Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Τηλεπικοινωνιών
Τμήμα Ψηφιακών Συστημάτων

Μεταπτυχιακό Πρόγραμμα Σπουδών
Ασφάλεια Ψηφιακών Συστημάτων

Διπλωματική Εργασία

Ανάπτυξη προκλήσεων στυλ Jeopardy CTF

(Development of Jeopardy CTF style challenges)

Επιβλέπον Καθηγητής: Ξενάκης Χρήστος

Ονοματεπώνυμο	E-mail	A.M.
Μαρίνος Ξυνής	marinos.xinis@ssl-unipi.gr	MTE2217

Πειραιάς 2024

Περίληψη

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη Capture the Flag προκλήσεων (challenges) τύπου Jeopardy. Τα challenges έχουν δημιουργηθεί με στόχο την ανάδειξη θεμάτων τα οποία αφορούν τις περιοχές της κρυπτογραφίας, της αντίστροφης μηχανικής, της στεγανογραφίας και της εκμετάλλευσης ευπαθειών διαδικτυακών εφαρμογών. Επιπλέον, έχουμε αναπτύξει μια web εφαρμογή η οποία προσομοιάζει ένα περιβάλλον διαγωνισμού CTF και η οποία αποτελεί την κύρια διεπαφή με την οποία οι χρήστες μπορούν να αποκτήσουν πρόσβαση στα εν λόγω challenges. Η υλοποίηση έχει γίνει σε ένα εικονικό μηχάνημα (VM).

Abstract

This thesis focuses on developing Jeopardy-style Capture the Flag (CTF) challenges. These challenges have been designed to highlight topics related to cryptography, reverse engineering, steganography, and the exploitation of vulnerabilities in web applications. Additionally, we have developed a web application that simulates a CTF competition environment and serves as the main interface through which users can access these challenges. The implementation was done on a virtual machine (VM).

Περιεχόμενα

Πίνακας Εικόνων

Εισαγωγή

Στην σύγχρονη ψηφιακή εποχή όπου οι τεχνολογίες αναπτύσσονται και εξελίσσονται με γοργούς ρυθμούς είναι γεγονός πως η κυβερνοασφάλεια αποτελεί, πλέον, έναν από τους σημαντικότερους παράγοντες που πρέπει να ληφθεί υπόψη για την εύρυθμη λειτουργία ενός ψηφιακού οικοσυστήματος. Καθώς ο αριθμός των απειλών και η ανάγκη για αποτελεσματικά μέτρα προστασίας στον κυβερνοχώρο ολοένα και αυξάνεται, οι διαγωνισμοί Capture The Flag (CTF) αναδεικνύονται ως ένας ευχάριστος και αποτελεσματικός τρόπος για την εκπαίδευση των επαγγελματιών που δραστηριοποιούνται στον τομέα της κυβερνοασφάλειας. Η συγκεκριμένη διπλωματική εργασία επικεντρώνεται στη δημιουργία Jeopardy-style CTF challenges που θα ενθαρρύνουν τους συμμετέχοντες να εφαρμόσουν και να εμπλουτίσουν τις γνώσεις τους σε διάφορους τομείς της κυβερνοασφάλειας. Επιπλέον, μέσω αυτών των προκλήσεων, οι συμμετέχοντες θα έχουν την ευκαιρία να ασκήσουν τις δεξιότητές τους σε ένα περιβάλλον που προσομοιώνει πιθανές επιθέσεις και ευπάθειες των συστημάτων πληροφορικής. Οι προκλήσεις έχουν δημιουργηθεί καλύπτουν μια ευρεία γκάμα θεμάτων, όπως ευπάθειες διαδικτυακών εφαρμογών (web exploitation), κρυπτογραφία, αντίστροφη μηχανική και στεγανογραφία.

Διαγωνισμός Capture the Flag (CTF)

Οι διαγωνισμοί Capture the Flag (CTF) αποτελούν ένα δημοφιλές μέσο εκπαίδευσης στο χώρο της κυβερνοασφάλειας. Στους διαγωνισμούς αυτού του είδους ο κάθε διαγωνιζόμενος, είτε ατομικά είτε ως μέλος μιας ομάδας, καλείται να δώσει λύση σε πρακτικά προβλήματα (challenges) ενισχύοντας, με αυτόν τον τρόπο, τις δεξιότητές του στην κυβερνοασφάλεια. Τα challenges περιλαμβάνουν σενάρια με θεματικές που εκτείνονται από την εκμετάλλευση ευπαθειών ιστοσελίδων και αποκωδικοποίηση κωδικών πρόσβασης έως την παραβίαση δικτύων. Στόχος του διαγωνιζόμενου είναι η εύρεση μιας συμβολοσειράς, γνωστή ως σημαία (flag), την οποία και πρέπει να υποβάλλει για να αποδείξει πως έλυσε το challenge. Οι διαγωνιζόμενοι μπορούν να δημοσιεύσουν τις λύσεις τους online μοιράζοντας, με αυτόν τον τρόπο, τις γνώσεις τους με τους υπόλοιπους. Οι λύσεις (writeups), μπορούν να χρησιμοποιηθούν σαν εκπαιδευτικό υλικό στο οποίο περιγράφεται η μεθοδολογία επίλυσης του εκάστοτε challenge, αλλά και σαν έμπνευση για τους δημιουργούς μελλοντικών CTF διαγωνισμών [1].

Κατηγορίες CTF

Jeopardy CTF

Η Jeopardy αποτελεί τη δημοφιλέστερη κατηγορία CTF διαγωνισμού. Στη συγκεκριμένη κατηγορία οι συμμετέχοντες επιδιώκουν την επίλυση όσο το δυνατόν περισσότερων challenges σε περιορισμένο χρονικό διάστημα. Η επιτυχής ολοκλήρωση του κάθε challenge προσφέρει πόντους οι οποίοι έχουν να κάνουν με τον βαθμό δυσκολίας του κάθε challenge. Τα challenges ομαδοποιούνται σε κατηγορίες και δεν υπάρχει κάποιος περιορισμός ως προς τη σειρά επίλυσης τους. Το τελευταίο παρέχει τη δυνατότητα στους διαγωνιζόμενους να επικεντρωθούν σε εκείνα τα challenges τα οποία θεωρούν πιο διασκεδαστικά αλλά και που ανταποκρίνονται στις ικανότητές τους. Τα challenges που συναντώνται σε Jeopardy CTFs αφορούν θέματα που σχετίζονται

με διάφορες κατηγορίες. Η επόμενη ενότητα παρουσιάζει μερικές από τις κατηγορίες αυτές [2].

Κατηγορίες Jeopardy CTF

Reverse Engineering

Σε αυτήν την κατηγορία παρέχεται, τις περισσότερες φορές, ένα εκτελέσιμο αρχείο στο οποίο βρίσκεται κρυμμένη η σημαία (flag) την οποία κατασκευάζει ένας αλγόριθμος. Ο διαγωνιζόμενος, κάνοντας χρήση εργαλείων, όπως decompilers και disassemblers, καλείται να κατανοήσει την λογική με την οποία εκτελείται ο αλγόριθμος προκειμένου να ανακτήσει το flag.

Binary Exploitation

Σε αυτή την κατηγορία, παρέχεται ένα ευάλωτο εκτελέσιμο αρχείο και πληροφορίες ενός απομακρυσμένου server. Σε πρώτη φάση ο διαγωνιζόμενος πρέπει να αναλύσει το εκτελέσιμο και να εντοπίσει τυχόν ευπάθειες που περιέχονται στον κώδικά του (buffer overflows, use after-free κ.ο.κ). Στη συνέχεια, εκμεταλλευόμενος τις ευπάθειες που ανακάλυψε, καλείται να κατασκευάσει exploits τα οποία θα τον βοηθήσουν να ανακτήσει το flag από τον server.

Web Exploitation

Σε αυτή την κατηγορία, δίνεται η διεύθυνση IP ενός απομακρυσμένου server στον οποίο εκτελείται κάποια ευάλωτη web εφαρμογή. Σε αντίθεση με τις δύο προηγούμενες, η συγκεκριμένη κατηγορία δεν παρέχει κάποιον κώδικα προς εξέταση. Γι αυτό, ο διαγωνιζόμενος θα πρέπει να εφαρμόσει τεχνικές ανάλυσης μαύρου κουτιού οι οποίες θα τον οδηγήσουν στην εύρεση ευπαθειών όπως SQL injection, XSS, command injection κ.α.

Digital Forensics

Τα challenges που ανήκουν σε αυτή την κατηγορία έχουν το flag κρυμμένο σε κάποιο από τα δοσμένα αρχεία προς εξέταση. Το είδος αυτών των αρχείων ποικίλει από αρχεία καταγραφής πακέτων δικτύου (network packets), αρχεία εγγράφων (document files) μέχρι αντίγραφα δίσκου συσκευής (disk images). Η γνώση του συστήματος αρχείων αλλά και βασικών εργαλείων digital forensics είναι απαραίτητη προκειμένου να βρεθεί το flag.

Cryptography

Τα προβλήματα της συγκεκριμένης κατηγορίας, συνήθως, περιέχουν το flag σε κρυπτογραφημένη μορφή. Στόχος είναι η αποκρυπτογράφηση του η οποία επιτυγχάνεται λαμβάνοντας υπόψη τυχόν κρυπτογραφικές ευπάθειες που παρουσιάζει το πρωτόκολλο - αλγόριθμος που έχει χρησιμοποιηθεί για την κρυπτογράφηση.

