόγραμμα σε ένα αρκετά κοντινό επίπεδο στον αρχικό κώδικα που το παρήγαγε. Ο χάρτης του κώδικα του προγράμματος μπορεί στη συνέχεια να υποβληθεί σε επεξεργασία για περαιτέρω έρευνα.

Η Hex-Rays αναπτύσσει και υποστηρίζει τον αποσυναρμολογητή IDA Pro. Αυτό το διάσημο εργαλείο ανάλυσης λογισμικού, το οποίο είναι de-facto πρότυπο στη βιομηχανία ασφάλειας λογισμικού, είναι ένα αναπόσπαστο στοιχείο στο εργαλείο ενός αναλυτή λογισμικού, ειδικού ασφαλείας, προγραμματιστή λογισμικού ή μηχανικού λογισμικού. Ο αποσυναρμολογητής και αποσφαλματωτής IDA Pro είναι ένας διαδραστικός, προγραμματιζόμενος, επεκτάσιμος αποσυναρμολογητής πολλαπλών επεξεργαστών που φιλοξενείται σε Windows, Linux ή Mac OS X. Το IDA Pro είναι το τέλειο εργαλείο για την ανάλυση εχθρικού κώδικα, την έρευνα ευπαθειών και την επικύρωση εμπορικών έτοιμων λύσεων.

Η εφαρμογή IDA Pro καλύπτει την έρευνα ευπαθειών, την ανάλυση κακόβουλου λογισμικού, τη δυναμική ανάλυση, την ψηφιακή εγκληματολογία, τη δοκιμή διείσδυσης, την πνευματική ιδιοκτησία, την διαλειτουργικότητα και την αξιολόγηση λογισμικού. Παρέχει ανάλυση ενσωματωμένου λογισμικού αυτοκινήτων, βελ tuning αυτοκινήτων, έρευνα ασφαλείας και λογισμικό κληρονομιάς. Η χειροκίνητη ανάλυση ή το ακατέργαστο υλικολογισμικό επίσης διαχειρίζονται, μαζί με την αποσφαλμάτωση ενσωματωμένου λογισμικού, διαδικτυακών εφαρμογών και εκπαίδευσης.

3.7 Το εργαλείο 'Binwalk':

Το Binwalk είναι ένα ισχυρό εργαλείο ανοιχτού κώδικα που χρησιμοποιείται για την ανάλυση και εξαγωγή δεδομένων από δυαδικά αρχεία, συγκεκριμένα από εικόνες firmware. Κατασκευασμένο για επαγγελματίες ασφάλειας πληροφοριών και μηχανικούς αντίστροφης μηχανικής, το Binwalk εντοπίζει ενσωματωμένα αρχεία, συμπιέσεις και πιθανή κρυπτογράφηση μέσα σε δυαδικά δεδομένα. Παρακάτω περιγράφονται οι βασικές λειτουργίες του Binwalk και η σημασία τους στην ανάλυση δυαδικών αρχείων.

Η **σάρωση υπογραφών** επιτρέπει στο Binwalk να αναγνωρίζει αυτόματα ενσωματωμένα αρχεία, κεφαλίδες και γνωστούς τύπους δεδομένων, συγκρίνοντάς τα με μια βάση δεδομένων υπογραφών αρχείων. Η βάση δεδομένων μαγικών υπογραφών του Binwalk (/etc/binwalk/magic) περιέχει ορισμούς υπογραφών αρχείων. Το εργαλείο σαρώνει δυαδικά αρχεία για ταιριαστά μοτίβα και μετατοπίσεις, προκειμένου να εντοπίσει ενσωματωμένα δεδομένα και να προσδιορίσει τη δομή του αρχείου.

Η **ανάλυση εντροπίας** βοηθά στην ανίχνευση κρυπτογραφημένων ή συμπιεσμένων περιοχών σε δυαδικά αρχεία μετρώντας την τυχαιότητα στα δεδομένα. Τα συμπιεσμένα και κρυπτογραφημένα δεδομένα εμφανίζουν υψηλή εντροπία, ενώ τα απλά ή δομημένα δεδομένα (π.χ., κείμενο ASCII) εμφανίζουν χαμηλότερη εντροπία. Αυτό επιτρέπει στους αναλυτές να εντοπίζουν περιοχές που μπορεί να περιέχουν σημαντικές πληροφορίες ή πιθανό κακόβουλο κώδικα.

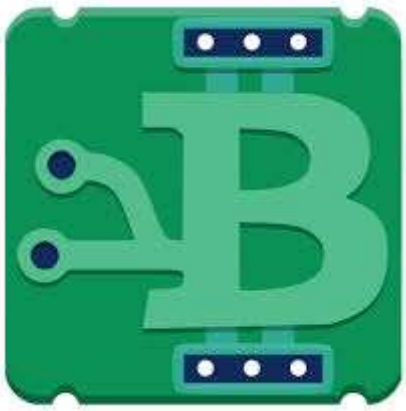
Η **εξαγωγή αρχείων** επιτρέπει στο Binwalk να εξάγει ενσωματωμένα αρχεία αυτόματα ή αναδρομικά χρησιμοποιώντας εξαγωγείς που ορίζονται στη διαμόρφωσή του. Οι προεπιλεγμένοι κανόνες εξαγωγής (extract.conf) καθορίζουν πώς να χειρίζεται συγκεκριμένους τύπους αρχείων (π.χ., gzip, tar). Η αναδρομική εξαγωγή εφαρμόζει αυτούς τους κανόνες στα εξαγόμενα δεδομένα, επιτρέποντας την εξαγωγή αρχείων μέσα σε άλλα αρχεία.

Η **ανάλυση δομής λογισμικού συσκευής** επιτρέπει την ανάλυση της δομής του υλικολογισμικού, συμπεριλαμβανομένων των συστημάτων αρχείων, των μεθόδων συμπίεσης και των αρχιτεκτονικών. Το Binwalk εντοπίζει τις μετατοπίσεις του συστήματος αρχείων, τις κεφαλίδες συμπίεσης και τις αρχιτεκτονικές λεπτομέρειες, βοηθώντας τους αναλυτές αντίστροφης μηχανικής να κατανοήσουν το υλικολογισμικό και τις λειτουργίες του.

Η **προσαρμοσμένη σάρωση με υπογραφές** επιτρέπει στους χρήστες να ορίσουν προσαρμοσμένους κανόνες για την ανίχνευση ιδιόκτητων ή άγνωστων τύπων αρχείων, προσθέτοντάς τους στη βάση δεδομένων υπογραφών του Binwalk. Οι προσαρμοσμένες υπογραφές ορίζονται στο /etc/binwalk/magic χρησιμοποιώντας τη σύνταξη του εργαλείου file. Κάθε εγγραφή καθορίζει ένα μοτίβο, μια μετατόπιση και μια περιγραφή των δεδομένων, διευκολύνοντας την αναγνώριση ειδικών τύπων αρχείων.

Η **οπτικοποίηση** επιτρέπει τη δημιουργία γραφημάτων εντροπίας για να οπτικοποιηθούν τα μοτίβα δεδομένων στο δυαδικό αρχείο. Τα γραφήματα εντροπίας είναι χρήσιμα για την ανίχνευση συμπιεσμένων ή κρυπτογραφημένων περιοχών, καθώς οι υψηλές τιμές εντροπίας υποδεικνύουν συχνά την παρουσία κρυπτογραφημένων ή συμπιεσμένων δεδομένων. Παρόλο που το Binwalk δεν υποστηρίζει πλέον άμεσα τη γραφική απεικόνιση, τα δεδομένα εντροπίας μπορούν να εξαχθούν και να αναλυθούν μέσω εξωτερικών εργαλείων, όπως το **Gnuplot**.

Το Binwalk, συνδυάζοντας τις παραπάνω λειτουργίες, προσφέρει ένα ισχυρό πλαίσιο για την ανάλυση binary αρχείων, διευκολύνοντας τη διαδικασία αναγνώρισης δομών, ενσωματωμένων αρχείων και πιθανών ευπαθειών.



Εικόνα 9. Το εργαλείο Binwalk.

3.8 Το εργαλείο 'Flawfinder':

Το Flawfinder είναι ένα απλό πρόγραμμα που σαρώνει τον πηγαίο κώδικα C/C++ και αναφέρει πιθανά προβλήματα ασφαλείας. Μπορεί να είναι ένα χρήσιμο εργαλείο για την εξέταση λογισμικού για ευπάθειες, και μπορεί επίσης να χρησιμεύσει ως μια απλή εισαγωγή στα εργαλεία στατικής ανάλυσης πηγαίου κώδικα γενικότερα. Είναι σχεδιασμένο να είναι εύκολο στην εγκατάσταση και τη χρήση. Ο Flawfinder υποστηρίζει την Καταμέτρηση Κοινών Αδυναμιών (CWE) και είναι επίσημα Συμβατός με CWE.

Το Flawfinder ,από μόνο του, παρέχει πολλά πλεονεκτήματα. Καθορίζει το επίπεδο κινδύνου καθώς παρέχει μια λίστα με πιθανές ευπάθειες ασφαλείας που ταξινομούνται κατά επίπεδο κινδύνου. Οι συναρτήσεις και οι παράμετροι που χρησιμοποιούνται στον κώδικα καθορίζουν το επίπεδο κινδύνου. Για παράδειγμα, οι σταθερές τιμές συμβολοσειρών είναι λιγότερο επικίνδυνες σε σύγκριση με τις μεταβλητές συμβολοσειρές. Σε ορισμένες περιπτώσεις, το Flawfinder μπορεί να είναι σε θέση να καθορίσει ότι η κατασκευή δεν είναι καθόλου επικίνδυνη, μειώνοντας τα ψευδώς θετικά αποτελέσματα. Επίσης, παρέχει περίληψη ανάλυσης ως έξοδο και αναφέρει τον αριθμό των ευπαθειών που βρέθηκαν στον κώδικα. Τέλος, ο πηγαίος κώδικας δεν μεταγλωττίζεται ποτέ και επομένως, ακόμη και αν ο κώδικας δεν λειτουργεί, το εργαλείο θα εμφανίσει τη λίστα των ευπαθειών σχεδόν αμέσως.

Πέρα από τα πλεονεκτήματα που μπορεί να προσφέρει, το εργαλείο αυτό συνοδεύεται από κάποια μειονεκτήματα. Το Flawfinder δεν εγγυάται ότι θα βρει όλες τις ευπάθειες. Κάθε χτύπημα που παράγεται δεν υποδηλώνει μια ευπάθεια ασφαλείας και ούτε κάθε ευπάθεια εντοπίζεται. Για παράδειγμα, σε ένα απλό πρόγραμμα διαίρεσης, ένας αριθμός διαιρείται με το 0. Ιδανικά, το εργαλείο θα έπρεπε να δείχνει τη διαίρεση με το 0 ως σφάλμα, αλλά αποτυγχάνει να το κάνει. Αυτό συμβαίνει επειδή το εργαλείο δεν μπορεί να κατανοήσει τη λογική του προγράμματος. Δεν μπορεί

να ανιχνεύσει κακόβουλους κώδικες. Το Flawfinder αναζητά συγκεκριμένα μοτίβα που είναι γνωστά ως κοινά λάθη στον κώδικα εφαρμογών. Έτσι, είναι πιθανό να είναι λιγότερο αποτελεσματικό στην ανάλυση προγραμμάτων που μπορεί να περιέχουν κακόβουλους κώδικες.

3.9 Το εργαλείο 'pwntools':

Η Pwntools είναι μια ισχυρή και ευέλικτη βιβλιοθήκη Python που έχει σχεδιαστεί για να διευκολύνει τη δημιουργία exploits και την ανάλυση binary αρχείων, ειδικά σε σενάρια CTF (Capture The Flag) και ηθικού hacking. Χρησιμοποιείται κυρίως για την εκμετάλλευση ευπαθειών σε binary αρχεία, παρέχοντας ένα σύνολο εργαλείων και λειτουργιών που αυτοματοποιούν και απλοποιούν τη διαδικασία ανάπτυξης exploits.

Η βιβλιοθήκη θεωρείται βασικό εργαλείο για την ανάλυση και εκμετάλλευση binary αρχείων, καθώς προσφέρει ένα ολοκληρωμένο σύνολο εργαλείων για την ανάπτυξη exploits και την παραβίαση μηχανισμών ασφαλείας. Συγκεκριμένα, επιτρέπει τη δημιουργία και επικοινωνία με διεργασίες (processes) είτε τοπικά είτε απομακρυσμένα μέσω socket ή pipes, διευκολύνοντας την ανάλυση ή εκτέλεση επιθέσεων. Μπορεί να δημιουργήσει αυτόματα payloads και ακολουθίες Return-Oriented Programming (ROP) χρησιμοποιώντας ενσωματωμένους μηχανισμούς εύρεσης gadgets σε ένα binary αρχείο. Παρέχει προκατασκευασμένα shellcodes ή επιτρέπει την κατασκευή προσαρμοσμένων shellcodes για χρήση σε επιθέσεις εκμετάλλευσης. Προσφέρει αυτόματη διαχείριση διαφορετικών αρχιτεκτονικών (x86, x86_64, ARM) και μορφών endianness (little-endian, big-endian), διευκολύνοντας τη δημιουργία εκμεταλλεύσεων σε διαφορετικά περιβάλλοντα. Χρησιμοποιεί το Capstone για την αποσυναρμολόγηση (disassembly) κώδικα, επιτρέποντας την εύκολη κατανόηση της λειτουργίας ενός binary. Μπορεί να εντοπίσει τμήματα μνήμης, όπως το stack και το heap, καθώς και να αναλύσει την κατάσταση της μνήμης κατά τη διάρκεια μιας επίθεσης. Τελειώνοντας, μπορεί να παρακάμψει ή να εκμεταλλευτεί μηχανισμούς προστασίας όπως το Address Space Layout Randomization (ASLR) και το Non-Executable (NX) stack.

3.10 Το εργαλείο 'Angr':

Το angr είναι μια πλατφόρμα ανάλυσης δυαδικών αρχείων ανοιχτού κώδικα για Python. Συνδυάζει τόσο στατική όσο και δυναμική συμβολική ("συγκολική") ανάλυση, παρέχοντας εργαλεία για την επίλυση ποικιλίας εργασιών.

Το **angr** είναι μια ισχυρή πλατφόρμα ανάλυσης δυαδικών αρχείων, προσφέροντας δυνατότητες όπως ανάκτηση γραφήματος ροής ελέγχου, συμβολική εκτέλεση, αυτόματη δημιουργία αλυσίδας ROP με το **angrop**, αυτόματο hardening δυαδικών αρχείων με το **patcherex** και αυτόματη δημιουργία εκμεταλλεύσεων για δυαδικά αρχεία **DECREE** και Linux με το **rex**. Διαθέτει το **angr-management**, ένα GUI για ανάλυση δυαδικών αρχείων. Επιπλέον, το **Mechanical Phish**, ο τρίτος νικητής του **DARPA Cyber Grand Challenge**, αξιοποιεί το angr για κυβερνοαυτονομία. Το angr αποτελείται από διάφορα υποέργα, όπως το **CLE** (φορτωτής εκτελέσιμων και βιβλιοθηκών), το **archinfo** (περιγραφή αρχιτεκτονικών), το **PyVEX** (Python wrapper γύρω από τον ανυψωτή VEX), το **Claripy** (backend δεδομένων για στατικούς και συμβολικούς τομείς) και τη σουίτα ανάλυσης προγράμματος **angr**.

Εικόνα 10. Το framework angr.

Στο πεδίο της ανάλυσης δυαδικών αρχείων έχουν αναπτυχθεί διάφορα εργαλεία που επιτρέπουν την εξερεύνηση της δομής και της λειτουργίας ενός εκτελέσιμου αρχείου. Στον ακόλουθο πίνακα παρουσιάζονται συνοπτικά τα σημαντικότερα εργαλεία στατικής ανάλυσης που αναφέρονται στο παρόν κεφάλαιο, καθώς και οι δυνατότητες και περιορισμοί τους σε σχέση με την υλοποίηση του προτεινόμενου συστήματος.

Εργαλείο	Χρησιμότητα του Εργαλείου	Τι δεν κάνει σε σχέση με το script
strings	Εξάγει αναγνώσιμες συμβολοσειρές από δυαδικά αρχεία, όπως URLs, ονόματα αρχείων ή πιθανά passwords. Χρησιμοποιείται συχνά σε αρχική ανάλυση για τον εντοπισμό χρήσιμων πληροφοριών μέσα σε ένα binary.	Αποτελεί μερικό υποσύνολο της λειτουργίας του script μας. Το δικό μας script το καλεί αυτοματοποιημένα και ενσωματώνει τα αποτελέσματά του σε μια ευρύτερη ανάλυση, χωρίς να απαιτείται χειροκίνητη εκτέλεση.
objdump	Εργαλείο του GNU binutils που επιτρέπει την εξέταση της εσωτερικής δομής ενός εκτελέσιμου αρχείου. Μπορεί να εμφανίσει sections, symbol tables και αποσυναρμολογημένο κώδικα, βοηθώντας τον αναλυτή να κατανοήσει τη δομή του binary.	Αποτελεί βασικό δομικό στοιχείο που το script μας ενσωματώνει. Αντί ο χρήστης να θυμάται περίπλοκες παραμέτρους της objdump, το script εκτελεί την κατάλληλη εντολή και εξάγει τα σημαντικά δεδομένα (π.χ., sections headers) αυτόματα.
radare2	Framework reverse engineering που παρέχει δυνατότητες αποσυναρμολόγησης, debugging και ανάλυσης συναρτήσεων ενός binary. Μπορεί να εντοπίσει functions, imports και τη δομή ενός εκτελέσιμου αρχείου.	Στο script χρησιμοποιείται για να παρέχει μια στοχευμένη, αυτοματοποιημένη διεπαφή για συγκεκριμένες πληροφορίες.

Ghidra	Προηγμένο εργαλείο reverse engineering που παρέχει αποσυναρμολόγηση και decompilation δυαδικών αρχείων σε ψευδοκώδικα. Διαθέτει γραφικό περιβάλλον και ισχυρές δυνατότητες ανάλυσης προγραμμάτων.	Είναι ένα βαρύ, διαδραστικό GUI εργαλείο που απαιτεί ανθρώπινη παρέμβαση. Το script μας είναι πλήρως αυτοματοποιημένο, γραμμής εντολών, και στοχεύει στην ταχεία εξαγωγή μιας πρώτης έκθεσης, όχι σε εις βάθος, διαδραστική ανάλυση..
IDA Pro	Επαγγελματικό εργαλείο reverse engineering που χρησιμοποιείται ευρέως στην ανάλυση malware. Παρέχει αποσυναρμολόγηση, ανάλυση ροής προγράμματος και δυνατότητες debugging.	Ομοίως με το Ghidra, είναι ένα εμπορικό, διαδραστικό εργαλείο για εξειδικευμένη ανάλυση. Το δικό μας script είναι ανοιχτού κώδικα, αυτοματοποιημένο και προσανατολισμένο στην ταχεία αξιολόγηση.
Binwalk	Εργαλείο που χρησιμοποιείται κυρίως για ανάλυση firmware και embedded συστημάτων. Μπορεί να εντοπίσει ενσωματωμένα αρχεία, συμπιεσμένα δεδομένα και συστήματα αρχείων μέσα σε ένα binary.	Η ανάλυση εντροπίας του Binwalk είναι μια προηγμένη λειτουργία που το script μας υλοποιεί με απλούστερο τρόπο. Το Binwalk υπερέχει στον εντοπισμό και εξαγωγή ενσωματωμένων αρχείων, κάτι που δεν αποτελεί πρωταρχικό στόχο του δικού μας script.
checksec	Εργαλείο που ελέγχει αν ένα binary χρησιμοποιεί μηχανισμούς προστασίας όπως NX, PIE, RELRO και Stack Canary. Βοηθά στην αξιολόγηση της ασφάλειας ενός εκτελέσιμου αρχείου.	Στο script ο αντίστοιχος έλεγχος γίνεται μέσω της βιβλιοθήκης pwntools και όχι με απευθείας χρήση του checksec.
flawfinder	Εργαλείο στατικής ανάλυσης που εντοπίζει πιθανά προβλήματα ασφαλείας σε πηγαίο κώδικα C/C++. Ελέγχει τη χρήση επικίνδυνων συναρτήσεων που μπορεί να οδηγήσουν σε ευπάθειες όπως buffer overflow.	Λειτουργεί σε πηγαίο κώδικα, όχι σε δυαδικά αρχεία. Είναι θεμελιωδώς διαφορετικό ως προς το αντικείμενο ανάλυσης, αν και μοιράζεται τον κοινό στόχο της αυτοματοποιημένης εύρεσης αδυναμιών.
angr	Πλατφόρμα ανάλυσης δυαδικών που χρησιμοποιεί τεχνικές symbolic execution για τον εντοπισμό ευπαθειών και τη διερεύνηση πιθανών execution paths ενός προγράμματος.	Είναι ένα ερευνητικό, υψηλής πολυπλοκότητας framework. Το script μας είναι ελαφρύ, πρακτικό και άμεσο, εστιάζοντας στην εξαγωγή μετρήσιμων χαρακτηριστικών και metadata, όχι στη συμβολική εκτέλεση ή στην αυτόματη δημιουργία exploits. Καλύπτει διαφορετικό στάδιο της ανάλυσης.
pwntools	Βιβλιοθήκη Python που χρησιμοποιείται για exploitation και ανάλυση ELF binaries. Στο script αξιοποιείται για τον έλεγχο μηχανισμών ασφαλείας όπως NX, PIE, Stack Canary και RELRO.	Δεν χρησιμοποιείται για ανάπτυξη exploits ή dynamic exploitation, αλλά μόνο για εξαγωγή πληροφοριών ασφαλείας του binary.

Πίνακας 1. Οι Λειτουργίες των Εργαλείων.

ΚΕΦΑΛΑΙΟ 4 Ευπάθειες και Απειλές

4.1 Τύποι Ευπαθειών στα Binary Αρχεία

Η ανάλυση binary αρχείων δεν περιορίζεται μόνο στην εξαγωγή πληροφοριών και την κατανόηση της λειτουργίας τους, αλλά αποτελεί και ένα κρίσιμο εργαλείο για τον εντοπισμό πιθανών ευπαθειών που μπορεί να εκμεταλλευτούν επιτιθέμενοι. Οι ευπάθειες αυτές, που προκύπτουν από σφάλματα στον κώδικα ή εσφαλμένες διαμορφώσεις, μπορούν να οδηγήσουν σε σοβαρούς κινδύνους ασφαλείας, όπως μη εξουσιοδοτημένη πρόσβαση, διαρροή ευαίσθητων δεδομένων και εκτέλεση κακόβουλου κώδικα.

4.1.1 Buffer Overflow – Η υπερχείλιση buffer είναι μια ευπάθεια όπου τα δεδομένα εισόδου υπερβαίνουν τον καθορισμένο χώρο αποθήκευσης, διαφθείροντας ή ελέγχοντας τη ροή εκτέλεσης του προγράμματος. Παρέχοντας κατάλληλα διαμορφωμένα δεδομένα εισόδου, μπορεί να αλλάξει η επόμενη εντολή εκτέλεσης του προγράμματος, αντικαθιστώντας τη διεύθυνση επιστροφής της τρέχουσας συνάρτησης. Οι υπερχειλίσεις buffer μπορούν να αξιοποιηθούν από επιτιθέμενους για την εκτέλεση αυθαίρετου κώδικα, την απόκτηση μη εξουσιοδοτημένης πρόσβασης ή την πρόκληση άλλης κακόβουλης συμπεριφοράς. Οι υπερχειλίσεις buffer μπορούν να οδηγήσουν σε διάφορους κινδύνους ασφαλείας που αν δεν αντιμετωπιστούν ενδεχομένως να υπάρξουν σοβαρές συνέπειες.

Η εκτέλεση αυθαίρετου κώδικα αποτελεί μία από τις πιο σοβαρές συνέπειες, καθώς επιτρέπει σε έναν επιτιθέμενο να εκτελέσει αυθαίρετο κώδικα με τα δικαιώματα της επηρεαζόμενης διεργασίας. Αυτό μπορεί να οδηγήσει σε πλήρη παραβίαση του συστήματος, συμπεριλαμβανομένης της κλοπής ευαίσθητων πληροφοριών. Για την ανάλυση και τον εντοπισμό τέτοιων ευπαθειών, ένα κατάλληλο εργαλείο είναι το **Ghidra**. Το Ghidra είναι ένα εργαλείο reverse engineering που παρέχει δυνατότητες disassembly, ανάλυσης ροής ελέγχου control flow analysis και αποκάλυψης της δομής της μνήμης. Μέσω του Ghidra, μπορεί να εντοπιστεί η διαδρομή εκτέλεσης του προγράμματος, να αναλυθούν οι εντολές χαμηλού επιπέδου και να εντοπιστούν σημεία όπου η ροή εκτέλεσης μπορεί να παραβιαστεί από κακόβουλη εισαγωγή δεδομένων.

Μία υπερχείλιση buffer μπορεί επίσης να οδηγήσει σε επίθεση Denial of Service (DoS), καθώς μπορεί να προκαλέσει την κατάρρευση ή την αδράνεια ενός προγράμματος, οδηγώντας σε προσωρινή ή μόνιμη διακοπή υπηρεσίας. Ένα κατάλληλο εργαλείο για τον εντοπισμό και την αποτροπή τέτοιων ευπαθειών είναι το **checksec**. Το checksec επιτρέπει τον έλεγχο των μηχανισμών προστασίας του εκτελέσιμου αρχείου, όπως ASLR (Address Space Layout Randomization), DEP (Data Execution Prevention) και Stack Canaries. Αν το checksec αποκαλύψει ότι λείπει κάποιος από αυτούς τους μηχανισμούς προστασίας, αυτό σημαίνει ότι το πρόγραμμα είναι ευάλωτο σε εκμετάλλευση μέσω buffer overflow και ενδέχεται να καταρρεύσει εάν δεχτεί επίθεση.

Η διαρροή πληροφοριών μέσω buffer overflow μπορεί να επιτρέψει σε έναν επιτιθέμενο να αποκτήσει πρόσβαση σε ευαίσθητες πληροφορίες, όπως κωδικούς πρόσβασης, κλειδιά κρυπτογράφησης ή εμπιστευτικά δεδομένα που βρίσκονται στη μνήμη της επηρεαζόμενης διεργασίας. Το **strings** είναι ένα χρήσιμο εργαλείο σε αυτήν την περίπτωση, καθώς επιτρέπει την εξαγωγή και εμφάνιση όλων των ευανάγνωστων συμβολοσειρών από ένα εκτελέσιμο αρχείο ή ένα τμήμα μνήμης. Αν σε ένα πρόγραμμα υπάρχουν συμβολοσειρές που σχετίζονται με ευαίσθητα δεδομένα και είναι αποθηκευμένες στη μνήμη χωρίς κρυπτογράφηση ή προστασία, το strings μπορεί να τις αποκαλύψει, επιτρέποντας στον επιτιθέμενο να εκμεταλλευτεί αυτήν την αδυναμία.

Τέλος, η αύξηση δικαιωμάτων πρόσβασης μέσω buffer overflow μπορεί να επιτρέψει σε έναν επιτιθέμενο να αποκτήσει αυξημένα δικαιώματα στο σύστημα, παρακάμπτοντας τους ελέγχους ασφαλείας και αποκτώντας πρόσβαση σε ευαίσθητα συστήματα και δεδομένα. Το εργαλείο **pwntools** μπορεί να χρησιμοποιηθεί για την αυτοματοποίηση της διαδικασίας εκμετάλλευσης buffer overflow. Το pwntools παρέχει έτοιμες συναρτήσεις και εργαλεία για τη δημιουργία payloads, τη σύνδεση με απομακρυσμένες υπηρεσίες, τη διαμόρφωση του heap και του stack, καθώς και τη δοκιμή exploit σε περιβάλλον προσομοίωσης. Αυτό καθιστά δυνατή την ανάπτυξη και τη δοκιμή επιθέσεων με ελεγχόμενο και δομημένο τρόπο, επιτρέποντας στους ερευνητές ασφαλείας να εντοπίσουν και να διορθώσουν τέτοιες ευπάθειες πριν αξιοποιηθούν από κακόβουλους επιτιθέμενους.

4.1.2 Library Exploitation – Εκμετάλλευση Κακόβουλων Βιβλιοθηκών

Η εκμετάλλευση κακόβουλων βιβλιοθηκών (malicious library exploitation) αποτελεί μια από τις πιο ανησυχητικές επιθέσεις στο οικοσύστημα ανάπτυξης λογισμικού. Αυτή η τεχνική περιλαμβάνει την εισαγωγή κακόβουλου κώδικα μέσα σε βιβλιοθήκες ή εξαρτήσεις που χρησιμοποιούνται από προγραμματιστές για την ανάπτυξη εφαρμογών. Συχνά, οι βιβλιοθήκες αυτές είναι ανοιχτού κώδικα και, εάν χρησιμοποιηθούν από ανυποψίαστους προγραμματιστές, μπορεί να επιτρέψουν σε έναν επιτιθέμενο να αποκτήσει μη εξουσιοδοτημένη πρόσβαση, να κλέψει δεδομένα ή να προκαλέσει σοβαρές ζημιές στο σύστημα.

Πώς Γίνεται η Εκμετάλλευση Κακόβουλων Βιβλιοθηκών

Η εκμετάλλευση των βιβλιοθηκών μπορεί να συμβεί με διάφορους τρόπους. Ένας από τους πιο διαδεδομένους είναι η δημιουργία κακόβουλων βιβλιοθηκών. Οι επιτιθέμενοι μπορούν να αντικαταστήσουν δημοφιλή πακέτα με κακόβουλες εκδόσεις τους, εφαρμόζοντας την τεχνική typosquatting. Για παράδειγμα, μια βιβλιοθήκη με όνομα "request" αντί για "requests" μπορεί να περιέχει κακόβουλο κώδικα, παγιδεύοντας προγραμματιστές που πληκτρολογούν λανθασμένα το όνομα του πακέτου. Επιπλέον, οι επιτιθέμενοι μπορούν να δημιουργήσουν νέες βιβλιοθήκες που περιέχουν κακόβουλο κώδικα, τις οποίες δημοσιεύουν σε αποθετήρια ανοιχτού κώδικα, ελπίζοντας ότι θα χρησιμοποιηθούν από ανυποψίαστους χρήστες.

Μια άλλη μέθοδος εκμετάλλευσης αφορά την παραβίαση υπαρχόντων πακέτων. Σε αυτή την περίπτωση, οι επιτιθέμενοι αποκτούν πρόσβαση σε λογαριασμούς συντηρητών λογισμικού και εισάγουν κακόβουλο κώδικα σε δημοφιλείς βιβλιοθήκες. Αυτή η προσέγγιση είναι ιδιαίτερα επικίνδυνη, καθώς οι χρήστες εμπιστεύονται ήδη τις συγκεκριμένες βιβλιοθήκες. Επιπλέον, οι επιτιθέμενοι μπορούν να εισάγουν κρυφό κακόβουλο κώδικα σε βιβλιοθήκες που ενημερώνονται συχνά, καθιστώντας την ανίχνευσή τους δύσκολη.

Ένας ακόμη τρόπος εκμετάλλευσης αφορά την αλυσίδα εξαρτήσεων (dependency chain). Οι περισσότερες εφαρμογές βασίζονται σε βιβλιοθήκες που, με τη σειρά τους, εξαρτώνται από άλλες βιβλιοθήκες. Εάν κάποια από αυτές περιέχει κακόβουλο κώδικα, τότε μπορεί να επηρεάσει ολόκληρη την εφαρμογή, ακόμα και αν η αρχική βιβλιοθήκη που χρησιμοποίησε ο προγραμματιστής φαίνεται ασφαλής.

Κίνδυνοι από Κακόβουλες Βιβλιοθήκες

Η εκμετάλλευση κακόβουλων βιβλιοθηκών μπορεί να οδηγήσει σε σοβαρούς κινδύνους ασφαλείας. Ένας από τους μεγαλύτερους κινδύνους είναι η διαρροή δεδομένων. Κακόβουλες βιβλιοθήκες μπορεί να συλλέγουν και να αποστέλλουν ευαίσθητες πληροφορίες, όπως διαπιστευτήρια πρόσβασης, προσωπικά δεδομένα χρηστών ή εμπορικά μυστικά, σε επιτιθέμενους.

Επιπλέον, μέσω αυτών των βιβλιοθηκών, οι επιτιθέμενοι μπορούν να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε συστήματα. Αυτό μπορεί να συμβεί με την εγκατάσταση backdoors ή άλλων μηχανισμών που επιτρέπουν απομακρυσμένο έλεγχο.

Μια ακόμα απειλή είναι η εκτέλεση απομακρυσμένων εντολών. Ο κακόβουλος κώδικας που περιλαμβάνεται σε μια βιβλιοθήκη μπορεί να επιτρέψει σε έναν επιτιθέμενο να εκτελεί εντολές στο σύστημα του θύματος, οδηγώντας σε πλήρη παραβίαση του συστήματος.

Οι κακόβουλες βιβλιοθήκες μπορούν, επίσης, να προκαλέσουν αλλοίωση δεδομένων ή υπηρεσιών, εισάγοντας παραμορφωμένα δεδομένα ή καταστρέφοντας αρχεία και βάσεις δεδομένων.

Τέλος, οι βιβλιοθήκες αυτές μπορούν να χρησιμοποιηθούν για την κακόβουλη κλιμάκωση επιθέσεων. Εάν μια βιβλιοθήκη είναι ενσωματωμένη σε πολλές εφαρμογές, μπορεί να χρησιμοποιηθεί για τη μόλυνση επιπλέον συστημάτων, διευρύνοντας σημαντικά την έκταση της επίθεσης.

Για τον εντοπισμό και την ανάλυση κακόβουλων βιβλιοθηκών, ένα κατάλληλο εργαλείο είναι το **Flawfinder**. Το Flawfinder είναι ένα εργαλείο στατικής ανάλυσης κώδικα που αναζητά γνωστά μοτίβα αδυναμιών στον κώδικα, συμπεριλαμβανομένων προβλημάτων σε βιβλιοθήκες. Μπορεί να αναλύσει τον πηγαίο κώδικα των βιβλιοθηκών και να εντοπίσει σημεία όπου γίνεται εισαγωγή εξωτερικών εξαρτήσεων, υποδεικνύοντας ενδεχόμενες ευπάθειες.

Επιπλέον, το **Binwalk** μπορεί να χρησιμοποιηθεί για την ανάλυση αρχείων βιβλιοθηκών και πακέτων λογισμικού. Το Binwalk επιτρέπει την αποσυμπίεση και την ανάλυση δυαδικών αρχείων, βοηθώντας στον εντοπισμό κρυφού ή κακόβουλου κώδικα που μπορεί να είναι ενσωματωμένος σε βιβλιοθήκες. Αυτό καθιστά δυνατή την αποκάλυψη επικίνδυνων εξαρτήσεων και την εξουδετέρωση πιθανών απειλών πριν αυτές αξιοποιηθούν.

4.1.3 Format string – Η ευπάθεια της μορφοποίησης συμβολοσειράς (format string vulnerability), που συμβαίνει όταν τα δεδομένα εισόδου επεξεργάζονται εσφαλμένα, μπορεί να οδηγήσει σε μη εξουσιοδοτημένη πρόσβαση ή αλλοίωση της μνήμης. Με την αποστολή μίας ειδικά διαμορφωμένης συμβολοσειράς που περιέχει χαρακτήρες μορφοποίησης (format specifiers), η επίθεση εκμεταλλεύεται τη μνήμη του προγράμματος για να αλλάξει τη ροή εκτέλεσής του.

Η διαμορφωμένη είσοδος περιλαμβάνει χαρακτήρες σημαίας της μορφοποίησης συμβολοσειράς, όπως:

%d: Εμφάνιση ακέραιου αριθμού σε δεκαδική μορφή.