Attack and Defence

Στη συγκεκριμένη κατηγορία, οι συμμετέχοντες διαγωνίζονται σε ομάδες. Κάθε ομάδα έχει στη διάθεσή της έναν ή περισσότερους servers οι οποίοι παρουσιάζουν κενά ασφαλείας και στους οποίους βρίσκονται κρυμμένα τα flags. Ο ρόλος της κάθε ομάδας είναι τόσο αμυντικός (blue team) όσο και επιθετικός (red team) . Η blue team έχει ως στόχο να προστατεύσει, όσο το δυνατόν αποτελεσματικότερα, τους server της εντοπίζοντας και διορθώνοντας (patching) τυχόν ευπάθειες που παρουσιάζονται. Αποτρέποντας, έτσι, τις επιθέσεις από αντίπαλες ομάδες που έχουν ως στόχο την απόκτηση των κρυμμένων flags. Από την άλλη πλευρά, η red team επιτίθεται στα συστήματα των αντίπαλων ομάδων αναζητώντας ευπάθειες που θα οδηγήσουν στην παραβίαση των servers και στην ανάκτηση των flags.

King of the Hill

Το King of the Hill αποτελεί κατηγορία CTF η οποία παρουσιάζει αρκετές ομοιότητες με την Attack and Defense CTF κατηγορία. Το King of The Hill διεξάγεται σε κλειστό ελεγχόμενο περιβάλλον όπου κάθε ομάδα πρέπει να επιτεθεί στις υπόλοιπες ομάδες και να επιδιορθώσει ευπάθειες που παρουσιάζουν ορισμένα services. Η βασική ιδέα του King of the Hill είναι η διατήρηση του ελέγχου του συστήματος για όσο το δυνατόν περισσότερο χρονικό διάστημα. Κάθε ομάδα έχει στη διάθεσή της περιορισμένο αριθμό υπηρεσιών (services) και όλες οι ομάδες μπορούν να έχουν πρόσβαση σε κάθε υπηρεσία για να ανακαλύψουν τα τρωτά σημεία τους. Αφού αποκτήσουν τον έλεγχο μιας υπηρεσίας μπορούν να επιχειρήσουν να την επιδιορθώσουν εμποδίζοντας άλλες ομάδες να λάβουν τον έλεγχο. Όσο περισσότερο μια ομάδα διατηρεί τον έλεγχο της υπηρεσίας, τόσο υψηλότερη βαθμολογία λαμβάνει. Αντίθετα με το Attack and Defence CTF , όπου κάθε ομάδα που καταφέρνει να πάρει τον έλεγχο μιας υπηρεσίας μπορεί να κερδίσει σκορ, μόνο η ομάδα που διατηρεί τον έλεγχο μπορεί να λάβει τη βαθμολογία [3].

Περιγραφή JeopardyCTF Web App

Τεχνολογίες Υλοποίησης

Γλώσσα Προγραμματισμού C

Η γλώσσα προγραμματισμού C δημιουργήθηκε από τον Dennis Ritchie τη δεκαετία του 1970. Χρησιμοποιήθηκε για πρώτη φορά για την ανακατασκευή του λειτουργικού συστήματος Unix, το οποίο, αρχικά, είχε αναπτυχθεί στη γλώσσα προγραμματισμού assembly. Σήμερα, η C χρησιμοποιείται στον προγραμματισμό εφαρμογών χαμηλού επιπέδου κυρίως σε περιβάλλοντα όπου δίνεται έμφαση στην ταχύτητα και στον έλεγχο (λειτουργικά συστήματα, ενσωματωμένα συστήματα κ.α). Επειδή η συγκεκριμένη η γλώσσα μεταγλωτιζεται απευθείας σε assembly και δεν διαθέτει, σχεδόν, κανένα μηχανισμό ασφαλείας αποτελεί την κύρια επιλογή για την κατασκευή reverse engineering και binary exploitation challenges.

HTML/CSS/JAVASCRIPT

Οι HTML, CSS και Javascript αποτελούν τις βασικότερες τεχνολογίες που χρησιμοποιούνται στην κατασκευή web εφαρμογών και ιστοσελίδων. Η HTML (Hyper-Text Markup Language) και τα CSS (Cascading Style Sheets) ευθύνονται για τον τρόπο με τον οποίο παρουσιάζεται η ιστοσελίδα στον χρήστη. Συγκεκριμένα, η HTML χρησιμοποιείται για την περιγραφή των ιδιοτήτων των στοιχείων που απαρτίζουν μια ιστοσελίδα και τα CSS χρησιμοποιούνται για την προσαρμογή της εμφάνισης της ιστοσελίδας. Τέλος, η Javascript αποτελεί μια γλώσσα προγραμματισμού με τη οποία επιτυγχάνεται η λειτουργικότητα και η διαδραστικότητα που έχει μια ιστοσελίδα.

LAMP Stack

Η στοίβα LAMP [4] αποτελείται από ένα σύνολο τεχνολογιών ανοικτού κώδικα οι οποίες παρέχονται δωρεάν και χρησιμοποιούνται ευρέως από τους προγραμματιστές ιστοσελίδων και web εφαρμογών. Αποτελεί ακρωνύμιο για το λειτουργικό σύστημα Linux, τον Apache web server, την MySQL και τη γλώσσα προγραμματισμού PHP. Καθίσταται ιδιαίτερα αποδοτικό επιταχύνοντας την ανάπτυξη με δοκιμασμένες λύσεις. Επιπλέον, παρέχει ευκολία στη συντήρηση μέσω των παγκόσμιων συνεισφορών στον κώδικα. Η ισχυρή υποστήριξη από τη μεγάλη κοινότητα των επαγγελματιών πληροφορικής εξασφαλίζει άριστη υποστήριξη, ενώ η ευελιξία του επιτρέπει στους προγραμματιστές να το προσαρμόσουν σύμφωνα με τις ανάγκες τους.

Linux OS

Το Linux είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα το οποίο μπορεί να εγκατασταθεί και να διαμορφωθεί ώστε να ανταποκρίνεται σε διαφορετικές απαιτήσεις εφαρμογών. Το λειτουργικό σύστημα Linux βρίσκεται στο πρώτο επίπεδο της στοίβας LAMP και είναι υπεύθυνο για την υποστήριξη άλλα στοιχείων τα οποία βρίσκονται στα ανώτερα επίπεδα της LAMP στοίβας.

Apache Web Server

Ο Apache [5] είναι ένας web server ανοιχτού κώδικα και αποτελεί το δεύτερο επίπεδο της στοίβας LAMP. Η μονάδα Apache επιτελεί λειτουργίες που σχετίζονται με την αποθήκευση αρχείων ιστότοπου και την ανταλλαγή πληροφοριών με τον browser κάνοντας χρήση του διαδικτυακού πρωτοκόλλου HTTP.

MySQL

Η MySQL [6] αποτελεί ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα και είναι το τρίτο επίπεδο της στοίβας LAMP. Το μοντέλο

LAMP χρησιμοποιεί MySQL για αποθήκευση, διαχείριση και αναζήτηση πληροφοριών σε σχεσιακές βάσεις δεδομένων.

PHP

Η PHP [7] (Hypertext Preprocessor) αποτελεί το τέταρτο και τελευταίο επίπεδο της στοίβας LAMP. Είναι μια γλώσσα scripting που επιτρέπει τη δημιουργία ιστοτόπων με δυναμικό περιεχόμενο. Η χρήση της PHP επιτρέπει στον web server, τη βάση δεδομένων και το λειτουργικό σύστημα να επεξεργάζονται με συνοχή αιτήματα τα οποία προέρχονται από τον browser.

Virtual Machines

Ένα εικονικό μηχάνημα (VM) παρέχει ένα περιβάλλον λειτουργίας απομονωμένο από το σύστημα του κεντρικού υπολογιστή (host) και από άλλα εικονικά μηχανήματα. Το λειτουργικό σύστημα του VM (guest) μπορεί να διαφέρει από αυτό του host υπολογιστή και των άλλων εικονικών μηχανημάτων. Ένας υπολογιστής μπορεί να φιλοξενήσει πολλά VM στα οποία εκτελούνται διάφορα λειτουργικά συστήματα χωρίς να αλληλεπιδρά με αυτά. Αν και τα VM εξαρτώνται από τους φυσικούς πόρους του host υπολογιστή, οι πόροι αυτοί είναι εικονικοποιημένοι (virtualized) και διανέμονται ανάλογα με τις ανάγκες. Αυτό επιτρέπει την ταυτόχρονη εκτέλεση διαφορετικών λειτουργικών συστημάτων και τη δυνατότητα προσαρμογής του φόρτου εργασίας. Από την άποψη του χρήστη, το VM λειτουργεί ως ανεξάρτητο σύστημα. Οι χρήστες μπορούν να διαμορφώσουν και να ενημερώσουν το λειτουργικό σύστημα και τις εφαρμογές τους χωρίς επιπτώσεις στον host υπολογιστή ή άλλα VMs. Επίσης, παρέχουν ένα επιπλέον επίπεδο προστασίας όσον αφορά επιθέσεις στον κυβερνοχώρο υποστηρίζοντας λειτουργίες όπως λήψη στιγμιότυπων και αντιγράφων ασφαλείας. Με αυτόν τον τρόπο παρέχεται η δυνατότητα στους διαχειριστές να διαγράψουν ή να επαναφέρουν ένα παραβιασμένο VM σε ένα πρόσφατο αντίγραφο ασφαλείας ή στιγμιότυπο. Επειδή το παραβιασμένο VM είναι απομονωμένο από τον host υπολογιστή και

άλλα εικονικά μηχανήματα, η απειλή περιορίζεται σε ένα μόνο συγκεκριμένο VM.