%s: Εμφάνιση συμβολοσειράς.

%x: Εμφάνιση αριθμού σε δεκαεξαδική μορφή.

%p: Εμφάνιση δείκτη μνήμης (address).

%n: Εγγραφή του αριθμού χαρακτήρων που έχουν εκτυπωθεί μέχρι εκείνο το σημείο σε μια μεταβλητή μνήμης.

Αυτή η ευπάθεια επιτρέπει σε έναν επιτιθέμενο να διαβάσει ευαίσθητα δεδομένα από τη μνήμη, να προκαλέσει κατάρρευση του προγράμματος ή ακόμη και να εκτελέσει αυθαίρετο κώδικα.

Το Pwntools είναι μια ισχυρή βιβλιοθήκη που χρησιμοποιείται ευρέως για την εκμετάλλευση δυαδικών αρχείων και την ανάπτυξη exploits. Στην περίπτωση της (format string vulnerability, το Pwntools επιτρέπει στον αναλυτή ασφαλείας να στείλει ειδικά διαμορφωμένες συμβολοσειρές σε ένα πρόγραμμα και να αναλύσει την έξοδο που επιστρέφεται. Μέσω των format specifiers, όπως %p, %x και %n, το εργαλείο μπορεί να αποκαλύψει κρίσιμες πληροφορίες για τη διάταξη της μνήμης (π.χ., διευθύνσεις στη στοίβα ή στη μνήμη heap) ή να τροποποιήσει τιμές στη μνήμη του προγράμματος. Επιπλέον, το Pwntools παρέχει λειτουργίες για την αυτοματοποίηση επιθέσεων, όπως η τροποποίηση της διεύθυνσης επιστροφής μιας συνάρτησης (write-what-where), καθιστώντας δυνατή την εκτέλεση αυθαίρετου κώδικα. Χρησιμοποιώντας τις δυνατότητες του Pwntools, ένας επιτιθέμενος μπορεί να διαρρεύσει δεδομένα, να παρακάμψει μέτρα ασφαλείας (π.χ., ASLR) και να αποκτήσει μη εξουσιοδοτημένη πρόσβαση στο σύστημα.

4.1.4 Integer underflow

– Η ευπάθεια του **integer underflow** εμφανίζεται όταν μια μαθηματική πράξη οδηγεί σε τιμή μικρότερη από την ελάχιστη αναπαραστάσιμη τιμή για έναν τύπο ακεραίου, προκαλώντας συχνά απρόβλεπτη συμπεριφορά ή προβλήματα ασφαλείας. Στο παράδειγμα μας, θα υποβάλουμε έναν αρνητικό αριθμό για να χειριστούμε τη λογική του προγράμματος, προκαλώντας μη αναμενόμενη αύξηση σε μια κρίσιμη μεταβλητή ελέγχου.

Για να κατανοήσουμε καλύτερα την ευπάθεια ας φανταστούμε ένα εύρος τιμών για έναν **int** είναι: **"-2,147,483,648 έως 2,147,483,647".** Η μέγιστη τιμή για έναν **unsigned int** είναι: **"4,294,967,295".** Εάν υποβληθεί η τιμή **"-1"** σε μια μεταβλητή τύπου **unsigned int**, η τιμή αναπαρίσταται ως **"4,294,967,295"**—η μέγιστη τιμή που μπορεί να πάρει ένας **unsigned int**. Αυτό μπορεί να χρησιμοποιηθεί από επιτιθέμενους για να εκμεταλλευτούν λογικές συνθήκες, να παρακάμψουν ελέγχους ή να δημιουργήσουν απροσδόκητα αποτελέσματα σε ένα πρόγραμμα.

Το **Angr** είναι ένα ισχυρό εργαλείο ανάλυσης δυαδικού κώδικα που μπορεί να χρησιμοποιηθεί για την ανίχνευση και εκμετάλλευση ευπαθειών integer underflow. Το Angr λειτουργεί μέσω symbolic execution, επιτρέποντας στον αναλυτή να εξετάσει όλες τις πιθανές διαδρομές εκτέλεσης ενός προγράμματος. Στην περίπτωση του integer underflow, το Angr μπορεί να ανιχνεύσει σφάλματα στα όρια τιμών και να εντοπίσει πώς η υποβολή αρνητικών ή υπερβολικά μεγάλων τιμών μπορεί να επηρεάσει τη λογική του προγράμματος. Για παράδειγμα, μπορεί να μοντελοποιήσει ένα σενάριο όπου μια αρνητική τιμή εισόδου σε unsigned int προκαλεί αύξηση σε μια κρίσιμη μεταβλητή, οδηγώντας σε παράκαμψη ελέγχων ασφαλείας ή αλλαγή της ροής εκτέλεσης. Με τις δυνατότητες του Angr για αυτοματοποιημένη δημιουργία exploits, οι αναλυτές μπορούν να δοκιμάσουν διαφορετικές καταστάσεις integer underflow και να εντοπίσουν πιθανές ευπάθειες προτού αυτές γίνουν στόχος εκμετάλλευσης.

4.1.5 Race condition – Η ευπάθεια **race condition** εμφανίζεται όταν ο χρονισμός ή η σειρά των γεγονότων σε ένα πολυνηματικό ή ασύγχρονο σύστημα μπορεί να χειραγωγηθεί από επιτιθέμενους για να παραβιάσει την ασφάλεια. Οι επιτιθέμενοι μπορούν να εκμεταλλευτούν τέτοιες ευπάθειες για να μεταβάλλουν τη συμπεριφορά ενός προγράμματος ή συστήματος με ανεπιθύμητους τρόπους.
Ο στόχος μιας επίθεσης σε ευπάθεια race condition είναι η χειραγώγηση του περιεχομένου ενός αρχείου ανάμεσα στις φάσεις επικύρωσης και εκτέλεσης. Ένα ευάλωτο πρόγραμμα ελέγχει το hash ενός dummy εκτελέσιμου αρχείου και το εκτελεί μόνο αν το hash ταιριάζει με μια προκαθορισμένη τιμή.

Η επίθεση λειτουργεί αντικαθιστώντας επανειλημμένα το dummy εκτελέσιμο με ένα κακόβουλο αρχείο, μετά τον έλεγχο του hash αλλά πριν από την εκτέλεση.

Η χρονική διαφορά μεταξύ του ελέγχου και της εκτέλεσης του αρχείου επιτρέπει την εισαγωγή κακόβουλου κώδικα. Η επιτυχής εκμετάλλευση επιτυγχάνεται όταν το πρόγραμμα εκτελέσει το αντικατασταθέν αρχείο, αποδεικνύοντας ότι ο έλεγχος ασφαλείας γίνεται άκυρος λόγω της χρονικής απόκλισης. Αυτό επιτρέπει την εκτέλεση μη εξουσιοδοτημένου κώδικα.

Το Radare2 μπορεί να χρησιμοποιηθεί για την ανάλυση και τον εντοπισμό ευπαθειών race condition σε δυαδικά αρχεία. Το Radare2 είναι ένα προηγμένο εργαλείο αντίστροφης μηχανικής (reverse engineering) που επιτρέπει στους αναλυτές ασφαλείας να αποδομήσουν τον κώδικα ενός προγράμματος, να εξετάσουν τη ροή εκτέλεσης και να εντοπίσουν πιθανές ευπάθειες χρονισμού. Στην περίπτωση ενός race condition, το Radare2 μπορεί να χρησιμοποιηθεί για τη χαρτογράφηση των system calls που σχετίζονται με τη διαχείριση αρχείων, όπως το άνοιγμα, ο έλεγχος και η εκτέλεση, αποκαλύπτοντας τη χρονική αλληλουχία των γεγονότων. Μέσω ανάλυσης της διαδρομής εκτέλεσης, το Radare2 μπορεί να εντοπίσει τη χρονική απόκλιση μεταξύ του ελέγχου και της εκτέλεσης ενός αρχείου, βοηθώντας στον εντοπισμό κρίσιμων σημείων όπου μπορεί να εισαχθεί κακόβουλος κώδικας. Επιπλέον, υποστηρίζει scripting και αυτοματοποίηση, επιτρέποντας στους αναλυτές να δημιουργούν δοκιμές για διαφορετικές συνθήκες εκμετάλλευσης race conditions.

4.1.6 Heap overflow – Η ευπάθεια **heap overflow** συμβαίνει όταν ένα πρόγραμμα γράφει περισσότερα δεδομένα σε μια μνήμη που έχει κατανεμηθεί δυναμικά στη heap απ' ό,τι μπορεί να χωρέσει. Αυτό μπορεί να οδηγήσει σε υπερχείλιση γειτονικών περιοχών μνήμης, προκαλώντας πιθανά καταρρεύσεις προγράμματος (crashes), διαφθορά δεδομένων (corruption), παραβιάσεις ασφαλείας (security breaches).

Οι επιτιθέμενοι μπορούν να εκμεταλλευτούν υπερχειλίσεις στη heap για να εκτελέσουν αυθαίρετο κώδικα, αλλοιώνοντας τη μνήμη και εισάγοντας κακόβουλο κώδικα, ή να τροποποιήσουν τη ροή ελέγχου του προγράμματος, αλλάζοντας pointers ή κρίσιμα δεδομένα ώστε να ελέγξουν τη λειτουργία του. Η heap είναι συχνά πιο δύσκολο να προστατευθεί από την stack λόγω της δυναμικής της φύσης, κάτι που καθιστά τις heap overflows επικίνδυνο εργαλείο στα χέρια επιτιθέμενων.

Το Ghidra είναι εξαιρετικά χρήσιμο για την ανάλυση heap overflow, καθώς επιτρέπει την αποδόμηση και αποσφαλμάτωση δυαδικών αρχείων, επιτρέποντας στον αναλυτή να χαρτογραφήσει τη διάταξη της heap και να εντοπίσει πιθανές αδυναμίες. Το Ghidra προσφέρει μια λεπτομερή οπτική αναπαράσταση της δομής μνήμης, επιτρέποντας στον αναλυτή να δει πώς

κατανεμήθηκε η heap, πώς χρησιμοποιούνται οι pointers και πώς γίνεται η διαχείριση της μνήμης. Αν ένα πρόγραμμα προσπαθεί να γράψει περισσότερα δεδομένα από όσα έχει κατανεμημένα σε μια περιοχή της heap, το Ghidra μπορεί να βοηθήσει στον εντοπισμό του σημείου εκτέλεσης όπου συμβαίνει το overflow. Με τις λειτουργίες decompilation και ανάλυσης ροής εκτέλεσης, το Ghidra επιτρέπει την αναγνώριση των μοτίβων μνήμης και των buffer που ενδέχεται να επηρεαστούν από overflow, διευκολύνοντας τη δημιουργία διορθώσεων και την πρόληψη παρόμοιων ευπαθειών στο μέλλον.

4.1.7 Use after free – Η ευπάθεια Use After Free (UAF) συμβαίνει όταν ένα πρόγραμμα προσπελάζει μνήμη αφού αυτή έχει ήδη αποδεσμευτεί, οδηγώντας σε απρόβλεπτη συμπεριφορά ή κατάρρευση.

> Αυτός ο τύπος ευπάθειας εξελίσσεται ως εξής:
> Πρώτα, το πρόγραμμα δεσμεύει μνήμη σε έναν δείκτη (pointer), η οποία χρησιμοποιείται για διάφορες λειτουργίες. Αργότερα, η μνήμη αποδεσμεύεται, αλλά ο δείκτης δεν εκκαθαρίζεται ή δεν ανατίθεται εκ νέου. Εξακολουθεί να δείχνει στη διεύθυνση της μνήμης που είχε αρχικά δεσμευτεί. Αυτός ο dangling pointer μπορεί στη συνέχεια να χρησιμοποιηθεί λανθασμένα.
> Όταν αυτή η αποδεσμευμένη μνήμη προσπελαστεί μέσω του παλιού δείκτη, το πρόγραμμα μπορεί να συμπεριφερθεί απρόβλεπτα, καθώς η μνήμη μπορεί πλέον να περιέχει διαφορετικά δεδομένα ή να χρησιμοποιείται από άλλο μέρος του προγράμματος.
>
> Εκμεταλλευόμενος αυτή την ευπάθεια, ένας επιτιθέμενος μπορεί να χειραγωγήσει τη ροή εκτέλεσης του προγράμματος και να εκτελέσει αυθαίρετο κώδικα. Αυτή η χειραγώγηση συμβαίνει επειδή ο επιτιθέμενος μπορεί να επηρεάσει τα δεδομένα που αποθηκεύονται στη θέση μνήμης που είχε προηγουμένως αποδεσμευτεί, αποκτώντας έλεγχο στη λειτουργία του προγράμματος.

Το **Angr** είναι ιδιαίτερα χρήσιμο στην ανάλυση και εκμετάλλευση ευπαθειών τύπου UAF. Το Angr είναι ένα framework για δυναμική και συμβολική εκτέλεση που επιτρέπει στον αναλυτή να χαρτογραφήσει τη μνήμη του προγράμματος και να εντοπίσει πότε ένα πρόγραμμα προσπελαύνει μνήμη που έχει ήδη αποδεσμευτεί. Μπορεί να χρησιμοποιηθεί για την αυτοματοποιημένη εύρεση διαδρομών εκτέλεσης όπου παρουσιάζονται dangling pointers και να εντοπίσει αν η κατάσταση αυτή μπορεί να οδηγήσει σε αυθαίρετη εκτέλεση κώδικα. Το Angr μπορεί επίσης να μοντελοποιήσει τη ροή δεδομένων και να αναλύσει ποιες συνθήκες πρέπει να ικανοποιούνται ώστε να εκδηλωθεί το UAF, βοηθώντας τους ερευνητές ασφαλείας να δημιουργήσουν exploits ή να σχεδιάσουν διορθώσεις για το πρόβλημα.

4.2 Συστήματα Ταξινόμησης Ευπαθειών:

Για την αποτελεσματική διαχείριση και αντιμετώπιση των ευπαθειών στα binary αρχεία, είναι απαραίτητο να υπάρχει ένα οργανωμένο σύστημα κατηγοριοποίησης. Τα συστήματα ταξινόμησης ευπαθειών παρέχουν ένα τυποποιημένο πλαίσιο που επιτρέπει την καταγραφή, αξιολόγηση και ανάλυση των αδυναμιών ασφαλείας. Μέσω αυτών των συστημάτων, οι ερευνητές ασφαλείας, οι προγραμματιστές και οι διαχειριστές συστημάτων μπορούν να κατανοούν καλύτερα τις απειλές και

να εφαρμόζουν κατάλληλα μέτρα προστασίας. Παρακάτω αναλύονται τρία από τα σημαντικότερα Συστήματα Ταξινόμησης Ευπαθειών.

4.2.1 About CWE

Common Weakness Enumeration (CWE™) είναι μια λίστα που αναπτύσσεται από την κοινότητα και περιλαμβάνει κοινές αδυναμίες λογισμικού και υλικού. Μια "αδυναμία" (weakness) είναι μια κατάσταση σε ένα λογισμικό, firmware, hardware ή στοιχείο υπηρεσίας που, υπό ορισμένες συνθήκες, μπορεί να συμβάλει στην εισαγωγή ευπαθειών.

Η λίστα CWE και η σχετική ταξινομική κατηγοριοποίηση εντοπίζουν και περιγράφουν αδυναμίες με όρους CWEs.

Η κατανόηση των αδυναμιών που οδηγούν σε ευπάθειες επιτρέπει στους προγραμματιστές λογισμικού, στους σχεδιαστές υλικού και στους αρχιτέκτονες ασφαλείας να τις εξαλείψουν πριν από την ανάπτυξη των συστημάτων, κάτι που είναι πολύ ευκολότερο και οικονομικότερο να επιτευχθεί.

Εικόνα 11. Ανάλυση εισαγωγής και ανίχνευσης μιας ευπάθειας.

CWE List

Η λίστα CWE ενημερώνεται τρεις έως τέσσερις φορές τον χρόνο για να προστεθούν νέες και να ανανεωθούν οι υπάρχουσες πληροφορίες για αδυναμίες. Πριν δημοσιευτούν στον ιστότοπο της CWE, οι αδυναμίες αναπτύσσονται στο CWE Content Development Repository (CDR) στο GitHub.com. Το CDR παρέχει ορατότητα στην ουρά εργασίας της CWE και αποτελεί πλατφόρμα συνεργασίας για τους συνεργάτες της κοινότητας CWE σχετικά με την ανάπτυξη περιεχομένου.

Χρήση της λίστας CWE

Η λίστα CWE είναι πλήρως διαθέσιμη για αναζήτηση και μπορεί να προβληθεί ή να ληφθεί εξ ολοκλήρου. Επίσης, υπάρχει το CWE REST API που καθιστά το περιεχόμενο της CWE διαθέσιμο σε εφαρμογές και ιστότοπους της κοινότητας με πιο βολικό τρόπο.

Οι αδυναμίες μπορούν να περιηγηθούν μέσα από “Views” που σχετίζονται με συγκεκριμένα περιβάλλοντα ή τομείς όπως για παράδειγμα το Software Development View που οργανώνει τις αδυναμίες με βάση τις έννοιες που χρησιμοποιούνται συχνά κατά την ανάπτυξη λογισμικού, το Hardware Design View το οποίο εστιάζει σε αδυναμίες σχετικές με τον σχεδιασμό υλικού και το Research Concepts με ρόλο να διευκολύνει την έρευνα τύπων αδυναμιών, οργανώνοντάς τις με βάση συμπεριφορές.

Άλλες προβολές παρέχουν πληροφορίες για συγκεκριμένους τομείς ή περιπτώσεις χρήσης, όπως αδυναμίες που εισάγονται κατά τον σχεδιασμό ή την υλοποίηση, έμμεσες επιπτώσεις στην ασφάλεια, αδυναμίες σε λογισμικό γραμμένο σε γλώσσες όπως C, C++, Java και PHP, καθώς και σε εφαρμογές για κινητά.

Μια ακόμα χρήσιμη δυνατότητα είναι η εξωτερική αντιστοίχιση του περιεχομένου CWE με σχετικούς πόρους, όπως η CWE Top 25 (ετήσια λίστα), OWASP Top Ten, Seven Pernicious Kingdoms, Software Fault Pattern Clusters και SEI CERT Coding Standards για C, Java, και Perl.