Iptables

Το iptables [8] είναι ένα τυπικό open source τείχος προστασίας (firewall) το οποίο διατίθεται προ εγκατεστημένο στις περισσότερες διανομές των Linux. Αποτελεί μια διεπαφή γραμμής εντολών με την οποία μπορεί κανείς να διαχειρίζεται τη στοίβα δικτύου του Linux σε επίπεδο πυρήνα (kernel level). Η λειτουργία του βασίζεται στην αντιστοίχιση ορισμένων χαρακτηριστικών δικτυακών πακέτων με ένα σύνολο κανόνων οι οποίοι καθορίζουν τον τρόπο διαχείρισης του εκάστοτε πακέτου στο δίκτυο. Υπάρχουν αρκετές επιλογές για να καθοριστούν ποια πακέτα ταιριάζουν με έναν συγκεκριμένο κανόνα. Μερικές από αυτές μπορεί να είναι η αντιστοίχιση του τύπου πρωτοκόλλου πακέτου, η διεύθυνση ή η θύρα προέλευσης ή προορισμού, η διεπαφή δικτύου που χρησιμοποιείται ή η σχέση ενός πακέτου με προηγούμενα πακέτα. Γενικά, το iptables μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς. Ορισμένες κύριες χρήσεις περιλαμβάνουν:

- Φιλτράρισμα Πακέτων: Έλεγχος εισερχόμενης και εξερχόμενης κίνηση προς και από το σύστημα μας. Προσδιορισμός των τύπων πακέτων που επιτρέπονται ή απορρίπτονται βάσει διευθύνσεων IP, θυρών, πρωτοκόλλων και άλλων παραμέτρων.
- Μετάφραση Διευθύνσεων Δικτύου (NAT): Το iptables μπορεί, επίσης, να χρησιμοποιηθεί για τη μετάφραση διευθύνσεων IP και θυρών. Αυτό είναι χρήσιμο για την απόκρυψη των πραγματικών διευθύνσεων IP των υπηρεσιών που βρίσκονται πίσω από έναν δρομολογητή, ή για την προώθηση θυρών (port forwarding).
- Διαχείριση Κίνησης Δικτύου: Το iptables μπορεί να χρησιμοποιηθεί για τον περιορισμό του ρυθμού μετάδοσης των πακέτων ή τον περιορισμό εύρους ζώνης συγκεκριμένων θυρών.
- Περιορισμός Πρόσβασης: Το iptables μπορεί να χρησιμοποιηθεί με στόχο τον έλεγχο της πρόσβασης είτε επιτρέποντας είτε αποκλείοντας την

πρόσβαση σε συγκεκριμένες υπηρεσίες ή πόρους από συγκεκριμένες διευθύνσεις IP.

Inotifywait

Το inotifywait [9] αποτελεί εργαλείο του Linux το οποίο επιτρέπει στους χρήστες να παρακολουθούν αλλαγές σε αρχεία και φακέλους σε πραγματικό χρόνο χρησιμοποιώντας τη διεπαφή inotify του πυρήνα του συστήματος. Όταν ορίζεται να παρακολουθεί έναν κατάλογο ή ένα αρχείο, το inotifywait παραμένει ενεργό και αναμένει για αλλαγές όπως πρόσβαση, προσθήκη, αφαίρεση, τροποποίηση αρχείου κ.α. Όταν ανιχνεύσει κάποιου είδους ενέργεια, όπως τις προηγούμενες, το inotifywait εκτελεί κάποια ενέργεια που έχει καθοριστεί από το χρήστη, όπως εκτέλεση ενός script ή εμφάνιση ενός μηνύματος. Μπορεί να χρησιμοποιηθεί για να ανιχνεύσει αλλαγές σε αρχεία και φακέλους που μπορεί να είναι χρήσιμες σε διάφορες εφαρμογές, όπως η αυτοματοποίηση διαδικασιών, η παρακολούθηση του συστήματος αρχείων ή η ανίχνευση δραστηριότητας χρηστών.

JeopardyCTF Server

Για τις ανάγκες της παρούσας εργασίας δημιουργήσαμε έναν web server στον οποίο δώσαμε το όνομα“JeopardyCTF” server. Ο διακομιστής “JeopardyCTF”, ο οποίος έχει υλοποιηθεί με χρήση των τεχνολογιών LAMP (Linux, Apache, MySQL, PHP) stack, φιλοξενεί τα challenges τύπου Jeopardy CTF και τρέχει σε virtual machine το οποίο διαθέτει την έκδοση Ubuntu 22.04.1 LTS . Οι συμμετέχοντες μπορούν να κατεβάσουν τον διακομιστή και να λύσουν τα challenges σε ατομικό αλλά και σε ανταγωνιστικό επίπεδο με άλλους παίκτες οι οποίοι ανήκουν στο ίδιο δίκτυο .

Σε περίπτωση που επιλέξουμε να χρησιμοποιήσουμε τον JeopardyCTF server για διαγωνισμό, παρέχονται μηχανισμοί οι οποίοι αποτρέπουν έναν διαγωνιζόμενο από το να φέρει εις πέρας τον διαγωνισμό με τρόπους που αντιβαίνουν τους κανονισμούς. Αυτοί οι μηχανισμοί έχουν υλοποιηθεί με τη χρήση του iptables και της λειτουργίας του inotifywait του Linux. Σε πρώτο επίπεδο, ο server διαθέτει ένα αρχείο με (βασικούς) iptables κανόνες οι οποίοι αποτρέπουν ενέργειες όπως την απομακρυσμένη είσοδο στην βάση δεδομένων της εφαρμογής και την απαγόρευση εισόδου ευαίσθητα αρχεία.

Σε δεύτερο επίπεδο, έχουμε δημιουργήσει μηχανισμό ο οποίος ελέγχει αν κάποιος χρήστης έχει αποκτήσει πρόσβαση στον φάκελο που βρίσκονται αποθηκευμένα κρίσιμα δεδομένα. Σε περίπτωση που ανιχνευθεί κάποια απάτη, ο υπεύθυνος του Jeopardy CTF event ενημερώνεται από το inotifywait και ο κακόβουλος χρήστης αποκλείεται από το δίκτυο χρησιμοποιώντας τους κανόνες του iptables. Σημαντικό είναι να τονίσουμε, ο υπεύθυνος του γεγονότος πρέπει να ενεργοποιήσει αυτούς τους μηχανισμούς πριν την έναρξη του διαγωνισμού.

Τελος, έχουμε κάνει χρήση κανόνων αρχείων .htaccess [10] προκειμένου να απαγορεύσουμε στους χρήστες να αποκτήσουν πρόσβαση, μέσω browser, σε directories και αρχεία της εφαρμογής πληκτρολογώντας το path που βρίσκονται αποθηκευμένα στον server.

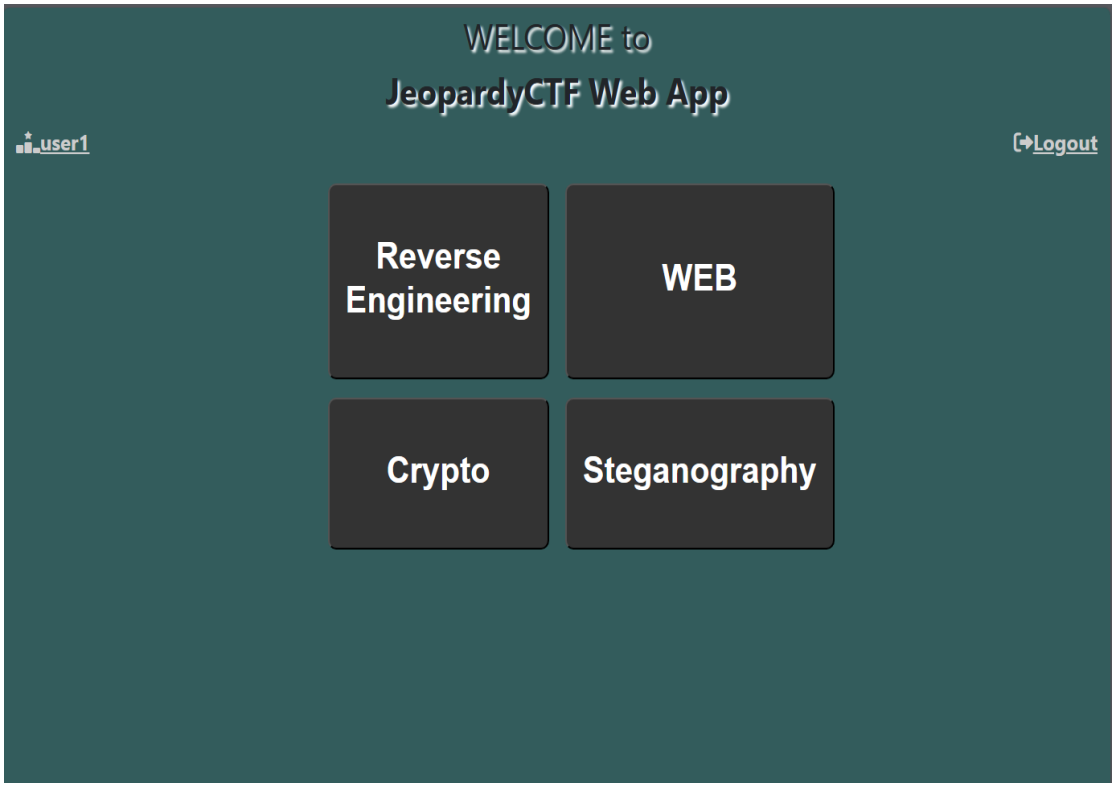
JeopardyCTF Web App Interface

Προκειμένου να προσφέρουμε στον χρήστη - διαγωνιζόμενο μια ολοκληρωμένη εμπειρία CTF διαγωνισμού, δημιουργήσαμε ένα web application, το οποίο ονομάσαμε “JeopardyCTF Web App”, με το οποίο η αλληλεπίδραση χρήστη και εφαρμογής επιτυγχάνεται όσο το δυνατόν πιο ομαλό και φιλικό προς τον χρήστη τρόπο.

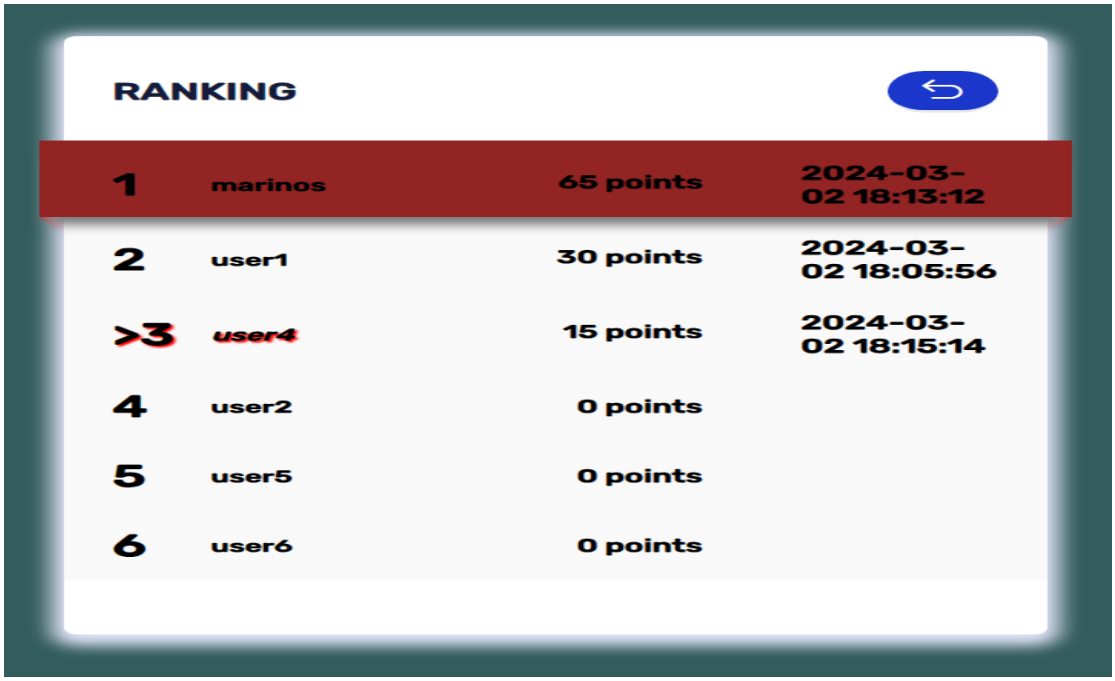
Ο διαγωνιζόμενος αρκεί να εισάγει στον browser την διεύθυνση IP του “JeopardyCTF” server, στον οποίο “τρέχει” η web εφαρμογή, προκειμένου να οδηγηθεί στην αρχική σελίδα όπου του ζητείται να συμπληρώσει το username και το password για να αποκτήσει πρόσβαση στα challenges. Σε περίπτωση που ο διαγωνιζόμενος δεν έχει εγγραφεί μπορεί, κάνοντας κλικ στην αντίστοιχη επιλογή, να δημιουργήσει έναν λογαριασμό.

Εικόνα 1: Login σελίδα JeopardyCTF Web App

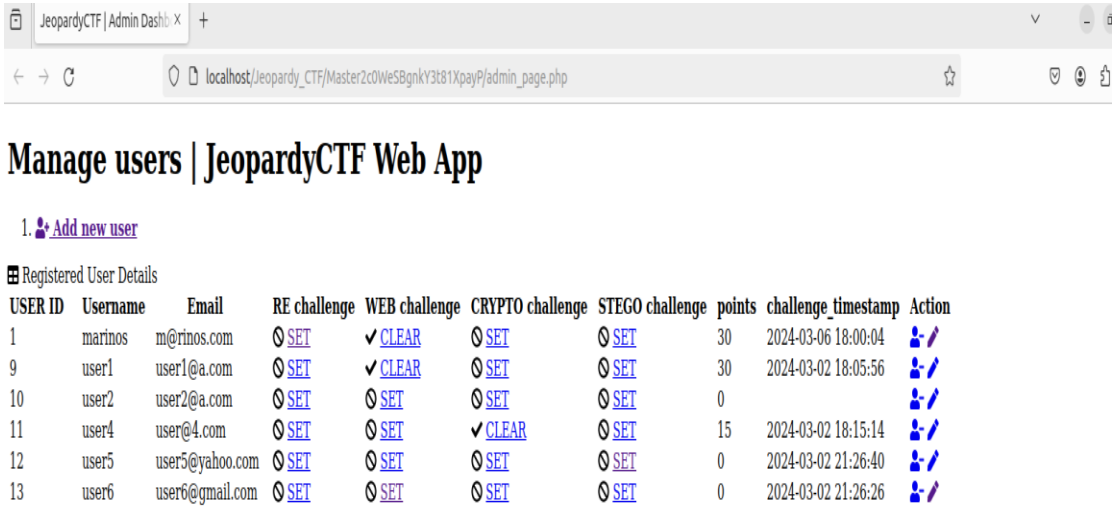
Η κεντρική σελίδα της web εφαρμογής αποτελείται από έναν πίνακα στον οποίο αναγράφεται το είδος του κάθε challenge. Ο διαγωνιζόμενος κάνοντας κλικ στην κατηγορία που τον ενδιαφέρει αποκτά πρόσβαση στην εκφώνηση του challenge που επιθυμεί να λύσει. Η εφαρμογή παρέχει στον διαγωνιζόμενο τη δυνατότητα παρακολούθησης της κατάταξής του σε περίπτωση που ανταγωνίζεται με άλλους διαγωνιζόμενους. Έτσι, κάνοντας κλικ στο username (πάνω αριστερά) εμφανίζεται ο πίνακας κατάταξης στον οποίο αναγράφονται τα username των χρηστών, οι συνολικοί πόντοι που έχει συγκεντρώσει ο κάθε χρήστης μαζί με ένα timestamp το οποίο αναγράφει την χρονική στιγμή που επίλυσης του τελευταίου challenge που επέλεξε ο εκάστοτε χρήστης. Τέλος, παρέχεται ξεχωριστή εφαρμογή (Admin Dashboard) με την οποία ο administrator του event μπορεί να επιτελεί διάφορες λειτουργίες όπως την προσθήκη και αφαίρεση ενός χρήστη, το μαρκάρισμα ενός challenge ως μη ολοκληρωμένο ή ολοκληρωμένο καθώς και την παρακολούθηση της πορείας συνολικά των χρηστών. Να τονίσουμε πως η πρόσβαση στον Admin Dashboard επιτυγχάνεται μόνο από το (εικονικό) μηχάνημα που τρέχει ο JeopardyCTF server.



Εικόνα 2: Κεντρική σελίδα JeopardyCTF Web App



Εικόνα 3: Πίνακας κατάταξης JeopardyCTF Web App



Εικόνα 4: Administrator panel JeopardyCTF Web App

Περιγραφή Challenges

Fortunate Sam (Reverse Engineering Challenge)

Το πρώτο, από τα τέσσερα συνολικά challenges που θα περιγράψουμε, έχει τη μορφή ενός παιχνιδιού τύχης το οποίο έχει υλοποιηθεί σε γλώσσα C. Το challenge αποτελείται, συνολικά, από δύο επίπεδα. Το κάθε επίπεδο ξεκινάει με την υποβολή μιας ερώτηση για την οποία ο παίκτης πρέπει να μαντέψει τη σωστή απάντηση. Με κάθε σωστή απάντηση, ο παίκτης, μεταβαίνει στο επόμενο επίπεδο και ανταμείβεται με ένα hint το οποίο θα τον βοηθήσει να ανακαλύψει το flag και να ολοκληρώσει το challenge.

Η δυσκολία του συγκεκριμένου challenge βασίζεται, ως επί τω πλείστον, στην χρήση μεθόδων obfuscation - απόκρυψης στοιχείων και σε μικρότερο βαθμό στην χρήση μηχανισμών anti-tracing (π.χ ανίχνευση μηχανισμών debugging). Συνδυασμός τεχνικών και μηχανισμών τέτοιου είδους χρησιμοποιούνται συχνά και από προγραμματιστές κακόβουλου λογισμικού προκειμένου, αν όχι να αποτρέψουν, να επιβραδύνουν σημαντικά τη διαδικασία ανάλυσης και κατανόησης του κώδικα. Μερικές από τις τεχνικές που χρησιμοποιήσαμε στο challenge είναι:

- Αναγνώριση της ptrace [11]. Στη συγκεκριμένη Linux system call βασίζεται η λειτουργία πολλών tracing εργαλείων (όπως ltrace και gdb debugger).
- Δυναμική κατασκευή strings που περιέχουν σημαντικά μηνύματα (hints) με στόχο την απόκρυψή τους από εργαλεία static analysis (π.χ strings εντολή).
- Απόκρυψη σημαντικών strings εφαρμόζοντας την λογική πράξη xor.
- Δυναμική κλίση συναρτήσεων που παρουσιάζουν ιδιαίτερο ενδιαφέρον (π.χ strncmp, ptrace) με στόχο την απόκρυψή τους από εργαλεία static analysis.
- Αφαίρεση debugging πληροφοριών όπως ονόματα συναρτήσεων και μεταβλητών. Τέτοιου είδους εκτελέσιμα ονομάζονται stripped**.**

Site Compromised (Web Exploitation Challenge)