Όλες αυτές οι μοναδικές οπτικές στις πληροφορίες της CWE επιτρέπουν την άμεση αξιοποίησή της για τις δικές σας ανάγκες. Το περιεχόμενο της λίστας CWE διατίθεται δωρεάν για ενσωμάτωση σε έρευνες, εκπαιδευτικά υλικά, διαδικασίες και εργαλεία, σύμφωνα με τους όρους χρήσης.

4.2.2 About CVE

Το Common Vulnerabilities and Exposures (CVE) είναι ένα σύνολο απειλών ασφαλείας που περιλαμβάνονται σε ένα σύστημα αναφοράς που περιγράφει δημόσια γνωστούς κινδύνους. Η λίστα απειλών CVE διατηρείται από την MITRE Corporation, μια μη κερδοσκοπική οργάνωση που διαχειρίζεται ερευνητικά και αναπτυξιακά κέντρα που χρηματοδοτούνται από την ομοσπονδιακή κυβέρνηση. Η CVE υποστηρίζεται από το Υπουργείο Εσωτερικής Ασφάλειας των Η.Π.Α. μέσω της Εθνικής Διεύθυνσης Κυβερνοασφάλειας (NCSD). Μια ευπάθεια είναι ένα σφάλμα στον κώδικα λογισμικού που οι κυβερνοεπιτιθέμενοι μπορούν να χρησιμοποιήσουν για να αποκτήσουν πρόσβαση σε ένα υπολογιστικό σύστημα και να εκτελέσουν μη εξουσιοδοτημένες ενέργειες προσποιούμενοι ότι είναι νόμιμος χρήστης. Οι επιτιθέμενοι μπορεί να χρησιμοποιήσουν τις ευπάθειες για να εκτελέσουν κώδικα, να αποκτήσουν πρόσβαση στη μνήμη του συστήματος, να εγκαταστήσουν διάφορες μορφές κακόβουλου λογισμικού και να κλέψουν, διαγράψουν ή τροποποιήσουν ευαίσθητα δεδομένα. Μια έκθεση είναι διαφορετική. Είναι ένα λάθος στον κώδικα λογισμικού ή στη διαμόρφωση που δίνει σε έναν επιτιθέμενο πρόσβαση σε ένα σύστημα ή δίκτυο. Οι παραβιάσεις δεδομένων, οι διαρροές δεδομένων και η πώληση προσωπικά αναγνωρίσιμων πληροφοριών (PII) στο σκοτεινό διαδίκτυο μπορούν όλα να προκύψουν από εκθέσεις.

Πώς Λειτουργεί το Σύστημα CVE;

Η λίστα και το σύστημα CVE διατηρούνται από την MITRE Corporation. Παρέχει μια τυποποιημένη μέθοδο για την αναγνώριση γνωστών ευπαθειών και εκθέσεων ασφαλείας. Το CVE έχει σχεδιαστεί για να επιτρέπει τη σύγκριση εργαλείων και υπηρεσιών ασφαλείας και τη σύνδεση βάσεων δεδομένων ευπαθειών. Παρέχει τυπικούς αναγνωριστικούς αριθμούς που επιτρέπουν στους διαχειριστές ασφαλείας να έχουν γρήγορη πρόσβαση σε πληροφορίες σχετικά με συγκεκριμένες απειλές. Σημαντικά, οι καταχωρίσεις CVE περιέχουν μόνο τον τυπικό αριθμό αναγνωριστικού και τον δείκτη κατάστασης μιας ευπάθειας, καθώς και μια σύντομη περιγραφή και

σχετικές αναφορές σε συμβουλές και εκθέσεις. Αυτό σημαίνει ότι δεν περιλαμβάνουν λεπτομερείς τεχνικές πληροφορίες σχετικά με τον κίνδυνο, τις διορθώσεις ή τον αντίκτυπο της ευπάθειας. Αυτές οι λεπτομέρειες καταγράφονται σε βάσεις δεδομένων όπως η Εθνική Βάση Δεδομένων Ευπαθειών (NVD) και η Βάση Δεδομένων Σημειώσεων Ευπαθειών του Κέντρου Συντονισμού CERT (CERT/CC) (VND).

Εικόνα 12. Διασφάλιση συμβατότητας με το σύστημα CVE.

Πώς ορίζει το CVE τις ευπάθειες;

Η CVE ορίζει τις ευπάθειες ως ένα λάθος στον κώδικα λογισμικού, το οποίο επιτρέπει σε έναν επιτιθέμενο να αποκτήσει άμεση μη εξουσιοδοτημένη πρόσβαση σε υπολογιστικά συστήματα και δίκτυα και να διαδώσει κακόβουλο λογισμικό. Αυτό συνήθως επιτρέπει στους επιτιθέμενους να προσποιούνται τους διαχειριστές συστήματος ή υπερχρήστες με πλήρη δικαιώματα πρόσβασης στους εταιρικούς πόρους. Η CVE ορίζει την έκθεση ως σφάλματα στον κώδικα λογισμικού ή στη διαμόρφωση, τα οποία επιτρέπουν σε έναν επιτιθέμενο να αποκτήσει έμμεση πρόσβαση σε συστήματα και δίκτυα. Αυτό θα μπορούσε να επιτρέψει στον επιτιθέμενο να παραμονεύει σε δίκτυα υπολογιστών και να συλλέγει μυστικά ευαίσθητα δεδομένα, διαπιστευτήρια χρηστών και πληροφορίες πελατών.

Κορυφαίες 3 Βάσεις Δεδομένων CVE

Αυτές είναι οι τρεις κορυφαίες βάσεις δεδομένων για τις Κοινές Ευπάθειες και Εκθέσεις (CVE):

National Vulnerability Database (NVD): Η NVD προσφέρει μια ανάλυση ασφαλείας και μια

πιο λεπτομερή περιγραφή της ευπάθειας, σε αντίθεση με το πλαίσιο MITRE, το οποίο παρέχει μόνο το αναγνωριστικό της και μια σύντομη περιγραφή.

Vulnerability Assessment Platform (Vulners): Το Vulners είναι μια τακτικά ενημερωμένη βάση δεδομένων εκμεταλλεύσεων και ευπαθειών. Κάθε εγγραφή της βάσης δεδομένων περιλαμβάνει αναγνωριστικά, ορισμούς και πληροφορίες για τη σοβαρότητα. Η Vulners παρέχει έναν σαρωτή ευπαθειών, ένα πρόσθετο σαρωτή Nmap, μια επέκταση σαρωτή για τον περιηγητή και ένα εργαλείο αξιολόγησης ευπαθειών τεχνητής νοημοσύνης (AI).

Vulnerability Database (VulDB): Μια βάση δεδομένων ευπαθειών παρακολουθεί όλα τα αναφερόμενα προβλήματα ασφαλείας. Είναι μια δωρεάν υπηρεσία που χρησιμοποιείται από ερευνητές ασφαλείας για τη διαχείριση ευπαθειών, την ανάλυση απειλών και την ανταπόκριση σε περιστατικά.

4.2.3 About CAPEC

Η πρωτοβουλία **Common Attack Pattern Enumeration and Classification (CAPEC™)** παρέχει έναν δημόσια διαθέσιμο κατάλογο κοινών μοτίβων επιθέσεων, βοηθώντας τους χρήστες να κατανοήσουν πώς οι επιτιθέμενοι εκμεταλλεύονται αδυναμίες σε εφαρμογές και άλλες δυνατότητες που υποστηρίζονται από την κυβερνοτεχνολογία.

Τι είναι τα "Attack Patterns";

Τα "Attack Patterns" περιγράφουν τα κοινά χαρακτηριστικά και τις προσεγγίσεις που χρησιμοποιούν οι επιτιθέμενοι για να εκμεταλλευτούν γνωστές αδυναμίες, ορίζοντας τις προκλήσεις που αντιμετωπίζουν και πώς τις ξεπερνούν. Βασίζονται στην έννοια των design patterns, αλλά εφαρμόζονται σε καταστροφικό πλαίσιο, και προκύπτουν από εις βάθος ανάλυση πραγματικών παραδειγμάτων εκμετάλλευσης αδυναμιών.

Πώς λειτουργούν τα Attack Patterns

Κάθε μοτίβο επίθεσης συγκεντρώνει γνώση για το σχεδιασμό και την εκτέλεση μιας επίθεσης, παρέχει καθοδήγηση για τον μετριασμό της αποτελεσματικότητάς της και βοηθά τους δημιουργούς εφαρμογών και διαχειριστές να κατανοήσουν και να αποτρέψουν τα στοιχεία μιας επίθεσης.

Με τη χρήση του καταλόγου CAPEC™, οι οργανισμοί μπορούν να προετοιμαστούν καλύτερα ενάντια σε γνωστές στρατηγικές επιθέσεων και να βελτιώσουν την ασφάλεια των συστημάτων τους.

Some Well-Known Attack Patterns:

- HTTP Response Splitting (CAPEC-34)
- Session Fixation (CAPEC-61)
- Cross Site Request Forgery (CAPEC-62)
- SQL Injection (CAPEC-66)
- Cross-Site Scripting (CAPEC-63)
- Buffer Overflow (CAPEC-100)
- Clickjacking (CAPEC-103)
- Relative Path Traversal (CAPEC-139)
- XML Attribute Blowup (CAPEC-229)

Ιστορία

Το CAPEC δημιουργήθηκε από το Υπουργείο Εσωτερικής Ασφάλειας των ΗΠΑ στο πλαίσιο της στρατηγικής πρωτοβουλίας Software Assurance (SwA) του Γραφείου Κυβερνοασφάλειας και Επικοινωνιών (CS&C). Αρχικά κυκλοφόρησε το 2007 και ο κατάλογος CAPEC συνεχίζει να εξελίσσεται με τη συμμετοχή του κοινού και τις συνεισφορές για να δημιουργηθεί ένας τυποποιημένος μηχανισμός για τον εντοπισμό, τη συλλογή, τη βελτίωση και την κοινοποίηση μοτίβων επιθέσεων στην κοινότητα κυβερνοασφάλειας.

Οφέλη

Τα μοτίβα επίθεσης, όταν καταγράφονται επίσημα, προσφέρουν αξία στην ανάπτυξη και συντήρηση δυνατοτήτων που υποστηρίζονται από την κυβερνοτεχνολογία, συμβάλλοντας στην εκπαίδευση προγραμματιστών και διευθυντών, στον ορισμό απειλών, στην ανάλυση αρχιτεκτονικού κινδύνου, στην προτεραιοποίηση δραστηριοτήτων ανασκόπησης, στη δοκιμή διείσδυσης, στην παρακολούθηση τάσεων και στην παροχή προληπτικής καθοδήγησης βάσει εμπειριών.

Φυσικά, τα μοτίβα επίθεσης δεν είναι το μόνο χρήσιμο εργαλείο για την κατασκευή ασφαλών δυνατοτήτων που υποστηρίζονται από την κυβερνοτεχνολογία. Πολλά άλλα εργαλεία,

όπως τα misuse/abuse cases (περιπτώσεις κακής χρήσης/κακοποίησης), οι ασφαλιστικές απαιτήσεις, τα μοντέλα απειλών, η γνώση κοινών αδυναμιών και ευπαθειών, και τα δέντρα επιθέσεων μπορούν να βοηθήσουν. Τα μοτίβα επίθεσης διαδραματίζουν έναν μοναδικό ρόλο μέσα σε αυτή τη μεγαλύτερη αρχιτεκτονική ασφάλειας και τεχνικών.

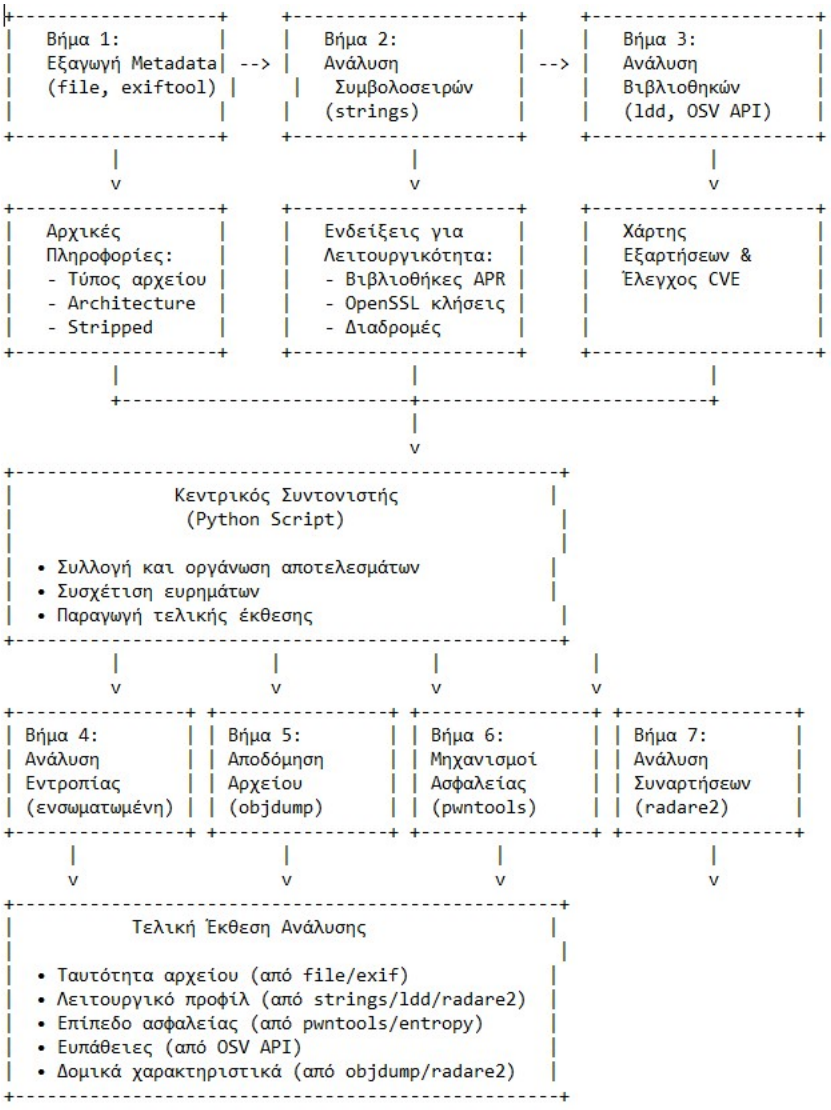
ΚΕΦΑΛΑΙΟ 5: ΠΑΡΟΥΣΙΑΣΗ ΤΟΥ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΥ SCRIPT

Στο πλαίσιο της παρούσας εργασίας υλοποιήθηκε ένα εργαλείο στατικής ανάλυσης δυαδικών αρχείων, το οποίο συνδυάζει τη λειτουργικότητα διαφόρων εξειδικευμένων εργαλείων ανάλυσης. Η υλοποίηση πραγματοποιήθηκε σε γλώσσα προγραμματισμού Python, η οποία επιλέχθηκε λόγω της ευκολίας ενσωμάτωσης εξωτερικών εργαλείων μέσω βιβλιοθηκών και της δυνατότητας αυτοματοποίησης διαδικασιών ανάλυσης.

Το προτεινόμενο σύστημα λειτουργεί ως ένα ενιαίο περιβάλλον εκτέλεσης διαφορετικών εργαλείων ανάλυσης, τα οποία χρησιμοποιούνται για την εξαγωγή πληροφοριών σχετικά με τη δομή, τις εξαρτήσεις και τους μηχανισμούς ασφαλείας ενός δυαδικού αρχείου. Συγκεκριμένα, το σύστημα αξιοποιεί, από τα εργαλεία που προαναφέραμε, το **strings**, **objdump**, **radare2** και τη βιβλιοθήκη **pwntools**, ενώ παράλληλα πραγματοποιεί επιπλέον επεξεργασία δεδομένων μέσω συναρτήσεων που έχουν υλοποιηθεί σε Python.

Η λειτουργία του εργαλείου βασίζεται στη διαδοχική εκτέλεση διαφορετικών σταδίων ανάλυσης. Αρχικά πραγματοποιείται συλλογή βασικών πληροφοριών για το δυαδικό αρχείο και στη συνέχεια εφαρμόζονται επιμέρους τεχνικές ανάλυσης, όπως η εξέταση της εντροπίας του αρχείου, η ανίχνευση δυναμικών βιβλιοθηκών και ο έλεγχος μηχανισμών ασφαλείας του εκτελέσιμου. Τα αποτελέσματα από κάθε εργαλείο συγκεντρώνονται και παρουσιάζονται στον χρήστη με ενιαίο τρόπο, επιτρέποντας μια γρήγορη και συνοπτική επισκόπηση των βασικών χαρακτηριστικών του binary.

Στο ακόλουθο διάγραμμα παρουσιάζεται η γενική αρχιτεκτονική λειτουργίας του προτεινόμενου συστήματος και η συνεργασία των επιμέρους εργαλείων που χρησιμοποιούνται κατά τη διαδικασία ανάλυσης.