Στο δεύτερο, σε σειρά, challenge που θα περιγράψουμε, παρέχεται ο σύνδεσμος url ο οποίος ανακατευθύνει τον διαγωνιζόμενο σε μια ιστοσελίδα που φιλοξενείται στον vulnerable JeopardyCTF server. Η ιστοσελίδα, της οποίας το frontend κομμάτι έχει υλοποιηθεί με HTML/CSS και Javascript, περιέχει διάφορα μηνύματα, καθώς και μηχανισμούς που επιτρέπουν στον χρήστη να αλληλεπιδρά με αυτή. Συγκεκριμένα, υπάρχουν δύο μηχανισμοί αλληλεπίδρασης· ένα κουμπί υπεύθυνο για την εμφάνιση ενός βοηθητικού μηνύματος (hint) και ένας μηχανισμός με τον οποίο επιτυγχάνεται το "ανέβασμα" και η αποθήκευση αρχείων στον vulnerable server (file upload). Η γλώσσα PHP είναι εκείνη που έχει χρησιμοποιηθεί για την υλοποίηση των λειτουργιών που εκτελούνται στο backend όπως είναι ο έλεγχος του περιεχομένου του αρχείου το οποίο προορίζεται για αποθήκευση στον server.
Η ιστοσελίδα του challenge, έχει υλοποιηθεί με τέτοιο τρόπο που την καθιστά επιρρεπή σε file inclusion [12] ευπάθεια. Μια τέτοιου είδους ευπάθεια επιτρέπει σε έναν κακόβουλο να συμπεριλάβει, να εκτελέσει αρχεία τα οποία βρίσκονται αποθηκευμένα στον server και στην εκτέλεση απομακρυσμένου κώδικα (Remote Code Execution). Πολλές φορές συνδυάζεται με path traversal [13] ή με την χρήση κάποιου php wrapper με στόχο την ανάγνωση ευαίσθητων πληροφοριών από κρίσιμα αρχεία που βρίσκονται αποθηκευμένα στον server.

Silent Betrayal (Cryptography Challenge)

Το τρίτο, σε σειρά, challenge πραγματεύεται θέματα που σχετίζονται με τον κλάδο της κρυπτογραφίας. Ο διαγωνιζόμενος έχει στη διάθεσή του το αρχείο που περιέχει τα διάφορα μηνύματα τα οποία έχουν κρυπτογραφηθεί με την χρήση του συμμετρικού αλγορίθμου κρυπτογράφησης One-Time-Pad (OTP) [14]. Η κρυπτογράφηση/αποκρυπτογράφηση επιτυγχάνεται εκτελώντας την πράξη xor μεταξύ μηνύματος και κλειδιού (για την κρυπτογράφηση) ή μεταξύ

κρυπτογραφημένου μηνύματος και κλειδιού (για την αποκρυπτογράφηση). Ο αλγόριθμος αυτός θεωρείται απόλυτα ασφαλής στην περίπτωση που υπακούει τις παρακάτω υποθέσεις:

- Το κλειδί έχει ίδιο μήκος με το προς κρυπτογράφηση κείμενο (plaintext).
- Το κλειδί είναι τυχαίο.
- Χρήση διαφορετικού κλειδιού για κάθε καινούργιο μήνυμα προς κρυπτογράφηση.

Ο διαγωνιζόμενος θα πρέπει να λάβει υπόψη του τις αδυναμίες του αλγορίθμου OTP οι οποίες, σε συνδυασμό με τεχνικές κρυπτανάλυσης, θα τον βοηθήσουν να ανακτήσει το κρυπτογραφημένο flag το οποίο εντοπίζεται σε κάποιο από τα κρυπτογραφημένα μηνύματα.

Stealthy Snapshots (Steganography Challenge)

Το τελευταίο, σε σειρά, challenge ανήκει στην κατηγορία της στεγανογραφίας. Η στεγανογραφία ("*στεγανός* + *γραφή*) αποτελεί την τέχνη με την οποία επιτυγχάνεται η απόκρυψη και μετάδοση πληροφοριών μέσω αρχείων (π.χ εικόνας, ήχου, κειμένου κ.α) τα οποία, εκ πρώτης όψεως, δεν εγείρουν υποψίες ως προς τα δεδομένα που περιέχονται κρυμμένα σε αυτά [15]. Να επισημάνουμε πως η στεγανογραφία δεν πρέπει να συγχέεται με την κρυπτογραφία καθώς έχει ως κύριο στόχο την μείωση της πιθανότητας ανίχνευσης ενός μηνύματος. Γι αυτό και αρκετές φορές μπορεί να χρησιμοποιηθεί σαν επιπλέον επίπεδο προστασίας ενός κρυπτογραφημένου μηνύματος.

Προχωρώντας στην περιγραφή του challenge, ο τίτλος “Stealthy Snapshots” έχει επιλεγεί, σκόπιμα, προκειμένου να προϊδεάσει τον διαγωνιζόμενο πως στην (αρχική) εικόνα περιέχονται άλλες κρυμμένες εικόνες/αρχεία. Αρχικά, παρέχεται στον διαγωνιζόμενο το zip αρχείο το οποίο περιέχει την (αρχική) εικόνα με την οποία ξεκινά το challenge. Συνολικά, το challenge, αποτελείται από πέντε (νοητά) επίπεδα. Τα πρώτα τέσσερα επίπεδα, σχετίζονται με την εύρεση κρυμμένων εικόνων/αρχείων και την συλλογή χρήσιμων πληροφοριών -

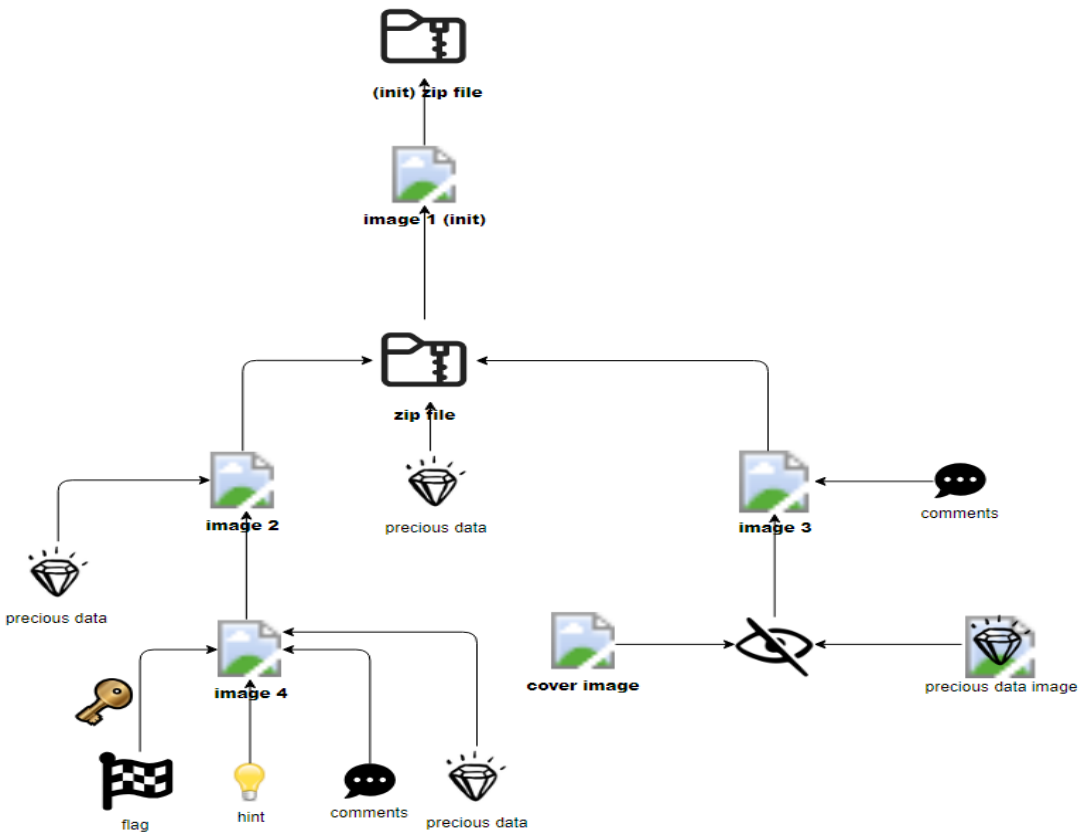
hints που βρίσκονται κρυμμένα στα προηγούμενα. Στο πέμπτο επίπεδο, ο διαγωνιζόμενος θα πρέπει να χρησιμοποιήσει κατάλληλα τα βοηθητικά δεδομένα, τα οποία έχει ανακτήσει στα προηγούμενα επίπεδα, με στόχο την εύρεση του flag.

Η παρακάτω εικόνα παρουσιάζει το σκεπτικό το οποίο ακολουθήσαμε προκειμένου να υλοποιήσουμε το challenge. Ο τρόπος με τον οποίο εργαστήκαμε έχει ως εξής:

1. image_4:
    a. Κάνοντας χρήση του εργαλείου στεγανογραφίας *steghide* [16], ενσωματώσαμε το flag του challenge στην image_4 προστατεύοντάς το με έναν passphrase κωδικό.
    b. Προσθέσαμε, κάνοντας append στο τέλος της image_4, το hint μας εκτελώντας την εντολή *cat hint >> image_4.*
    c. Προσθέσαμε βοηθητικά δεδομένα (*precious data*) με την εντολή *echo <precious_data> >> image_4.*
    d. Με τη βοήθεια του εργαλείου *ExifTool* [16] δημιουργήσαμε ένα "Comment" πεδίο, στα metadata της image_4, στο οποίο προσθέσαμε ένα βοηθητικό σχόλιο (comments). Το προηγούμενο το πετύχαμε με τη βοήθεια της εντολής
    exiftool -comment='<comments>' image_4.
2. image_3:
    a. Η image_3 αποτελείται από δύο εικόνες. Στην πρώτη εικόνα απεικονίζονται τα βοηθητικά δεδομένα. Για να "κρύψουμε" τα βοηθητικά δεδομένα τοποθετήσαμε ως "κάλυμμα" μια δεύτερη εικόνα.
    b. Βοηθητικό σχόλιο (comment) αποτελεί το όνομα που δώσαμε στην image_3.
    c. Εργαστήκαμε όπως το 1d και δημιουργήσαμε ένα "Comment" πεδίο, στα metadata της image_3, στο οποίο προσθέσαμε το hint.
3. image_2:

a. Εργαστήκαμε όπως το 1d και δημιουργήσαμε ένα “Comment” πεδίο, στα metadata της image_2, στο οποίο προσθέσαμε την base64 κωδικοποίηση της image_3 .
b. Δημιουργήσαμε (όπως το 1d), ένα επιπλέον, metadata πεδίο με το όνομα “Description“ στο οποίο προσθέσαμε τα βοηθητικά δεδομένα.

4. image_1
   a. Στη συνέχεια, αποθηκεύσαμε τα βοηθητικά δεδομένα και τις image_2 και image_3 σε ένα zip αρχείο το οποίο ενσωματώσαμε στην image_1 (init) με την εντολή cat <zip_file> >> image_1.



Εικόνα 5: Διάγραμμα υλοποίησης “Stealthy Snapshots” challenge

Challenges Walkthroughs

Reverse Engineering Challenge

Στόχος

Το συγκεκριμένο challenge (*Fortunate Sam*) εμπεριέχει μηχανισμό anti-tracing και τεχνικές obfuscation που επιβραδύνουν τη διαδικασία του reverse engineering. Ο διαγωνιζόμενος καλείται να παρακάμψει τους εν λόγω μηχανισμούς οι οποίοι εμποδίζουν την εύρεση του flag.

Σενάριο

Ο Sam θεωρεί τον εαυτό του ιδιαίτερα τυχερό υποστηρίζοντας πως είναι σε θέση να κερδίσει οποιοδήποτε τυχερό παιχνίδι. Πρόσφατα αναδείχθηκε νικητής σε ένα παιχνίδι υψηλών στοιχημάτων κερδίζοντας, εκτός από ένα μεγάλο έπαθλο, την "άπιαστη" σημαία (flag). Ο Sam καυχιέται πως κανένας άλλος δεν διαθέτει τις ικανότητες να βρει τη σημαία που κρύβεται μέσα στο παιχνίδι. Χρησιμοποιήστε τις ικανότητές σας στο reverse engineering για να ξεγελάσετε τον Sam. Το challenge δίνει και κάποια βοηθητικά στοιχεία (hints 1-2) όπως:

1. Το format του flag.
2. Χρήση οποιασδήποτε reverse engineering μεθόδου.

Εικόνα 6: *Σενάριο reverse engineering challenge*

Ενδεικτική Λύση

Έχοντας στη διάθεσή μας το εκτελέσιμο αρχείο, εκτελούμε την εντολή file <file_name> προκειμένου να εξάγουμε πληροφορίες όπως το είδος του εκτελέσιμου, τον τρόπο με τον οποίο έγινε linking κ.α. Από τα αποτελέσματα που μας επιστρέφει η εντολή file ενημερωνόμαστε πως έχουμε να κάνουμε με ένα ELF 64-bit LSB pie εκτελέσιμο αρχείο από το οποίο έχουν αφαιρεθεί debugging πληροφορίες (stripped). Το τελευταίο επιβραδύνει τη διαδικασία του reversing καθώς αφαιρούνται χρήσιμες πληροφορίες όπως ονόματα μεταβλητών και συναρτήσεων.

```
mrns@mrns-virtual-machine:~/Desktop/RE_Challenge$ file re_challenge
re_challenge: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamical
ly linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=cea0bb8fc5d009
0eb123ccef8968c16e2a4f3330, for GNU/Linux 3.2.0, stripped
```

Εικόνα 7: Αποτέλεσμα εντολής file

Στη συνέχεια, εκτελούμε το πρόγραμμα προκειμένου να μελετήσουμε, σε πρώτο στάδιο, τον τρόπο λειτουργίας του. Το πρόγραμμα ζητάει από τον χρήστη να μαντέψει έναν αριθμό. Ο χρήστης έχει στη διάθεσή του τρεις προσπάθειες.

```
mrns@mrns-virtual-machine:~/Desktop/RE_Challenge$ ./re_challenge
Can you guess the number?
 number=1
You entered=1
I'm sorry :(
You have 2 attemps
Can you guess the number?
 number=2
You entered=2
I'm sorry :(
You have 1 attemps
Can you guess the number?
 number=3
You entered=3
I'm sorry :(
You have 0 attemps
```

Εικόνα 8: : Αρχική συμπεριφορά προγράμματος

Σε αυτό το σημείο θα μπορούσαμε να εκτελέσουμε, επιπλέον, τις εντολές *ltrace* [18] και *strace* [19] για να εξάγουμε πληροφορίες που έχουν να κάνουν με την χρήση των dynamic libraries και των system calls που χρησιμοποιούνται στο πρόγραμμα. Εκτελώντας τις δυο προηγούμενες εντολές λαμβάνουμε τα αποτελέσματα που παρουσιάζονται στην παρακάτω εικόνα.

```
mrns@mrns-virtual-machine:~/Desktop/RE_Challenge$ ltrace ./re_challenge
dlopen(nil, 2)                                   = 0x7f65849642e0
dlsym(0x7f65849642e0, "ptrace")                  = 0x7f658471bf90
printf("I don't like cheaters! %s \n Exit"..., "\360\237\230\241"I don't like cheaters! 😡
 Exiting...
) = 41
exit(1 <no return ...>
+++ exited (status 1) +++
```

Εικόνα 9: Αποτέλεσμα ltrace

```
mrns@mrns-virtual-machine:~/Desktop/RE_Challenge$ strace ./re_challenge
execve("./re_challenge", ["./re_challenge"], 0x7ffdcfc99160 /* 55 vars */) = 0
brk(NULL)                                        = 0x561933b76000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd45d19c80) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd58f050000
access("/etc/ld.so.preload", R_OK)               = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=70923, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 70923, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd58f03e000
close(3)                                         = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd58ee00000
mprotect(0x7fd58ee28000, 2023424, PROT_NONE) = 0
mmap(0x7fd58ee28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28
mmap(0x7fd58efbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7
mmap(0x7fd58f016000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215
mmap(0x7fd58f01c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
                                                 = 0
Terminal
, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd58f03b000
arch_prctl(ARCH_SET_FS, 0x7fd58f03b740)          = 0
set_tid_address(0x7fd58f03ba10)                  = 14970
set_robust_list(0x7fd58f03ba20, 24)              = 0
rseq(0x7fd58f03c0e0, 0x20, 0, 0x53053053)        = 0
mprotect(0x7fd58f016000, 16384, PROT_READ)       = 0
mprotect(0x561932218000, 4096, PROT_READ)        = 0
mprotect(0x7fd58f08a000, 8192, PROT_READ)        = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fd58f03e000, 70923)                    = 0
ptrace(PTRACE_TRACEME)                           = -1 EPERM (Operation not permitted)
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
getrandom("\xbe\x7c\x59\xbf\x1d\x81\x63\xaf", 8, GRND_NONBLOCK) = 8
brk(NULL)                                        = 0x561933b76000
brk(0x561933b97000)                              = 0x561933b97000
write(1, "I don't like cheaters! \360\237\230\241 \n Ex"..., 41I don't like cheaters! 😡
 Exiting...
) = 41
exit_group(1)                                    = ?
+++ exited with 1 +++
```

Εικόνα 10: Αποτέλεσμα strace

Βάσει των αποτελεσμάτων που λαμβάνουμε μπορούμε να δούμε πως χρησιμοποιούνται οι system calls getrandom(), ptrace() και οι συναρτήσεις dlopen και dlsym. Η τελευταία σε συνδυασμό με το μήνυμα "*I don't like cheaters!*" μας προϊδεάζει για την χρήση anti-tracing μηχανισμού μέσα στο

πρόγραμμα. Επειδή, είναι πιθανό, να το πρόγραμμα να χρησιμοποιεί επιπλέον συναρτήσεις ή/και system calls που θα μας βοηθήσουν, περαιτέρω, στην διαδικασία του reverse engineering μπορούμε να εντοπίσουμε και να απενεργοποιήσουμε τον συγκεκριμένο anti-tracing μηχανισμό. Εξετάζοντας το εκτελέσιμό μας με το *Ghidra* [20], μεταβαίνουμε στο tab " Defined Strings " για να εντοπίσουμε την θέση που βρίσκεται το string "*I don't like cheaters!"* όπου θα μας οδηγήσει στον anti-tracing μηχανισμό. Στα αποτελέσματα που μας επιστρέφονται παρατηρούμε, εκτός από γνωστά strings (μαύρη υπογράμμιση) και strings τα οποία συναντάμε πρώτη φορά (μπλε και κόκκινη υπογράμμιση). Απο τα τελευταια, αξίζει να επικεντρωθούμε και σε εκείνα που κάνουν αναφορά στο flag και στο HINT (τονισμένα με κίτρινο χρώμα).