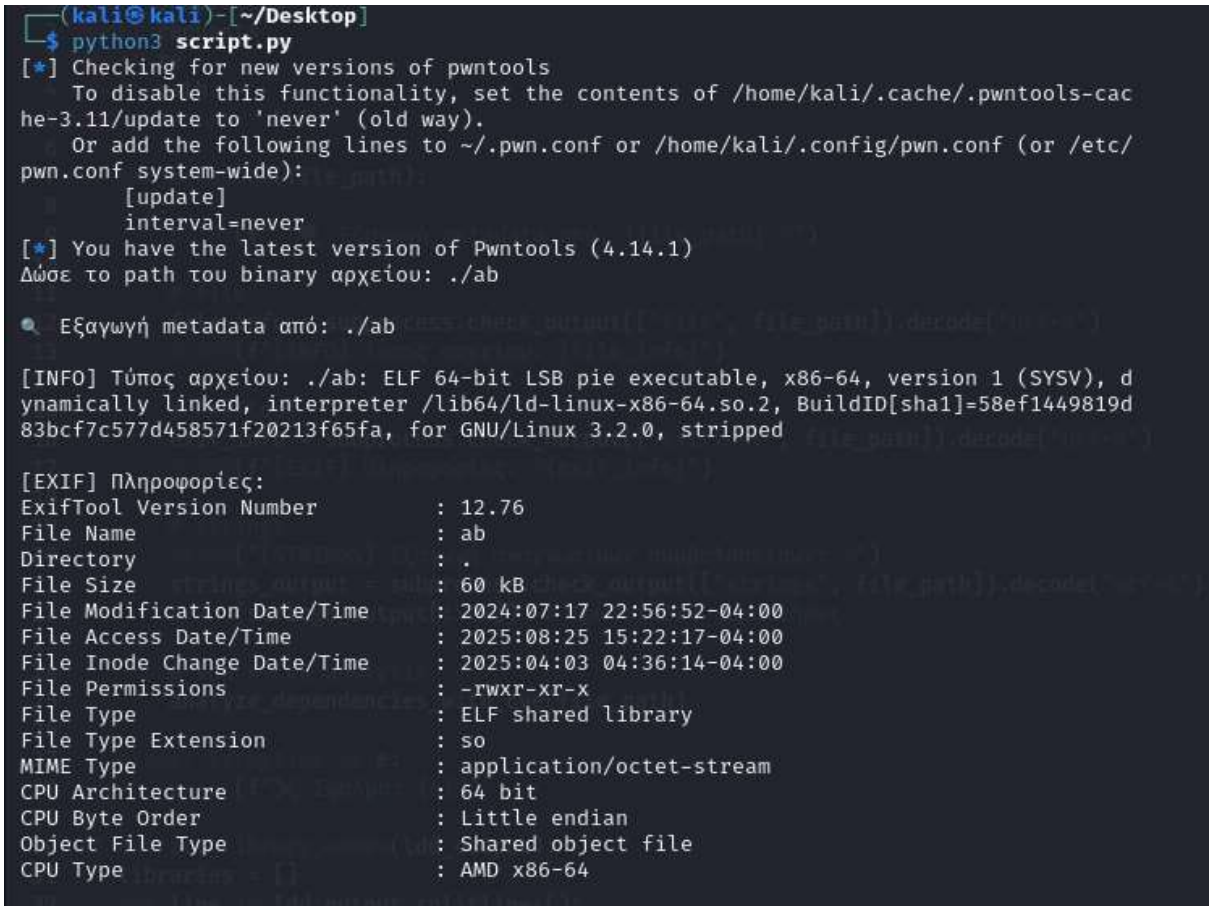
Εικόνα 13. Σχεδιάγραμμα δομής του Script.

Βήμα 1ο: Εξαγωγή metadata.

Τα εργαλεία που θα χρησιμοποιήσουμε για να εξάγουμε τα metadata του Binary αρχείου είναι το **file** και το **exif**. Για όλα τα εργαλεία που θα χρησιμοποιήσουμε θα βασιστούμε στο subprocess module το οποίο είναι ιδανικό για να εκτελεί command-line εργαλεία και να εξάγει τα αποτελέσματά τους.

```python
import subprocess
import requests
import re
import math
from pwn import ELF

def get_metadata(file_path):
    try:
        print(f"\n🔍 Εξαγωγή metadata από: {file_path}\n")

        # file
        file_info = subprocess.check_output(["file", file_path]).decode("utf-8")
        print(f"[INFO] Τύπος αρχείου: {file_info}")

        # exiftool
        exif_info = subprocess.check_output(["exiftool", file_path]).decode("utf-8")
        print(f"[EXIF] Πληροφορίες:\n{exif_info}")

        # strings
        print("[STRINGS] Εξαγωγή αναγνώσιμων συμβολοσειρών:\n")
        strings_output = subprocess.check_output(["strings", file_path]).decode("utf-8")
        print(strings_output[:500])  # Πρώτοι 500 χαρακτήρες

        # ldd + CVE analysis
        analyze_dependencies_with_cve(file_path)

    except Exception as e:
        print(f"❌ Σφάλμα: {e}")
```

Εικόνα 14. Τμήμα εξαγωγής μεταδεδομένων.



Εικόνα 15. Αποτελέσματα των εργαλείων file & EXIF.

FILE Tool: Το εργαλείο File μας λέει τα βασικά χαρακτηριστικά του binary

Σύμφωνα με το file δείχνει πως το αρχείο είναι:

- ELF 64-bit LSB pie executable : Το αρχείο είναι για little-endian αρχιτεκτονική. Ο κώδικας του προγράμματος μπορεί να φορτωθεί και να εκτελεστεί από οποιαδήποτε τυχαία διεύθυνση στη μνήμη, αντί από μια συγκεκριμένη, προκαθορισμένη διεύθυνση.
- x86-64 : Η αρχιτεκτονική του εκτελέσιμου.
- version 1 (SYSV): Χρησιμοποιεί την έκδοση 1 του προτύπου System V ABI, το οποίο είναι ένα πρωτόκολλο για αλληλεπίδραση μεταξύ του πυρήνα του λειτουργικού συστήματος και των εκτελέσιμων προγραμμάτων.
- dynamically linked: Χρησιμοποιεί shared libraries.

- interpreter /lib64/ld-linux-x86-64.so.2: Ο dynamic linker που φορτώνει τις βιβλιοθήκες στη μνήμη πριν την εκτέλεση του αρχείου.
- BuildID[sha1]=58ef1449819d83bcf7c577d458571f20213665fa: Hash που επαληθεύει ότι το αρχείο δεν έχει τροποποιηθεί.
- Stripped: Έχουν αφαιρεθεί τα debugging symbols όπως ονόματα συναρτήσεων και μεταβλητών από το εκτελέσιμο αρχείο. Αυτό το κάνει δύσκολο να γίνει reverse engineering και να αποσφαλματωθεί χωρίς ειδικά εργαλεία.

EXIF: Το exiftool δείχνει metadata του αρχείου:

- Όνομα και directory: ab στο Desktop.
- Μέγεθος: 60 kB.
- Ημερομηνίες: δημιουργίας, τροποποίησης, πρόσβασης.
- Permissions: εκτελέσιμο (-rwxr-xr-x).
- MIME type: application/octet-stream → γενικό binary.
- Τύπος του αρχείου : ELF shared library
- File Type Extension : so. Το exiftool ταξινομεί το αρχείο ab όχι ως executable αλλά ως shared library γιατί τα PIE μοιάζουν πολύ με shared libraries από την άποψη δομής.
- MIME Type : application/octet-stream. Δηλώνει ότι το αρχείο είναι ένα πρόγραμμα.
- Object File Type : Shared object file
- CPU Architecture : 64 bit
- CPU Byte Order : Little endian
- CPU Type : AMD x86-64

Η εξαγωγή μεταδεδομένων μέσω των εργαλείων file και exiftool συμβάλλει στην κατανόηση της δομής και της προέλευσης του εκτελέσιμου αρχείου. Η αναγνώριση της αρχιτεκτονικής, του τύπου αρχείου και πιθανών στοιχείων μεταγλώττισης επιτρέπει την αξιολόγηση της αξιοπιστίας , της ακεραιότητάς του και αποτελεί το πρώτο βήμα για την εκτίμηση του επιπέδου κινδύνου ενός δυαδικού.

Βήμα 2ο: Ανάλυση συμβολοσειρών με το Strings.

```
[STRINGS] Εξαγωγή αναγνώσιμων συμβολοσειρών:

/lib64/ld-linux-x86-64.so.2
 !?e
apr_file_read_full
_ITM_deregisterTMCloneTable
apr_pstrmemdup
apr_time_now
apr_pool_create_ex
apr_pool_clear
apr_file_info_get
apr_socket_connect
apr_pollset_create
apr_pollset_remove
apr_socket_recv
__gmon_start__
apr_file_close
apr_pstrdup
apr_socket_create
apr_socket_timeout_set
apr_snprintf
apr_psprintf
apr_socket_close
apr_sockaddr_info_get
apr_pool_destroy
apr_file_open
apr_pollset_poll
apr_pollset_add
apr_pstrcat
_ITM_registerTMCloneTable
apr_base64_encode
```

Εικόνα 16. Αποτελέσματα αναγνώσιμων συμβολοσειρών με το εργαλείο strings.

STRINGS: Είναι εργαλείο που σκανάρει ένα δυαδικό αρχείο και εξάγει όλες τις αναγνώσιμες ακολουθίες χαρακτήρων που βρίσκει μέσα σε αυτό.

Τα αποτελέσματα χωρίζονται σε τρεις κατηγορίες:

Σύστημα & Runtime:

- /lib64/ld-linux-x86-64.so.2: Ο dynamic linker.

- _gmon_start_: ένα πρόγραμμα profiling πρόγραμμα που μετράει το χρόνο εκτέλεσης των συναρτήσεων.
- _ITM_deregisterTMCloneTable & _ITM_registerTMCloneTable: Αυτά σχετίζονται με τον μεταγλωττιστή **GCC** και μηχανισμούς για τη διαχείριση threads.

APR functions:

Η πλειοψηφία των strings που βλέπουμε ξεκινούν με apr_. Αυτά είναι ονόματα συναρτήσεων από την βιβλιοθήκη APR. Η APR (Apache Portable Runtime) είναι μια βιβλιοθήκη που δημιουργήθηκε από το project του Apache HTTP Server και μας κάνει να υποψιαζόμαστε ότι το αρχείο αυτό είναι ApacheBench το οποίο είναι ένα εργαλείο γραμμής εντολών για να εξετάσουμε την απόδοση ενός HTTP server.

Random:

- !?e

Η ανάλυση συμβολοσειρών σχετίζεται με ευπάθειες όπως format string vulnerabilities και διαρροή δεδομένων, καθώς επιτρέπει την αποκάλυψη hard-coded πληροφοριών, κλειδιών, URL ή ευαίσθητων μηνυμάτων, όπως αναφέρθηκε στο Κεφάλαιο 4.

Βήμα 3ο: Ανάλυση βιβλιοθηκών με το ldd.

Επόμεν στάδιο είναι να εμφανίσουμε τις δυναμικές βιβλιοθήκες οι οποίες χρειάζονται για να εκτελεστεί το excecutable μας. Αυτό επιτυγχάνεται με την εντολή **ldd** (List Dynamic Dependencies).

```
def analyze_dependencies_with_cve(file_path):
    print("\n[LDD] Ανίχνευση δυναμικών βιβλιοθηκών:\n")
    try:
        ldd_output = subprocess.check_output(["ldd", file_path]).decode("utf-8")
        print(ldd_output)

        libs = extract_library_names(ldd_output)
        print("\n[CVE CHECK] Έλεγχος εξαρτήσεων για γνωστές ευπάθειες:\n")

        for lib in libs:
            print(f"🔎 Βιβλιοθήκη: {lib}")
            vulns = check_cves_osv(lib)
            if vulns:
                for v in vulns[:3]:  # εμφάνιση έως 3 CVEs για κάθε lib
                    print(f"⚠️ CVE: {v['id']} - {v['summary']}")
            else:
                print("✅ Δεν εντοπίστηκαν γνωστές ευπάθειες.\n")

    except Exception as e:
        print(f"❌ Σφάλμα στην ανίχνευση εξαρτήσεων ή CVE: {e}")
```

Εικόνα 17. Ανίχνευση δηλωμένων ευπαθείων στο σύστημα CVE.

```
[LDD] Ανίχνευση δυναμικών βιβλιοθηκών:

        linux-vdso.so.1 (0×00007ffc8bda0000)
        libaprutil-1.so.0 ⇒ /lib/x86_64-linux-gnu/libaprutil-1.so.0 (0×00007ff485d8d0
00)
        libapr-1.so.0 ⇒ /lib/x86_64-linux-gnu/libapr-1.so.0 (0×00007ff485d4f000)
        libm.so.6 ⇒ /lib/x86_64-linux-gnu/libm.so.6 (0×00007ff485c6d000)
        libssl.so.3 ⇒ /lib/x86_64-linux-gnu/libssl.so.3 (0×00007ff485b77000)
        libcrypto.so.3 ⇒ /lib/x86_64-linux-gnu/libcrypto.so.3 (0×00007ff485600000)
        libc.so.6 ⇒ /lib/x86_64-linux-gnu/libc.so.6 (0×00007ff48541b000)
        libexpat.so.1 ⇒ /lib/x86_64-linux-gnu/libexpat.so.1 (0×00007ff4853f0000)
        libcrypt.so.1 ⇒ /lib/x86_64-linux-gnu/libcrypt.so.1 (0×00007ff4853b4000)
        libuuid.so.1 ⇒ /lib/x86_64-linux-gnu/libuuid.so.1 (0×00007ff4853aa000)
        /lib64/ld-linux-x86-64.so.2 (0×00007ff485de9000)
        libz.so.1 ⇒ /lib/x86_64-linux-gnu/libz.so.1 (0×00007ff48538b000)
        libzstd.so.1 ⇒ /lib/x86_64-linux-gnu/libzstd.so.1 (0×00007ff4852c1000)
```

Εικόνα 18. Ανίχνευση δυναμικών βιλβιοθήκων.

Το πρώτο μέρος του κώδικα, όπως βλέπουμε ανιχνεύει τις δυναμικές βιβλιοθήκες. Εδώ βλέπουμε κάποιες βιβλιοθήκες που αξίζουν να αναφερθούν παραπάνω.

libapr-1.so.0 & libaprutil-1.so.0. Οι κύριες βιβλιοθήκες της APR και της APR Util. Όπως είδαμε και από τα strings, το πρόγραμμα ab βασίζεται βαριά σε αυτές για λειτουργίες όπως δικτύωση, διαχείριση μνήμης, work with files, κλπ.

libssl.so.3 & libcrypto.so.3. Αυτές είναι οι βιβλιοθήκες OpenSSL. Η libcrypto παρέχει βασικές κρυπτογραφικές λειτουργίες και η libssl παρέχει την υλοποίηση των πρωτοκόλλων SSL/TLS. Δείχνει ότι το ab υποστηρίζει σύνδεση μέσω HTTPS.

libc.so.6. Η βασική βιβλιοθήκη C (glibc). είναι η πιο θεμελιώδης βιβλιοθήκη. Παρέχει όλες τις βασικές λειτουργίες όπως η διαχείριση μνήμης (malloc, free), η εργασία με strings (strcpy, printf), κλήσεις συστήματος, κ.λπ.

Αυτή η λίστα είναι **κρίσιμη για τον έλεγχο ασφαλείας**. Το script στη συνέχεια πηγαίνει να ελέγξει αυτές ακριβώς τις βιβλιοθήκες (π.χ., libssl, libcrypto, libc) για γνωστές ευπάθειες (CVEs). Μια ευπάθεια σε μία από αυτές τις βιβλιοθήκες θα μπορούσε να κάνει το αρχείο ή οποιοδήποτε άλλο πρόγραμμα που τις χρησιμοποιεί ευάλωτο, ακόμα κι αν ο ίδιος ο κώδικας του είναι ασφαλής.

```
[CVE CHECK] Έλεγχος εξαρτήσεων για γνωστές ευπάθειες:

🔎 Βιβλιοθήκη: vdso.so.1
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: 1.so.0
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: 1.so.0
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libm.so.6
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libssl.so.3
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libcrypto.so.3
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libc.so.6
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libexpat.so.1
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libcrypt.so.1
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libuuid.so.1
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: 64.so.2
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libz.so.1
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.

🔎 Βιβλιοθήκη: libzstd.so.1
✔ Δεν εντοπίστηκαν γνωστές ευπάθειες.
```

Εικόνα 19. Διασταύρωση αποτελεσμάτων από το CVE.

Το script πάει να αντιστοιχίσει κάθε δυναμική βιβλιοθήκη που χρησιμοποιείται από το εκτελέσιμο με ευπάθειες που έχουν καταγραφεί στο database Common Vulnerabilities and Exposures. Απ' ότι βλέπουμε καμία από αυτές τις βιβλιοθήκες δεν έχει εκμεταλλευτεί στο παρελθόν.

Η ανάλυση δυναμικών βιβλιοθηκών και η αντιστοίχισή τους με CVE καταγραφές σχετίζεται άμεσα με την εκμετάλλευση κακόβουλων ή ευάλωτων βιβλιοθηκών (Library Exploitation), όπως

παρουσιάστηκε στο προηγούμενο κεφάλαιο. Ακόμα και αν ο κύριος κώδικας είναι ασφαλής, μια ευάλωτη βιβλιοθήκη μπορεί να καταστήσει ολόκληρη την εφαρμογή επικίνδυνη.

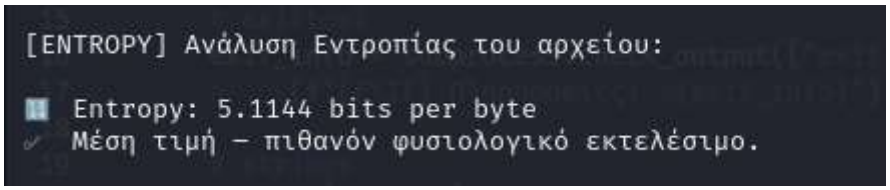
Βήμα 4ο: Υπολογισμός Εντροπίας.

Η εντροπία μετράει το "βαθμό απροσδιοριστίας" ή "την τυχαιότητα" των δεδομένων. Είναι ένας μαθηματικός τρόπος να ποσοτικοποιήσουμε πόσο προβλέψιμα ή μη είναι τα bytes σε ένα αρχείο. Η εντροπία μετριέται σε **bits ανά byte**. Το μέγιστο δυνατό είναι **8 bits/byte**, που σημαίνει απόλυτη τυχαιτητα.

```python
def calculate_entropy(file_path):
    try:
        with open(file_path, "rb") as f:
            byte_arr = list(f.read())

        if len(byte_arr) == 0:
            print("⚠️ Το αρχείο είναι κενό.")
            return

        print("\n[ENTROPY] Ανάλυση Εντροπίας του αρχείου:\n")

        freq_list = [0] * 256
        for b in byte_arr:
            freq_list[b] += 1

        entropy = 0.0
        for freq in freq_list:
            if freq == 0:
                continue
            p = freq / len(byte_arr)
            entropy -= p * math.log2(p)

        print(f"🔢 Entropy: {entropy:.4f} bits per byte")


        if entropy > 7.5:
            print("⚠️ Πολύ υψηλή εντροπία – πιθανή κρυπτογράφηση ή συμπίεση.")
        elif entropy < 3:
            print("ℹ️ Πολύ χαμηλή εντροπία – πιθανό απλό ή μη binary αρχείο.")
        else:
            print("✅ Μέση τιμή – πιθανόν φυσιολογικό εκτελέσιμο.")

    except Exception as e:
        print(f"❌ Σφάλμα κατά την ανάλυση εντροπίας: {e}")
```

Εικόνα 20. Υπολογισμός εντροπίας.