Defined Strings - 65 items

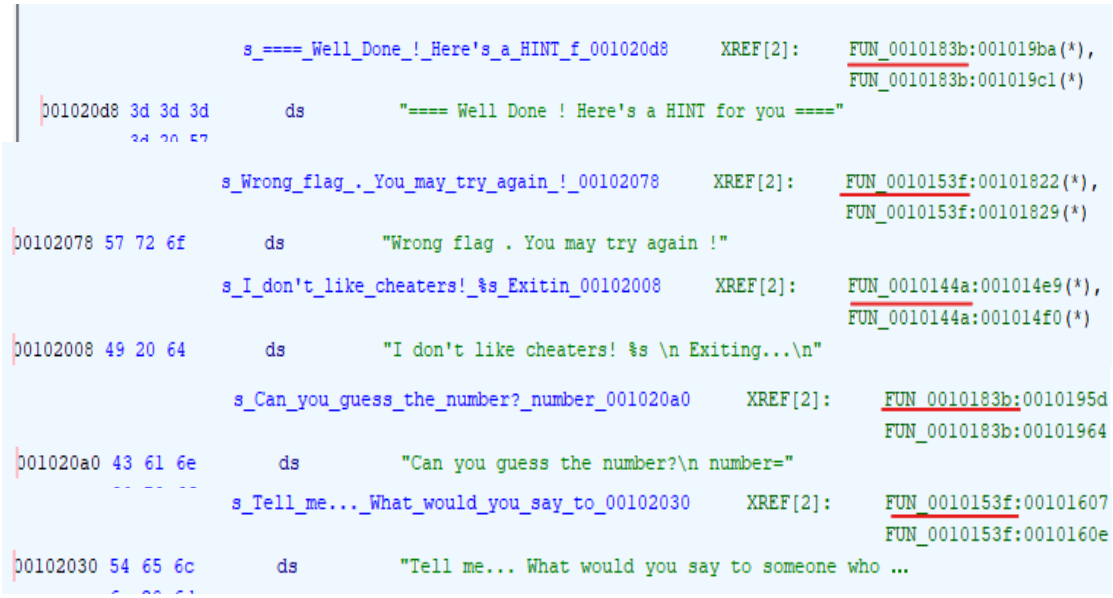
Location	String Value	String Representation	Data Type
001005c9	dlsym	"dlsym"	ds
001005cf	srand	"srand"	ds
001005d5	puts	"puts"	ds
001005da	strlen	"strlen"	ds
001005e1	dlopen	"dlopen"	ds
001005e8	dlclose	"dlclose"	ds
001005f0	__isoc99_scanf	"__isoc99_scanf"	ds
001005ff	exit	"exit"	ds
00100604	__stack_chk_fail	"__stack_chk_fail"	ds
00100615	printf	"printf"	ds
0010061c	time	"time"	ds
00100621	libc.so.6	"libc.so.6"	ds
0010062b	GLIBC_2.7	"GLIBC_2.7"	ds
00100635	GLIBC_2.4	"GLIBC_2.4"	ds
0010063f	GLIBC_2.2.5	"GLIBC_2.2.5"	ds
0010064b	GLIBC_2.34	"GLIBC_2.34"	ds
00100656	_ITM_deregisterTMCloneTable	"_ITM_deregisterTMCloneTable"	ds
00100672	__gmon_start__	"__gmon_start__"	ds
00100681	_ITM_registerTMCloneTable	"_ITM_registerTMCloneTable"	ds
00102008	I don't like cheaters! %s Exiting...	"I don't like cheaters! %s \n Exiti...	ds
00102030	Tell me... What would you say to ...	"Tell me... What would you say t...	ds
00102078	Wrong flag . You may try again !	"Wrong flag . You may try again !"	ds
001020a0	Can you guess the number? num...	"Can you guess the number?\n n...	ds
001020c6	You entered=%d	"You entered=%d\n"	ds
001020d8	==== Well Done ! Here's a HINT ...	"==== Well Done ! Here's a HIN...	ds
00102108	I'm sorry :(You have %d attemps	"I'm sorry :(\nYou have %d atte...	ds
00102191	zR	"zR"	ds
00104010	abcdefghijklmnopqrstuvwxyz{|}	"abcdefghijklmnopqrstuvwxyz{|} "	ds
00104030	0123456789	"0123456789"	ds
00104040	ABCDEFGHIJKLMNOPQRSTUVWXYZ	"ABCDEFGHIJKLMNOPQRSTUVW...	ds

Filter:

Εικόνα 11: Strings που εντοπίζει το Ghidra

Το *Ghidra* μας παρέχει τη δυνατότητα εντοπισμού του σημείου που εντοπίζεται το κάθε string στο εκτελέσιμο κάνοντας διπλό κλικ πάνω σε αυτά. Συγκεκριμένα η αντιστοιχία string και θέσης- συνάρτησης που καλείται είναι:

- HINT -> FUN_0010183b.
- Wrong_flag -> FUN_0010153f.
- "*I don't like cheaters!"* -> FUN_0010144a.
- *"Can you guess the number?"*-> FUN_0010183b.
- *"Tell me... What would you say to someone who is lucky ?"* -> FUN_0010153f.



Εικόνα 12: : Θέσεις - Συναρτήσεις που εμφανίζονται τα strings

Εξετάζοντας τον ψευδοκώδικα της FUN_0010144a μπορούμε να εντοπίσουμε το σημείο όπου γίνεται ο έλεγχος για τη χρήση tracing μηχανισμού (γραμμή 31). Στη συνέχεια, εντοπίζουμε και τροποποιούμε (patching) την αντίστοιχη εντολή assembly από CMP RAX,-0x1 σε CMP RAX,0xa απενεργοποιώντας τον anti-tracing μηχανισμό.

```
20   local_10 = *(long *)(in_FS_OFFSET + 0x28);
21   local_17 = s_abcdefghijklmnopqrstuvwxyz{|}_00104090[15];
22   local_16 = s_abcdefghijklmnopqrstuvwxyz{|}_00104090[19];
23   local_15 = 0x72;
24   local_14 = 0x61;
25   local_13 = 99;
26   local_12 = 0x65;
27   local_11 = 0;
28   uVar1 = dlopen(0,2);
29   pcVar2 = (code *)dlsym(uVar1,&local_17);
30   lVar3 = (*pcVar2)(0,0,0,0);
31   if (lVar3 == -1) {
32     local_1c = 0xa1989ff0;
33     local_18 = 0;
34     printf("I don\'t like cheaters! %s \n Exiting...\n",&local_1c);
35                     /* WARNING: Subroutine does not return */
36     exit(1);
37   }
38   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
39                     /* WARNING: Subroutine does not return */
40     __stack_chk_fail();
41   }
42   return;
43 }
```

Εικόνα 13: Ψευδοκώδικας FUN_0010144a

```
001014cf ff d0          CALL    RAX
001014d1 48 83 f8 ff    CMP     RAX,-0x1
001014d5 75 30          JNZ     LAB_00101507
001014d7 c7 45 ec       MOV     dword ptr [RBP + local_1c],0xa
         f0 9f 98 a1
001014de c6 45 f0 00    MOV     byte ptr [RBP + local_18],0x0
001014e2 48 8d 45 ec    LEA     RAX=>local_1c,[RBP + -0x14]
001014e6 48 89 c6       MOV     RSI,RAX
001014e9 48 8d 05       LEA     RAX,[s_I_don't_like_cheaters!_
```

```
23   local_15 = 0x72;
24   local_14 = 0x61;
25   local_13 = 99;
26   local_12 = 0x65;
27   local_11 = 0;
28   uVar1 = dlopen(0,2);
29   pcVar2 = (code *)dlsym(uVar1,&local_17
30   lVar3 = (*pcVar2)(0,0,0,0);
31   if (lVar3 == -1) {
32     local_1c = 0xa1989ff0;
```

Εικόνα 14: : Κρίσιμη εντολή assembly

Ακολουθώντας το παραπάνω σκεπτικό, μπορούμε να πετύχουμε την FUN_0010183b να επιστρέφει πάντα τον αριθμό 0 τροποποιώντας τις αντίστοιχες εντολές (βλ. παρακάτω εικόνα)

```
00101363 89 45 fc       MOV     dword ptr [RBP + local_c],EAX
00101366 8b 45 fc       MOV     EAX,dword ptr [RBP + local_c]
```

Εικόνα 15: Αρχικός κώδικας assembly

```
00101363 89 45 fc       MOV        dword ptr [RBP + local_c],EAX
00101366 31 c0          XOR        EAX,EAX
00101368 90             NOP
```

Εικόνα 16: Τροποποιημένες εντολές assembly

Έχοντας, πλέον, απενεργοποιήσει τον anti-tracing μηχανισμό μπορούμε να δούμε πως, εκτελώντας ξανά τις εντολές ltrace και strace δεν εντοπίζουμε νέες system calls ή/και νέες συναρτήσεις που καλεί το πρόγραμμα. Παρόλα αυτά λαμβάνουμε ένα HINT το οποίο, ουσιαστικά, μας ενημερώνει πως υπάρχουν strings τα οποία έχουν υποστεί κάποιου είδους obfuscation(“*some strings are hidden*”).

```
mrns@mrns-virtual-machine:~/Desktop/Challenges$ ./re_challenge
Can you guess the number?
 number=0
You entered=0
==== Well Done ! Here's a HINT for you ====
some strings are hidden
Tell me... What would you say to someone who is lucky ?
```

Εικόνα 17: HINT μήνυμα

Επόμενο βήμα είναι να εξετάσουμε την FUN_0010153f στην οποία εμφανίζεται το τελευταίο μήνυμα της παραπάνω εικόνας (“*Tell me… What would you say to someone who is lucky ?*”). Παρατηρώντας τον ψευδοκώδικα βλέπουμε πως η συνάρτηση εκτυπώνει, εκτός από το μήνυμα *“Wrong flag . You may try again !”*, τρία επιπλέον strings των οποίων οι τιμές βρίσκονται αποθηκευμένες στις μεταβλητές *__s*, *uVar4* και *pvVar2* (γραμμές 83, 85, 86).

```
76  iVar1 = FUN_0010136b(pvVar2,local_1e,0xf);
77  if (iVar1 != 0) {
78    puts("Wrong flag . You may try again !");
79                    /* WARNING: Subroutine does not return */
80    exit(1);
81  }
82  __s = (char *)FUN_001012c9(&DAT_00104060,0x4d,0x10);
83  puts(__s);
84  uVar4 = FUN_001012c9(&DAT_00104070,0x58,5);
85  printf("%s",uVar4);
86  printf("%s}\n",pvVar2);
87                    /* WARNING: Subroutine does not return */
88  exit(1);
89}
```

Εικόνα 18: : Μέρος FUN_0010153f που τυπώνει τα strings.