Η συνάρτηση calculate_entropy διαβάζει το αρχείο byte-προς-byte και υπολογίζει την εντροπία του ακολουθώντας μερικά βήματα. Πρώτα διαβάζει όλα τα bytes, μετράει τη συχνότητα εμφάνισης κάθε δυνατού byte, υπολογίζει την πιθανότητα p να εμφανιστεί ένα συγκεκριμένο byte και υπολογίζει την εντροπία χρησιμοποιώντας τον τύπο Shannon:
Entropy = - Σ [p(byte_i) * log2(p(byte_i))] για κάθε byte i από το 0 έως το 255.



Εικόνα 21. Αποτέλεσμα εντροπίας.

Βλέπουμε ότι εμφανίζει το μήνυμα Μέση Τιμή με τιμή 5.1144 bits per bytes. Αυτό σημαίνει ότι τα δεδομένα έχουν ένα μείγμα από τυχαιότητα και δομή. Αυτό είναι το πιο συνηθισμένο σενάριο για κανονικά binaries. Αυτλο οφείλεται στο ότι ο κώδικας του ELF έχει μια πολύπλοκη αλλά όχι τελείως τυχαία δομή. Περιέχει οδηγίες, δεδομένα, strings και άλλα sections που του δίνουν μια "φυσιολογική" εντροπία.

Βήμα 5ο: Αποδόμηση αρχείου με το objdump.

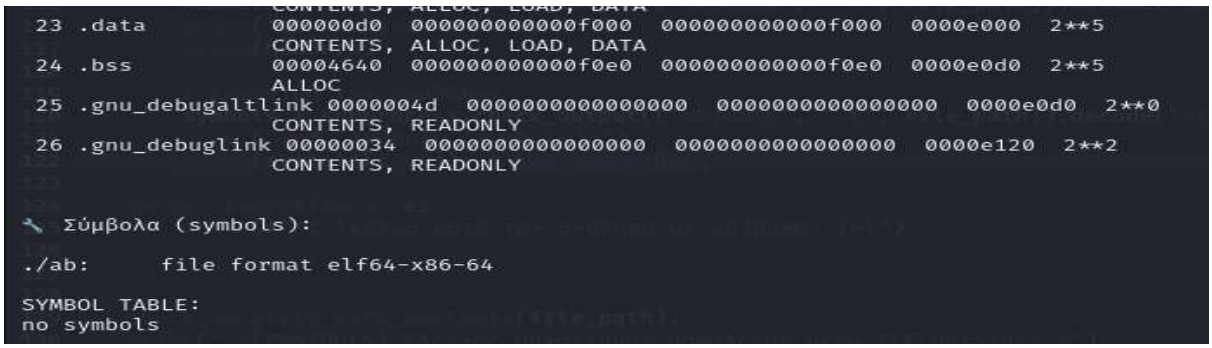
Το **objdump** είναι ένα εργαλείο που αποδομεί τα εκτελέσιμα αρχεία και εμφανίζει τις λεπτομέρειες της δομής τους. Το script τρέχει με δύο flags, -h που εμφανίζει τις κεφαλίδες των sections του αρχείου και -t που εμφανίζει το symbol table.

```python
def analyze_with_objdump(file_path):
    try:
        print("\n[OBJDUMP] Ανάλυση τμημάτων και συμβόλων:\n")

        # Εμφάνιση των sections
        headers = subprocess.check_output(["objdump", "-h", file_path]).decode("utf-8")
        print("📦 Sections (headers):")
        print(headers)

        # Εμφάνιση των συμβόλων
        symbols = subprocess.check_output(["objdump", "-t", file_path]).decode("utf-8")
        print("\n🔧 Σύμβολα (symbols):")
        print(symbols[:500])  # 500 χαρακτήρες

    except Exception as e:
        print(f"❌ Σφάλμα κατά την ανάλυση με objdump: {e}")
```

Εικόνα 22. Χρήση του εργαλείου objdump.

```
[OBJDUMP] Ανάλυση τμημάτων και συμβόλων:

📦 Sections (headers):

./ab:     file format elf64-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .interp       0000001c  0000000000000318  0000000000000318  00000318  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.gnu.property 00000020  0000000000000338  0000000000000338  00000338  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .note.gnu.build-id 00000024  0000000000000358  0000000000000358  00000358  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .note.ABI-tag 00000020  000000000000037c  000000000000037c  0000037c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .gnu.hash     00000030  00000000000003a0  00000000000003a0  000003a0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .dynsym       00000d38  00000000000003d0  00000000000003d0  000003d0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .dynstr       000008ed  0000000000001108  0000000000001108  00001108  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .gnu.version  0000011a  00000000000019f6  00000000000019f6  000019f6  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .gnu.version_r 000000e0  0000000000001b10  0000000000001b10  00001b10  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  9 .rela.dyn     000001b0  0000000000001bf0  0000000000001bf0  00001bf0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 10 .rela.plt     00000c60  0000000000001da0  0000000000001da0  00001da0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 11 .init         00000017  0000000000003000  0000000000003000  00003000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 12 .plt          00000850  0000000000003020  0000000000003020  00003020  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 13 .plt.got      00000008  0000000000003870  0000000000003870  00003870  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 14 .text         0000578e  0000000000003880  0000000000003880  00003880  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 15 .fini         00000009  0000000000009010  0000000000009010  00009010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 16 .rodata       00002b60  000000000000a000  000000000000a000  0000a000  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 17 .eh_frame_hdr 000000fc  000000000000cb60  000000000000cb60  0000cb60  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 18 .eh_frame     00000690  000000000000cc60  000000000000cc60  0000cc60  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 19 .init_array   00000008  000000000000e948  000000000000e948  0000d948  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 20 .fini_array   00000008  000000000000e950  000000000000e950  0000d950  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 21 .dynamic      00000240  000000000000e958  000000000000e958  0000d958  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 22 .got          00000468  000000000000eb98  000000000000eb98  0000db98  2**3
```

Εικόνα 23. Αποτελέσματα του εργαλείου objdump, part1.



Εικόνα 24. Αποτελέσματα του εργαλείου objdump, part2.

Τα sections είναι τα διαφορετικά τμήματα από τα οποία αποτελείται ένα ELF εκτελέσιμο. Κάθε section έχει έναν συγκεκριμένο σκοπό. Τα πιο κρίσιμα headers είναι:

- .interp: Περιέχει τη διαδρομή (/lib64/ld-linux-x86-64.so.2) προς τον dynamic linker. Επιβεβαιώνει ότι είναι δυναμικά συνδεδεμένο.
- .dynamic: Περιέχει πληροφορίες που χρειάζονται από τον δυναμικό linker, όπως τις βιβλιοθήκες που πρέπει να φορτώσει.
- .dynsym & .dynstr: Ο δυναμικός πίνακας συμβόλων και οι συμβολοσειρές του. Περιέχουν τα ονόματα των συναρτήσεων από τις δυναμικές βιβλιοθήκες (όπως οι apr_*) που χρειάζεται το πρόγραμμα.
- .rela.dyn & .rela.plt: Πίνακες relocation. Περιέχουν οδηγίες για τον linker πώς να διορθώσει τις διευθύνσεις μνήμης όταν φορτώσει το πρόγραμμα και τις βιβλιοθήκες.
- .text: Αυτό είναι πολύ σημαντικό section. Εδώ βρίσκεται ο πραγματικός κώδικας μηχανής που εκτελεί το πρόγραμμα.
- .note.gnu.build-id: Section που περιέχει ένα μοναδικό SHA1 hash (BuildID[sha1]=58ef1449819d83bcf7c577d458571f20213665fa) που ταιριάζει με αυτό που είδαμε στην εντολή file. Χρησιμοποιείται για να συσχετιστεί το εκτελέσιμο με τα αντίστοιχα αρχεία debugging.
- .init & .fini: Περιέχουν κώδικα που εκτελείται αυτόματα πριν από την main() και αφού τελειώσει η main().

Όσων αφορά των πίνακα των συμβόλων, εκτυπώνεται η τιμή no symbols. Αυτό είναι πολύ σημαντικό και συμβαδίζει απόλυτα με το προηγούμενο εύρημα της εντολής file που έλεγε stripped. Stripped σημαίνει ότι ο πίνακας που περιέχει τα ονόματα των συναρτήσεων, μεταβλητών και άλλων ετικετών που χρησιμοποιούν τα εργαλεία debugging, έχει αφαιρεθεί από το εκτελέσιμο αρχείο.

Έτσι μειώνεται το μέγεθος του αρχείου δυσκολεύει τεχνικές ,όπως reverse engineering, καθώς δεν μπορείς εύκολα να δεις ποια συνάρτηση κάνει τι. Ο linker δεν χρειάζεται τον πίνακα symbols για να εκτελέσει το πρόγραμμα. Χρειάζεται μόνο τα "εξωτερικά" symbols από βιβλιοθήκες, τα οποία βρίσκονται στον δυναμικό πίνακα συμβόλων (.dynsym), που δεν έχει αφαιρεθεί.

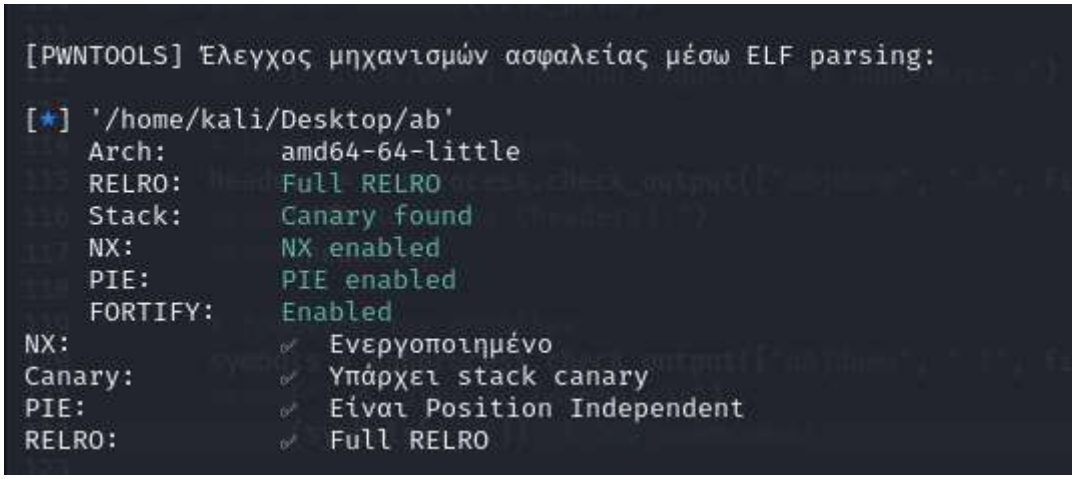
Άρα, παίζει πολύ σημαντικό ρόλο η κατανόηση της διάταξης της μνήμης και των τμημάτων του προγράμματος για να μπορούν να αποφευχθούν ευπάθειες όπως buffer overflow και οι out-of-bounds εγγραφές οι οποίες καταγράφονται οι οποίες καταγράφονται ως CWE-120, CWE-121 και CWE-787 στον πίνακα Common Weakness Enumeration.

Βήμα 6ο: Ανάλυση των Μηχανισμών Ασφαλείας με το pwntools.

Η βιβλιοθήκη **pwntools** εξαγάγει πληροφορίες από το header του ELF αρχείου σχετικά με τους μηχανισμούς ασφαλείας με τους οποίους έχει γίνει compiled το πρόγραμμα.

```python
def check_security_with_pwntools(file_path):
    print("\n[PWNTOOLS] Έλεγχος μηχανισμών ασφαλείας μέσω ELF parsing:\n")
    try:
        elf = ELF(file_path)

        print(f"NX:\t\t{'✅ Ενεργοποιημένο' if elf.nx else '❌ Απενεργοποιημένο'}")
        print(f"Canary:\t\t{'✅ Υπάρχει stack canary' if elf.canary else '❌ Δεν υπάρχει'}")
        print(f"PIE:\t\t{'✅ Είναι Position Independent' if elf.pie else '❌ Όχι PIE'}")

        relro = elf.relro
        if relro == 'Full':
            relro_status = '✅ Full RELRO'
        elif relro == 'Partial':
            relro_status = '🟡 Partial RELRO'
        else:
            relro_status = '❌ Χωρίς RELRO'
        print(f"RELRO:\t\t{relro_status}")

    except Exception as e:
        print(f"❌ Σφάλμα κατά την ανάλυση με pwntools: {e}")
```

Εικόνα 25. Χρήση της βιβλιοθήκης pwntools.



Εικόνα 26. Αποτελέσματα της βιβλιοθήκης pwntools.

NX (No-Execute / DEP) - Ενεργοποιημένο

Ο μηχανισμός No-eXecute διαχωρίζει τη μνήμη σε δύο μέρη, σε αυτό που μπορεί να εκτελεστεί (κώδικας) και αυτό που μπορεί να γραφτεί (δεδομένα). Ο μηχανισμός αυτός εμποδίζει έναν attacker να γράψει κακόβουλο κώδικα σε μια μνήμη και μετά να τον εκτελέσει. Η ενεργοποίησή του είναι υποχρεωτική στα σύγχρονα συστήματα.

Stack Canary - Υπάρχει stack canary

Το stack canary είναι μυστικό, τυχαίο value που τοποθετείται στο stack δίπλα σε κρίσιμα δεδομένα όπως την επιστροφή διεύθυνσης μιας συνάρτησης. Πριν η συνάρτηση τελειώσει, ελέγχεται ότι αυτή η τιμή δεν άλλαξε.

Αν ένας attacker ξεπεράσει ένα buffer στη στοίβα και αλλάξει την επιστροφή διεύθυνσης για να πάρει έλεγχο της ροής του προγράμματος, θα αλλάξει και το canary. Αυτό θα ανιχνευτεί και το πρόγραμμα θα τερματιστεί αμέσως. Πολύ αποτελεσματικός αμυντικός μηχανισμός κατά των stack-based buffer overflows.

PIE (Position-Independent Executable) - Ενεργοποιημένο

Όπως εξηγήσαμε νωρίτερα, σημαίνει ότι ο κώδικας του προγράμματος μπορεί να φορτωθεί και να εκτελεστεί από οποιαδήποτε τυχαία θέση στη μνήμη.

Κάνει πολύ πιο δύσκολο για έναν attacker να μαντέψει τις διευθύνσεις μνήμης συναρτήσεων ή κρισίμων κομματιών κώδικα που χρειάζεται για να κατασκευάσει ένα exploit (Return-Oriented

Programming). Η τεχνική ASLR (Address Space Layout Randomization) του πυρήνα γίνεται πολύ πιο αποτελεσματική όταν το εκτελέσιμο είναι PIE.

RELRO (Relocation Read-Only) - Full RELRO

Μηχανισμός που εφαρμόζει δικαιώματα μόνο για ανάγνωση σε κρίσιμες περιοχές μνήμης που σχετίζονται με τη δυναμική σύνδεση. Όταν είναι full relpo κάνει όλους τους πίνακες relocation read-only πριν αρχίσει να εκτελείται ο κώδικας του προγράμματος.

Αποτρέπει την τροποποίηση που θα μπορούσε να αλλάξει τη ροή εκτέλεσης του προγράμματος, π.χ., να κάνει redirect μιας κλήσης βιβλιοθήκης όπως η system() σε κακόβουλο κώδικα.

FORTIFY - Enabled

Ο μηχανισμός fortify ενισχύει ευάλωτες συναρτήσεις που δέχονται buffers (όπως memcpy, strcpy) με επιπλέον ελέγχους. Ο compiler αντικαθιστά αυτές τις κλήσεις με πιο ασφαλείς εκδοχές (π.χ., strcpy_chk). Ανιχνεύει buffer overflows κατά την εκτέλεση, πριν προκαλέσουν ζημιά, και τερματίζει το πρόγραμμα.

Η χρήση του pwntools στο script στοχεύει στην αξιολόγηση του επιπέδου προστασίας του binary και με τον τρόπο αυτό, η υλοποίηση ενσωματώνει μια προσέγγιση αξιολόγησης ασφάλειας που συνδέεται άμεσα με σύγχρονες πρακτικές ανάλυσης εκμεταλλευσιμότητάς.

Βήμα 7ο: Ανάλυση συναρτήσεων με το Radare2.

Με το radare2 θα αναλύσυμε όλες τις συναρτήσεις που χρησιμοποιεί το αρχείο, είτε είναι imported από δυναμικές βιβλιοθήκες που καλεί το πρόγραμμα είτε είναι function που περιέχει το ίδιο το εκτελέσιμο μέσα στο section .text.

```python
def analyze_with_radare2(file_path):
    print("\n[RADARE2] Ανάλυση δυαδικού με radare2:\n")
    try:

        functions_output = subprocess.check_output(
            ["radare2", "-c", "aaa;afl", "-q", file_path]
        ).decode("utf-8")

        print("🔧 Συναρτήσεις που εντοπίστηκαν:")
        print(functions_output if functions_output else "❌ Δεν βρέθηκαν συναρτήσεις ή το αρχείο δεν είναι κατάλληλο.\n")

        # (imports)
        imports_output = subprocess.check_output(
            ["radare2", "-c", "iij", "-q", file_path]
        ).decode("utf-8")

        print("\n📦 Εισαγόμενες συναρτήσεις (imports):")
        print(imports_output if imports_output else "❌ Δεν βρέθηκαν imports.\n")

    except FileNotFoundError:
        print("❌ Δεν βρέθηκε το εργαλείο radare2. Εγκατάστησέ το με: sudo apt install radare2")
    except subprocess.CalledProcessError as e:
        print(f"❌ Σφάλμα κατά την εκτέλεση radare2:\n{e.output.decode('utf-8')}")
    except Exception as e:
        print(f"❌ Άγνωστο σφάλμα με radare2: {e}")
```

Εικόνα 27. Χρήση του εργαλείου Radare2.

```
[RADARE2] Ανάλυση δυαδικού με radare2:

WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.ca
che=true` next time
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods (af @@ method.*)
INFO: Recovering local variables (afva@@@F)
INFO: Type matching analysis for all functions (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
🔧 Συναρτήσεις που εντοπίστηκαν:
0x00003030    1      6 sym.imp.__printf_chk
0x00003040    1      6 sym.imp.SSL_CTX_use_PrivateKey_file
0x00003050    1      6 sym.imp.SSL_get_error
0x00003060    1      6 sym.imp.SSL_read
0x00003070    1      6 sym.imp.apr_socket_connect
0x00003080    1      6 sym.imp.SSL_get_peer_cert_chain
0x00003090    1      6 sym.imp.__ctype_tolower_loc
0x000030a0    1      6 sym.imp.apr_file_info_get
0x000030b0    1      6 sym.imp.apr_pstrcat
0x000030c0    1      6 sym.imp.apr_file_read_full
0x000030d0    1      6 sym.imp.SSL_CIPHER_get_bits
0x000030e0    1      6 sym.imp.strstr
0x000030f0    1      6 sym.imp.apr_pool_abort_set
0x00003100    1      6 sym.imp.BIO_ctrl
0x00003110    1      6 sym.imp.rand
0x00003120    1      6 sym.imp.apr_file_close
0x00003130    1      6 sym.imp.X509_get_ext_count
0x00003140    1      6 sym.imp.apr_socket_opt_set
0x00003150    1      6 sym.imp.ERR_print_errors
0x00003160    1      6 sym.imp.apr_pollset_poll
0x00003170    1      6 sym.imp.SSL_set_connect_state
0x00003180    1      6 sym.imp.SSL_alert_desc_string_long
0x00003190    1      6 sym.imp.strchr
0x000031a0    1      6 sym.imp.ASN1_UTCTIME_print
0x000031b0    1      6 sym.imp.apr_time_now
0x000031c0    1      6 sym.imp.X509_getm_notAfter
0x000031d0    1      6 sym.imp.SSL_ctrl
0x000031e0    1      6 sym.imp.apr_ipsubnet_create
0x000031f0    1      6 sym.imp.BIO_dump
0x00003200    1      6 sym.imp.apr_pool_clear
0x00003210    1      6 sym.imp.strlen
0x00003220    1      6 sym.imp.BIO_set_callback_arg
0x00003230    1      6 sym.imp.strncmp
0x00003240    1      6 sym.imp.apr_socket_send
0x00003250    1      6 sym.imp.OPENSSL_sk_num
```

Εικόνα 28. Αποτελέσματα του εργαλείου radare2, part 1 .

```
0×00003360    1      6 sym.imp.sqrt
0×00003370    1      6 sym.imp.apr_socket_bind
0×00003380    1      6 sym.imp.apr_strerror
0×00003390    1      6 sym.imp.SSL_CTX_use_certificate_chain_file
0×000033a0    1      6 sym.imp.SSL_get_version
0×000033b0    1      6 sym.imp.memcpy
0×000033c0    1      6 sym.imp.perror
0×000033d0    1      6 sym.imp.EVP_PKEY_get_id
0×000033e0    1      6 sym.imp.__cxa_atexit
0×000033f0    1      6 sym.imp.OPENSSL_init_ssl
0×00003400    1      6 sym.imp.SSL_CTX_ctrl
0×00003410    1      6 sym.imp.time
0×00003420    1      6 sym.imp.strcpy
0×00003430    1      6 sym.imp.SSL_CTX_set_info_callback
0×00003440    1      6 sym.imp.srand
0×00003450    1      6 sym.imp.SSL_CTX_set_cipher_list
0×00003460    1      6 sym.imp.fclose
0×00003470    1      6 sym.imp.X509_get_issuer_name
0×00003480    1      6 sym.imp.apr_pstrdup
0×00003490    1      6 sym.imp.SSL_new
0×000034a0    1      6 sym.imp.SSL_set_bio
0×000034b0    1      6 sym.imp.apr_base64_encode_len
0×000034c0    1      6 sym.imp.apr_pollset_create
0×000034d0    1      6 sym.imp.X509_get_subject_name
0×000034e0    1      6 sym.imp.SSL_CIPHER_get_name
0×000034f0    1      6 sym.imp.OBJ_nid2sn
0×00003500    1      6 sym.imp.__ctype_b_loc
0×00003510    1      6 sym.imp.EVP_PKEY_get_bits
0×00003520    1      6 sym.imp.apr_app_initialize
0×00003530    1      6 sym.imp.__stack_chk_fail
0×00003540    1      6 sym.imp.SSL_shutdown
0×00003550    1      6 sym.imp.BIO_get_callback_arg
0×00003560    1      6 sym.imp.EVP_PKEY_get_utf8_string_param
0×00003570    1      6 sym.imp.apr_pool_create_ex
0×00003580    1      6 sym.imp.fflush
0×00003590    1      6 sym.imp.fopen
0×000035a0    1      6 sym.imp.SSL_CTX_set_ciphersuites
0×000035b0    1      6 sym.imp.X509_get_version
0×000035c0    1      6 sym.imp.BIO_new_socket
0×000035d0    1      6 sym.imp.exit
0×000035e0    1      6 sym.imp.apr_os_sock_get
0×000035f0    1      6 sym.imp.strncasecmp
0×00003600    1      6 sym.imp.apr_socket_create
0×00003610    1      6 sym.imp.apr_pstrmemdup
0×00003620    1      6 sym.imp.OPENSSL_sk_value
0×00003630    1      6 sym.imp.apr_file_open
0×00003640    1      6 sym.imp.malloc
0×00003650    1      6 sym.imp.SSL_get_session
0×00003660    1      6 sym.imp.apr_pool_destroy
0×00003670    1      6 sym.imp.putchar
0×00003680    1      6 sym.imp.SSL_do_handshake
0×00003690    1      6 sym.imp.SSL_CTX_new
0×000036a0    1      6 sym.imp.getpid
0×000036b0    1      6 sym.imp.SSL_alert_type_string_long
```

Εικόνα 29. Αποτελέσματα του εργαλείου radare2, part 2.

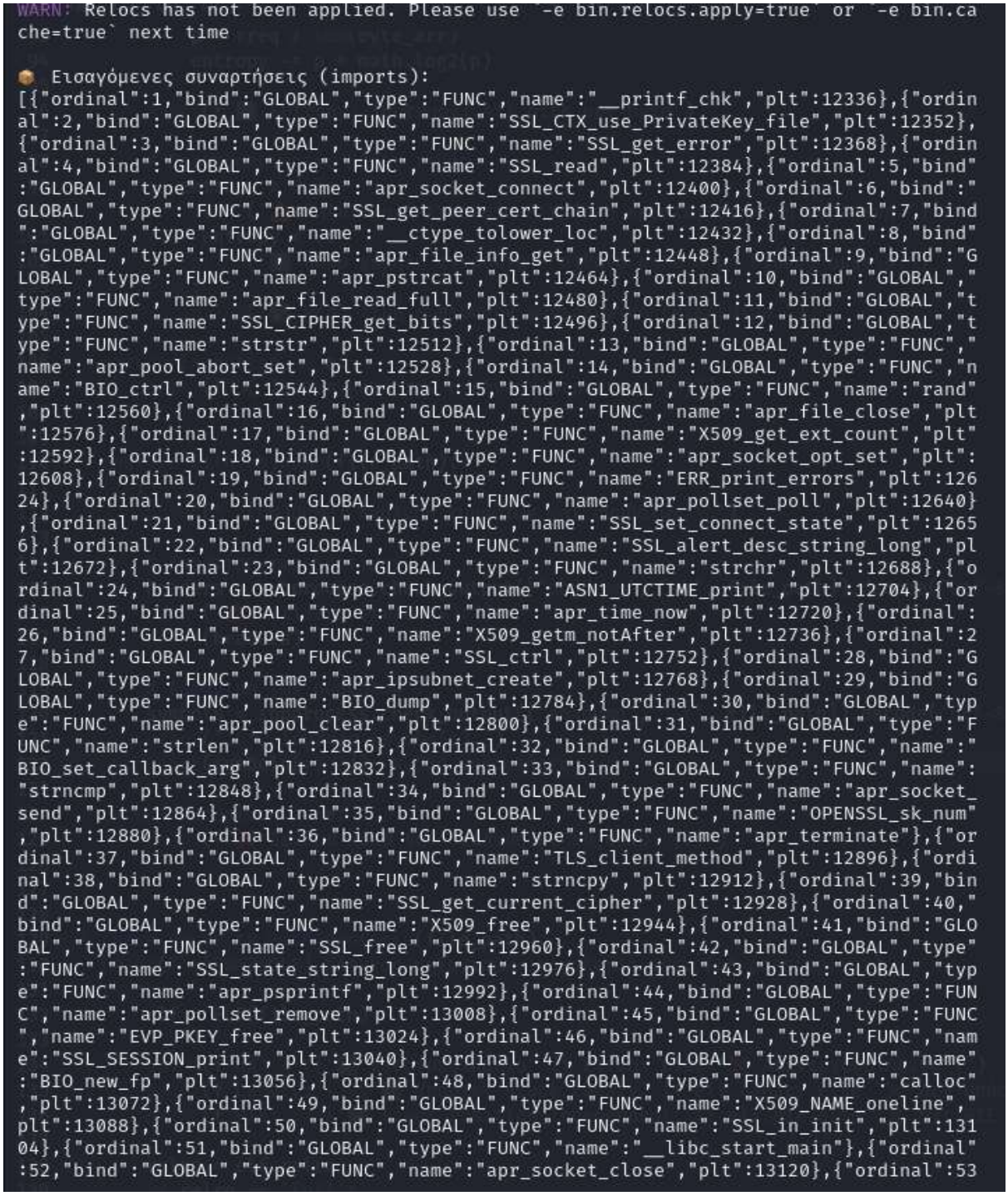
```
0×000036c0    1      6 sym.imp.apr_getopt_init
0×000036d0    1      6 sym.imp.SSL_is_server
0×000036e0    1      6 sym.imp.puts
0×000036f0    1      6 sym.imp.apr_snprintf
0×00003700    1      6 sym.imp.__fprintf_chk
0×00003710    1      6 sym.imp.SSL_CTX_check_private_key
0×00003720    1      6 sym.imp.apr_pollset_add
0×00003730    1      6 sym.imp.apr_base64_encode
0×00003740    1      6 sym.imp.X509_get_pubkey
0×00003750    1      6 sym.imp.apr_signal
0×00003760    1      6 sym.imp.apr_ctime
0×00003770    1      6 sym.imp.RAND_seed
0×00003780    1      6 sym.imp.apr_getopt
0×00003790    1      6 sym.imp.apr_socket_timeout_set
0×000037a0    1      6 sym.imp.strdup
0×000037b0    1      6 sym.imp.SSL_get1_peer_certificate
0×000037c0    1      6 sym.imp.SSL_CTX_set_options
0×000037d0    1      6 sym.imp.BIO_set_callback_ex
0×000037e0    1      6 sym.imp.qsort
0×000037f0    1      6 sym.imp.apr_sockaddr_info_get
0×00003800    1      6 sym.imp.SSL_write
0×00003810    1      6 sym.imp.apr_parse_addr_port
0×00003820    1      6 sym.imp.fwrite
0×00003830    1      6 sym.imp.apr_socket_recv
0×00003840    1      6 sym.imp.X509_getm_notBefore
0×00003850    1      6 sym.imp.__isoc23_strtol
0×00003860    1      6 sym.imp.BIO_printf
0×00003870    1      6 sym.imp.__cxa_finalize
0×00005580    1     33 entry0
0×000038b0  257   7318 main
0×00005660    5     60 entry.init0
0×00005620    5     54 entry.fini0
0×000055b0    4     34 fcn.000055b0
0×00009000    1     14 fcn.00009000
0×00007830   12    415 fcn.00007830
0×00005740    3     76 fcn.00005740
0×00005cc0    3     63 fcn.00005cc0
0×000074f0    3    157 fcn.000074f0
0×00005790    1   1315 fcn.00005790
0×00005d80    3     70 fcn.00005d80
0×00007f40   47   1072 fcn.00007f40
0×00008770  102   2116 fcn.00008770
0×000076a0    9    110 fcn.000076a0
0×00008540   25    528 fcn.00008540
0×000079f0   44   1302 fcn.000079f0
0×00005d00    3    118 fcn.00005d00
0×00006090  120   5208 fcn.00006090
0×00005e80    3    370 fcn.00005e80
0×00007720    3    269 fcn.00007720
0×00008390   16    412 fcn.00008390
0×00006000   11    136 fcn.00006000
```

Εικόνα 30. Εκτύπωση Εσωτερικών Συναρτήσεων με το εργαλείο radare2.

Ανάλυση Εσωτερικών Συναρτήσεων

Εδώ το **radare2** βρήκε τις συναρτήσεις που υπάρχουν μέσα στο δυαδικό.

- main στο 0x000038b0: Αυτή είναι η κύρια function του προγράμματος.
- entry0, entry.init0, entry.fini0: Αυτές είναι functions που σχετίζονται με την εκκίνηση και τον τερματισμό του προγράμματος .
- Άλλες functions (fcn.00005790, fcn.00008770): Λόγω του ότι το αρχείο είναι stripped, έχουν χάσει τα ονόματά τους και το radare2 τις ονομάζει με βάση τη διεύθυνση μνήμης τους.



Εικόνα 31. Εκτύπωση Εισαγόμενων Συναρτήσεων με το εργαλείο radare2, part 1.

```
,"bind":"GLOBAL","type":"FUNC","name":"SSL_CIPHER_get_version","plt":13136},{"ordinal"
:54,"bind":"GLOBAL","type":"FUNC","name":"sqrt","plt":13152},{"ordinal":55,"bind":"GLO
BAL","type":"FUNC","name":"apr_socket_bind","plt":13168},{"ordinal":56,"bind":"GLOBAL"
,"type":"FUNC","name":"apr_strerror","plt":13184},{"ordinal":57,"bind":"GLOBAL","type"
:"FUNC","name":"SSL_CTX_use_certificate_chain_file","plt":13200},{"ordinal":58,"bind":
"GLOBAL","type":"FUNC","name":"SSL_get_version","plt":13216},{"ordinal":59,"bind":"GLO
BAL","type":"FUNC","name":"memcpy","plt":13232},{"ordinal":60,"bind":"GLOBAL","type":"
FUNC","name":"perror","plt":13248},{"ordinal":61,"bind":"GLOBAL","type":"FUNC","name":
"EVP_PKEY_get_id","plt":13264},{"ordinal":62,"bind":"GLOBAL","type":"FUNC","name":"__c
xa_atexit","plt":13280},{"ordinal":63,"bind":"GLOBAL","type":"FUNC","name":"OPENSSL_in
it_ssl","plt":13296},{"ordinal":64,"bind":"GLOBAL","type":"FUNC","name":"SSL_CTX_ctrl"
,"plt":13312},{"ordinal":65,"bind":"GLOBAL","type":"FUNC","name":"time","plt":13328},{
"ordinal":66,"bind":"GLOBAL","type":"FUNC","name":"strcpy","plt":13344},{"ordinal":67,
"bind":"GLOBAL","type":"FUNC","name":"SSL_CTX_set_info_callback","plt":13360},{"ordina
l":68,"bind":"GLOBAL","type":"FUNC","name":"srand","plt":13376},{"ordinal":69,"bind":"
GLOBAL","type":"FUNC","name":"SSL_CTX_set_cipher_list","plt":13392},{"ordinal":70,"bin
d":"GLOBAL","type":"FUNC","name":"fclose","plt":13408},{"ordinal":71,"bind":"GLOBAL","
type":"FUNC","name":"X509_get_issuer_name","plt":13424},{"ordinal":72,"bind":"GLOBAL",
"type":"FUNC","name":"apr_pstrdup","plt":13440},{"ordinal":73,"bind":"GLOBAL","type":"
FUNC","name":"SSL_new","plt":13456},{"ordinal":74,"bind":"GLOBAL","type":"FUNC","name"
:"SSL_set_bio","plt":13472},{"ordinal":75,"bind":"GLOBAL","type":"FUNC","name":"apr_ba
se64_encode_len","plt":13488},{"ordinal":76,"bind":"GLOBAL","type":"FUNC","name":"apr_
pollset_create","plt":13504},{"ordinal":77,"bind":"GLOBAL","type":"FUNC","name":"X509_
get_subject_name","plt":13520},{"ordinal":78,"bind":"GLOBAL","type":"FUNC","name":"SSL
_CIPHER_get_name","plt":13536},{"ordinal":79,"bind":"GLOBAL","type":"FUNC","name":"OBJ
_nid2sn","plt":13552},{"ordinal":80,"bind":"GLOBAL","type":"FUNC","name":"__ctype_b_lo
c","plt":13568},{"ordinal":81,"bind":"GLOBAL","type":"FUNC","name":"EVP_PKEY_get_bits"
,"plt":13584},{"ordinal":82,"bind":"GLOBAL","type":"FUNC","name":"apr_app_initialize",
"plt":13600},{"ordinal":83,"bind":"GLOBAL","type":"FUNC","name":"__stack_chk_fail","pl
t":13616},{"ordinal":84,"bind":"GLOBAL","type":"FUNC","name":"SSL_shutdown","plt":1363
2},{"ordinal":85,"bind":"GLOBAL","type":"FUNC","name":"BIO_get_callback_arg","plt":136
48},{"ordinal":86,"bind":"GLOBAL","type":"FUNC","name":"EVP_PKEY_get_utf8_string_param
","plt":13664},{"ordinal":87,"bind":"GLOBAL","type":"FUNC","name":"apr_pool_create_ex"
,"plt":13680},{"ordinal":88,"bind":"GLOBAL","type":"FUNC","name":"fflush","plt":13696}
,{"ordinal":89,"bind":"GLOBAL","type":"FUNC","name":"fopen","plt":13712},{"ordinal":90
,"bind":"GLOBAL","type":"FUNC","name":"SSL_CTX_set_ciphersuites","plt":13728},{"ordina
l":91,"bind":"GLOBAL","type":"FUNC","name":"X509_get_version","plt":13744},{"ordinal":
92,"bind":"GLOBAL","type":"FUNC","name":"BIO_new_socket","plt":13760},{"ordinal":93,"b
ind":"GLOBAL","type":"FUNC","name":"exit","plt":13776},{"ordinal":94,"bind":"GLOBAL","
type":"FUNC","name":"apr_os_sock_get","plt":13792},{"ordinal":95,"bind":"GLOBAL","type
":"FUNC","name":"strncasecmp","plt":13808},{"ordinal":96,"bind":"GLOBAL","type":"FUNC"
,"name":"apr_socket_create","plt":13824},{"ordinal":97,"bind":"GLOBAL","type":"FUNC","
name":"apr_pstrmemdup","plt":13840},{"ordinal":98,"bind":"GLOBAL","type":"FUNC","name"
:"OPENSSL_sk_value","plt":13856},{"ordinal":99,"bind":"GLOBAL","type":"FUNC","name":"a
pr_file_open","plt":13872},{"ordinal":100,"bind":"GLOBAL","type":"FUNC","name":"malloc
","plt":13888},{"ordinal":101,"bind":"GLOBAL","type":"FUNC","name":"SSL_get_session","
plt":13904},{"ordinal":102,"bind":"GLOBAL","type":"FUNC","name":"apr_pool_destroy","pl
t":13920},{"ordinal":103,"bind":"GLOBAL","type":"FUNC","name":"putchar","plt":13936},{
"ordinal":104,"bind":"GLOBAL","type":"FUNC","name":"SSL_do_handshake","plt":13952},{"o
rdinal":105,"bind":"GLOBAL","type":"FUNC","name":"SSL_CTX_new","plt":13968},{"ordinal"
:106,"bind":"GLOBAL","type":"FUNC","name":"getpid","plt":13984},{"ordinal":107,"bind":
"GLOBAL","type":"FUNC","name":"SSL_alert_type_string_long","plt":14000},{"ordinal":108
,"bind":"GLOBAL","type":"FUNC","name":"apr_getopt_init","plt":14016},{"ordinal":109,"b
ind":"GLOBAL","type":"FUNC","name":"SSL_is_server","plt":14032},{"ordinal":110,"bind":
```


Εικόνα 32. Εκτύπωση Εισαγόμενων Συναρτήσεων με το εργαλείο radare2, part 2.

```
"GLOBAL","type":"FUNC","name":"puts","plt":14048},{"ordinal":111,"bind":"GLOBAL","type
":"FUNC","name":"apr_snprintf","plt":14064},{"ordinal":112,"bind":"GLOBAL","type":"FUN
C","name":"__fprintf_chk","plt":14080},{"ordinal":113,"bind":"GLOBAL","type":"FUNC","n
ame":"SSL_CTX_check_private_key","plt":14096},{"ordinal":114,"bind":"GLOBAL","type":"F
UNC","name":"apr_pollset_add","plt":14112},{"ordinal":115,"bind":"GLOBAL","type":"FUNC
","name":"apr_base64_encode","plt":14128},{"ordinal":116,"bind":"GLOBAL","type":"FUNC"
,"name":"X509_get_pubkey","plt":14144},{"ordinal":117,"bind":"GLOBAL","type":"FUNC","n
ame":"apr_signal","plt":14160},{"ordinal":118,"bind":"GLOBAL","type":"FUNC","name":"ap
r_ctime","plt":14176},{"ordinal":119,"bind":"GLOBAL","type":"FUNC","name":"RAND_seed",
"plt":14192},{"ordinal":120,"bind":"GLOBAL","type":"FUNC","name":"apr_getopt","plt":14
208},{"ordinal":121,"bind":"GLOBAL","type":"FUNC","name":"apr_socket_timeout_set","plt
":14224},{"ordinal":122,"bind":"WEAK","type":"NOTYPE","name":"_ITM_deregisterTMCloneTa
ble"},{"ordinal":123,"bind":"GLOBAL","type":"FUNC","name":"strdup","plt":14240},{"ordi
nal":124,"bind":"GLOBAL","type":"FUNC","name":"SSL_get1_peer_certificate","plt":14256}
,{"ordinal":125,"bind":"GLOBAL","type":"FUNC","name":"SSL_CTX_set_options","plt":14272
},{"ordinal":126,"bind":"WEAK","type":"NOTYPE","name":"__gmon_start__"},{"ordinal":127
,"bind":"GLOBAL","type":"FUNC","name":"BIO_set_callback_ex","plt":14288},{"ordinal":12
8,"bind":"GLOBAL","type":"FUNC","name":"qsort","plt":14304},{"ordinal":129,"bind":"WEA
K","type":"NOTYPE","name":"_ITM_registerTMCloneTable"},{"ordinal":130,"bind":"GLOBAL",
"type":"FUNC","name":"apr_sockaddr_info_get","plt":14320},{"ordinal":131,"bind":"GLOBA
L","type":"FUNC","name":"SSL_write","plt":14336},{"ordinal":132,"bind":"GLOBAL","type"
:"FUNC","name":"apr_parse_addr_port","plt":14352},{"ordinal":133,"bind":"GLOBAL","type
":"FUNC","name":"fwrite","plt":14368},{"ordinal":134,"bind":"GLOBAL","type":"FUNC","na
me":"apr_socket_recv","plt":14384},{"ordinal":135,"bind":"GLOBAL","type":"FUNC","name"
:"X509_getm_notBefore","plt":14400},{"ordinal":136,"bind":"GLOBAL","type":"FUNC","name
":"__isoc23_strtol","plt":14416},{"ordinal":137,"bind":"GLOBAL","type":"FUNC","name":"
BIO_printf","plt":14432},{"ordinal":139,"bind":"WEAK","type":"FUNC","name":"__cxa_fina
lize","plt":14448}]
```


Εικόνα 33. Εκτύπωση Εισαγόμενων Συναρτήσεων με το εργαλείο radare2, part 3.

Ανάλυση Εισαγόμενων Συναρτήσεων

- Συναρτήσεις Apache Portable Runtime
  apr_socket_*, apr_file_*, apr_pool_*, apr_pollset_*,
  apr_pstrcat, apr_psprintf, apr_snprintf, apr_time_now, apr_ctime,
  apr_base64_encode, apr_base64_encode_len, apr_getopt, apr_getopt_init,
  apr_signal, apr_terminate, apr_app_initialize.

  Βασικές λειτουργίες του προγράμματος.

- Συναρτήσεις OpenSSL
  SSL_*, SSL_CTX_*, SSL_get_peer_certificate, SSL_get_peer_cert_chain, X509_*,
  EVP_PKEY_*, BIO_*, OPENSSL_init_ssl, ERR_print_errors, RAND_seed.

  Δείχνουν ότι το πρόγραμμα δημιουργεί, διαχειρίζεται και αλληλεπιδρά με SSL/TLS συνδέσεις.

- Συναρτήσεις Standard C Library

malloc, free, exit, strlen, strcpy, strncmp, strstr, strchr, printf, fprintf, putchar, puts, perror, fopen, fclose, fwrite, fflush, time, rand, qsort, __stach_chk_fail.

Βασικές ρουτίνες κάθε προγράμματος σε C. Η παρουσία της __stack_chk_fail επιβεβαιώνει και πάλι την ύπαρξη stack canary.

Η αξιοποίηση του radare2 επιτρέπει τον εντοπισμό επικίνδυνων ή δυνητικά ευάλωτων συναρτήσεων, όπως οι system, execve, strcpy και gets. Η παρουσία τέτοιων συναρτήσεων σχετίζεται με κατηγορίες όπως OS Command Injection, Use of Inherently Dangerous Function και η Use of Potentially Dangerous Function . Μέσω της στατικής ανάλυσης των imports, το script μπορεί να αναγνωρίσει ενδείξεις που συνδέονται με καταγεγραμμένες κατηγορίες ευπαθειών και επιθέσεων.

Συνολικά, η διαδικασία στατικής ανάλυσης υπήρξε ένα εκφραστικό παράδειγμα του πώς η συνδυασμένη εφαρμογή command line tools μπορεί να συνθέσει μια ολοκληρωμένη και αξιόπιστη περιγραφή ενός δυαδικού αρχείου. Κάθε εργαλείο συμβάλλει με τον δικό του τρόπο εξάγωντας τις δικές τους πληροφορίες, και ο συνδυασμός τους οδηγεί σε ένα τεκμηριωμένο συμπέρασμα, επιδεικνύοντας την αξία της μεθοδολογίας αυτής για την αξιολόγηση ασφαλείας και την κατανόηση του λειτουργικού σκοπού άγνωστων εκτελέσιμων αρχείων.

Επίλογος

Συμπερασματικά, η εκμετάλλευση δυαδικών αρχείων αποτελεί έναν από τους πιο κρίσιμους και σύνθετους τομείς στην ασφάλεια λογισμικού. Η φύση των δυαδικών αρχείων τα καθιστά ιδιαίτερα ευάλωτα σε επιθέσεις χαμηλού επιπέδου. Η ανάλυση δυαδικών αρχείων, τόσο μέσω στατικών όσο και δυναμικών μεθόδων, επιτρέπει την κατανόηση της εσωτερικής λειτουργίας ενός προγράμματος και την έγκαιρη ανίχνευση πιθανών αδυναμιών πριν αυτές αξιοποιηθούν από κακόβουλους χρήστες. Τα εργαλεία reverse engineering και ανάλυσης ευπαθειών παρέχουν πολύτιμα δεδομένα σχετικά με τη δομή, τη ροή εκτέλεσης και τις αλληλεπιδράσεις ενός δυαδικού αρχείου με το λειτουργικό σύστημα και τη μνήμη. Η αποτελεσματική εκμετάλλευση των ευπαθειών στα δυαδικά αρχεία μπορεί να οδηγήσει σε πλήρη παραβίαση συστημάτων, εκτέλεση αυθαίρετου κώδικα και μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα. Ωστόσο, η συνεχής ανάπτυξη τεχνικών ανίχνευσης και η ενίσχυση των μηχανισμών ασφαλείας συνεισφέρουν στη βελτίωση της ανθεκτικότητας των συστημάτων απέναντι σε τέτοιες απειλές. Η κατανόηση των μεθόδων εκμετάλλευσης δυαδικών αρχείων αποτελεί ουσιαστικό βήμα για την ενίσχυση της ασφάλειας λογισμικού και την ανάπτυξη ισχυρότερων μέτρων προστασίας απέναντι σε προηγμένες επιθέσεις. Βασικό πλεονεκτήμα της προτεινόμενης υλοποίησης είναι η δυνατότητα γρήγορης και εύκολης εξαγωγής βασικών πληροφοριών σχετικά με τη δομή ενός δυαδικού αρχείου. Μέσω της συνδυαστικής χρήσης διαφορετικών εργαλείων ανάλυσης και της αυτοματοποίησής τους μέσα από ένα ενιαίο script, ο χρήστης μπορεί να αποκτήσει συνοπτική εικόνα για τα χαρακτηριστικά του εκτελέσιμου αρχείου, όπως τα sections, τις συναρτήσεις, τις εξαρτήσεις από βιβλιοθήκες και τους βασικούς μηχανισμούς ασφαλείας. Η αυτοματοποίηση αυτής της διαδικασίας μειώνει σημαντικά τον χρόνο που απαιτείται για την αρχική διερεύνηση ενός binary, επιτρέποντας στον αναλυτή να συγκεντρώνει γρήγορα χρήσιμες πληροφορίες χωρίς να χρειάζεται να εκτελεί ξεχωριστά κάθε εργαλείο. Σημαντικό μειονέκτημα, ωστόσο, αφορά την αντιστοίχιση των δυναμικών βιβλιοθηκών με γνωστές ευπάθειες μέσω βάσεων δεδομένων CVE. Η διαδικασία αυτή βασίζεται σε αναζήτηση πληροφοριών που σχετίζονται με τα ονόματα των βιβλιοθηκών και δεν πραγματοποιεί πλήρη και αξιόπιστη αξιολόγηση της πραγματικής ευπάθειας του συγκεκριμένου εκτελέσιμου αρχείου. Συνεπώς, τα αποτελέσματα που προκύπτουν από αυτόν τον έλεγχο δεν μπορούν να θεωρηθούν οριστική απόδειξη ύπαρξης ή απουσίας ευπαθειών, αλλά περισσότερο μια ενδεικτική πληροφορία που μπορεί να καθοδηγήσει περαιτέρω ανάλυση. Σε μελλοντική εξέλιξη της παρούσας υλοποίησης, το εργαλείο θα μπορούσε να επεκταθεί με την ενσωμάτωση πρόσθετων τεχνικών δυναμικής ανάλυσης. Η διασύνδεση με εργαλεία δυναμικής ανάλυσης θα μπορούσε να επιτρέψει μια πιο βαθιά διερεύνηση της συμπεριφοράς του δυαδικού αρχείου κατά την εκτέλεσή του. Με αυτόν τον τρόπο, η αξιολόγηση των πιθανών ευπαθειών θα μπορούσε να γίνει πιο ουσιαστική και αξιόπιστη, μετατρέποντας το εργαλείο από ένα σύστημα που παρέχει κυρίως πληροφορίες για την αρχιτεκτονική και τη δομή του binary σε ένα πιο ολοκληρωμένο σύστημα ανάλυσης ασφάλειας.

Πίνακας Ορολογίας

Δυαδικό αρχείο	binary
Εκμετάλλευση δυαδικών	Binary exploitation
Εκτελέσιμα αρχεία	Executables
Στοίβα	Stack
Σωρός	Heap
Αντίστροφη μηχανική	Reverse engineering
Κύκλος Ανάπτυξης Λογισμικού	Software Development Life Cycle
Δυναμική ανάλυση	Dynamic Analysis
Στατική ανάλυση	Static Analysis
Μεταδεδομένα	Metadata
Ακολουθία χαρακτήρων	Strings
Τυχαία Διάταξη Χώρου Διευθύνσεων	Address Space Layout Randomization
Υπερχείλιση buffer	Buffer overflow

Πίνακας συντμήσεων-αρτικόλεξων-ακρονύμιων

ICS	Industrial Control System
GUI	Grafical User Interface
BLOB	Binary Large Object
SDLC	Software Development Life Cycle
IAST	interactive application security testing
DAST	dynamic application security testing
ELF	Executable and Linkable Format
ASLR	Address Space Layout Randomization
CTF	Capture The Flag
ROP	Return-Oriented Programming
DoS	Denial of Service
DEP	Data Execution Prevention
UAF	Use after free
CWE	Common Weakness Enumerations
CAPEC	Common Attack Pattern Enumeration and Classification
CVE	Common Vulnerabilities and Exposures

Βιβλιογραφικές πηγές

1. Shoshitaishvili, Y., Wang, R., Salls (2016). A Next-Generation Binary Analysis Platform. In Proceedings of the Annual Computer Security Applications Conference (ACSAC).

2. Cybersecurity and Infrastructure Security Agency (CISA). (2022). Vulnerability Identification in Binary Files. ICSJWG.

3. InfoSec Institute. (2023). Binary Exploitation Techniques. Infosec Resources.

4. Brumley, D., Jager, I., Avgerinos, T., & Schwartz, E. J. (2011). BAP: A Binary Analysis Platform. In Proceedings of the International Conference on Computer Aided Verification (CAV).

5. Meng, X., Miller, B. P., & Jun, K. (2019). Osprey: A Practical Binary Analysis Framework for Full-System Reversible Debugging. In Proceedings of the 2019 ACM SIGPLAN International Symposium on Memory Management.

6. Rouse, M. (2023). What is a Binary File?.

7. Chess, B., & McGraw, G. (2004). A Survey of Static Analysis Methods for Identifying Security Vulnerabilities in Software Systems. IBM Systems Journal.

8. BugProve. (2023). Binary Analysis Fundamentals..

9. StackZero. (2023). Python String Analysis for Binary Exploitation..

10. Cobalt.io. (2022). A Pentester's Guide to Exploiting Buffer Overflow Vulnerabilities.

11. Cymulate. (2023). Binary Exploitation: Understanding the Basics.

12. GeeksforGeeks. (2023). objdump command in Linux with Examples.

13. Mosse Cyber Security Institute. (2022). The Strings Tool: Extracting Text for Digital Forensics.

14. Radare2. In Wikipedia (2023).

15. The MITRE Corporation. (2023). About CWE. Common Weakness Enumeration.

16. The MITRE Corporation. (2023). About CAPEC. Common Attack Pattern Enumeration and Classification.

17. Fortinet. (2023). What is a CVE?

18. IBM. (2023). Binary Strings.

19. Shin, E. C. R., Williams, L., & Song, D. (2015). Recognizing Functions in Binaries with Neural Networks.

20. Brumley, D., & Song, D. (2011). United States Patent and Trademark Office.

21. slimm609. (2018). checksec.

22. Varonis. (2023). How to Use Ghidra: Reverse Engineering Using Ghidra.

23. Software Advice. (2023). IDA Pro Reviews & Ratings.

24. Fr3ak. (2021). Analysing and Extracting Firmware Using Binwalk.

25. GeeksforGeeks. (2023). How to Use Flawfinder Python Tool to Find Vulnerabilities in C/C++ Code.

26. The angr Project. (2023). A platform-agnostic binary analysis framework.

27. Veracode Community. (2023). How does static analysis work? What type of internal modeling is performed?

28. BugProve. (2023). Basics of Binary Analysis.

29. Palo Alto Networks. (2023). Why You Need Static Analysis, Dynamic Analysis & Machine Learning.

30. vFunction. (2023). Static vs. Dynamic Code Analysis: Key Differences.