Εξετάζοντας την συνάρτηση FUN_001012c9 μπορούμε να καταλάβουμε πως επιστρέφει το αποτέλεσμα της πράξης xor μεταξύ κάθε χαρακτήρα ενός string (*param_1*) με μήκος *param_3* με ένα συγκεκριμένο χαρακτήρα (*param_2*).

```
1
2 void * FUN_001012c9(long param_1,byte param_2,int param_3)
3
4 {
5   void *pvVar1;
6   int local_14;
7
8   pvVar1 = malloc((long)param_3);
9   for (local_14 = 0; local_14 < param_3; local_14 = local_14 + 1) {
10    *(byte *)((long)pvVar1 + (long)local_14) = *(byte *)(param_1 + local_14) ^ param_2;
11  }
12  *(undefined *)((long)pvVar1 + (long)param_3) = 0;
13  return pvVar1;
14 }
```

Εικόνα 19: Ψευδοκώδικας FUN_001012c9.

Συνεχίζοντας, μεταβαίνουμε στον ψευδοκώδικα της συνάρτησης FUN_0010136b προκειμένου να εξάγουμε πληροφορίες σχετικά με την *pvVar2* και τις τιμές που μπορεί να λάβει η *iVar1*. Ενδιαφέρον παρουσιάζει η κλήση της *dlsym* η οποία χρησιμοποιείται για να καλέσουμε μια shared library/object. Συγκεκριμένα, καλεί τη shared library της οποίας το όνομα βρίσκεται αποθηκευμένο στο string του οποίου ο πρώτος χαρακτήρας ξεκινάει από την *local_18* και εκτείνεται μέχρι την local_11. Κάνοντας hover τον κέρσορα στην *local_18* μπορούμε να δούμε πως ο πρώτος χαρακτήρας του string local_18 είναι

ο χαρακτήρας “s” ο οποίος βρίσκεται στην 18 θέση του πίνακα/string “s_abcdefghijklmnopqrstuvwxyz{|}*00104010”*. Ακολουθώντας την προηγούμενη λογική μέχρι την *local*_15 και παίρνοντας την ASCII αναπαράσταση των *local_14-11 (*99=c, 0x6d=m κ.ο.κ *)* καταλαβαίνουμε πως εκτελείται η συνάρτηση “*strncmp*” η οποία συγκρίνει τους πρώτους 14 (0xe) χαρακτήρες δύο strings και επιστρέφει 0 αν είναι ίδιοι.

```
19  local_10 = *(long *)(in_FS_OFFSET + 0x28);
20  local_18 = s_abcdefghijklmnopqrstuvwxyz{|}_00104010[18];
21  local_17 = s_abcdefghijklmnopqrstuvwxyz{|}_00104010[19];
22  local_16 = s_abcdefghijklmnopqrstuvwxyz{|}_00104010[17];
23  local_15 = s_abcdefghijklmnopqrstuvwxyz{|}_00104010[13];
24  local_14 = 99;
25  local_13 = 0x6d;
26  local_12 = 0x70;
27  local_11 = 0;
28  uVar2 = dlopen(0,2);
29  pcVar3 = (code *)dlsym(uVar2,&local_18);
30  if (pcVar3 == (code *)0x0) {
31    dlclose(uVar2);
32    uVar2 = 0xffffffff;
33  }
34  else {
35    iVar1 = (*pcVar3)(param_1,param_2,0xe);
36    if (iVar1 == 0) {
37      dlclose(uVar2);
38      uVar2 = 0;
39    }
40    else {
41      dlclose(uVar2);
42      uVar2 = 1;
43    }
```

Εικόνα 20: Ψευδοκώδικας FUN_0010136b

Γνωρίζοντας τον ρόλο της *iVar1*, ένας γρήγορος τρόπος να δούμε τις τιμές των *_s* και *uVar4* ειναι να μετατρέψουμε τη συνθήκη *iVar1!=0* σε *iVar1==0* εφαρμόζοντας patching στις αντίστοιχες εντολές της FUN_0010136b. Με αυτόν τον τρόπο μπορούμε να αποφανθούμε πως η

- *__s=”*Here is the flag*”*.
- *uVar4*=”flag{”.
- *pvVar2* περιέχει το flag.

```
mrns@mrns-virtual-machine:~/Desktop/RE_Challenge$ ./re_challenge
Can you guess the number?
 number=0
You entered=0
==== Well Done ! Here's a HINT for you ====
some strings are hidden
Tell me... What would you say to someone who is lucky ?
123456789abcde
Here is the flag
flag{}
```

Εικόνα 21: Strings που επιστρέφουν οι μεταβλητές __s και uVar4

Εξετάζοντας, περεταίρω, την FUN_0010153f εστιάζουμε στο κομμάτι του ψευδοκώδικα το οποίο περιέχεται μέσα στο for loop . Στο συγκεκριμένο κομμάτι κατασκευάζεται το string που πρέπει να δώσει ο χρήστης ως είσοδο μετά την εκτύπωση του string "*Tell me… What would you say to someone who is lucky ?*". Πιο αναλυτικά:

1. Προσπελαύνεται κάθε τιμή του πίνακα *local_68* =[-0x15, 0x14, 2, 10, 0xfffffffd0, 0x12, 0xffffffcf, 0xd, 0xfffffff4, 0x11, 0xfffffffd4, 8, 3, 0xfffffffd2] (γραμμές 32-45).
2. Η τιμή του κάθε στοιχείο του *local_68=iVar* ελέγχεται αν ανήκει σε κάποιο διάστημα τιμών (γραμμές 55-57).
3. Αναλόγως το διάστημα στο οποίο ανήκει, δημιουργείται ο επιθυμητός χαρακτήρας *pvVar2* και συγκρίνεται με τον χαρακτήρα του string εισόδου *local_1e* της αντίστοιχης θέσης (γραμμές 58, 64-65, 71).
    a. Αν *pvVar2 != local_1e[i]* τότε σταματά η εκτέλεση του προγράμματος και δεν αποθηκεύεται ο επιθυμητός χαρακτήρας (εντολή break).
    b. Αλλιώς αποθηκεύεται και συνεχίζει η ροή του προγράμματος (γραμμές 59-60, 65-66, 72-73).
4. Επανάλαβε 1-3b για 13 φορές ακόμα.

```
31  local_80 = 0;
32  local_68[0] = -0x15;
33  local_68[1] = 0x14;
34  local_68[2] = 2;
35  local_68[3] = 10;
36  local_68[4] = 0xffffffd0;
37  local_68[5] = 0x12;
38  local_68[6] = 0xffffffcf;
39  local_68[7] = 0xd;
40  local_68[8] = 0xfffffff4;
41  local_68[9] = 0x11;
42  local_68[10] = 0xffffffd4;
43  local_68[11] = 8;
44  local_68[12] = 3;
45  local_68[13] = 0xffffffd2;
46  puts("Tell me... What would you say to someone who is lucky ? ");
47  __isoc99_scanf(&DAT_00102069,local_1e);
48  sVar3 = strlen(local_1e);
49  if (sVar3 != 0xe) {
50                      /* WARNING: Subroutine does not return */
51    exit(1);
52  }
53  for (local_7c = 0; local_7c < 0xe; local_7c = local_7c + 1) {
54    iVar1 = local_68[(int)local_7c];
55    if ((iVar1 < 0) || (0x19 < iVar1)) {
56      if ((iVar1 < -0x20) || (-7 < iVar1)) {
57        if ((-0x32 < iVar1) && (iVar1 < -0x27)) {
58          if (s_0123456789_00104030[iVar1 + 0x31] != local_1e[(int)local_7c]) break;
59          *(char *)((long)local_80 + (long)pvVar2) = s_0123456789_00104030[iVar1 + 0x31];
60          local_80 = local_80 + 1;
61        }
62      }
63      else {
64        if (s_ABCDEFGHIJKLMNOPQRSTUVWXYZ_00104040[iVar1 + 0x20] != local_1e[(int)local_7c]) break;
65        *(char *)((long)local_80 + (long)pvVar2) =
66             s_ABCDEFGHIJKLMNOPQRSTUVWXYZ_00104040[iVar1 + 0x20];
67        local_80 = local_80 + 1;
68      }
69    }
70    else {
71      if (s_abcdefghijklmnopqrstuvwxyz{|}_00104010[iVar1] != local_1e[(int)local_7c]) break;
72      *(char *)((long)local_80 + (long)pvVar2) = s_abcdefghijklmnopqrstuvwxyz{|}_00104010[iVar1];
73      local_80 = local_80 + 1;
74    }
75  }
76  iVar1 = FUN_0010136b(pvVar2,local_1e,0xf);
```

Εικόνα 22: FUN_0010153f

Με βάση το παραπάνω σκεπτικό και τροποποιώντας κατάλληλα τον ψευδοκώδικα μπορούμε να δημιουργήσουμε ένα ένα script σε γλώσσα Python για να βρούμε το επιθυμητό string το οποίο είναι “Luck1s0nUr5id3”. Τέλος εκτελούμε το πρόγραμμα και λαμβάνουμε το flag του challenge.

```python
def decode_string(input_str):


    pvVar2 = []

    s_0123456789_00104030 = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz{|}"
    s_ABCDEFGHIJKLMNOPQRSTUVWXYZ_00104040 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ{|}abcdefghijklmnopqrstuvwxyz0123456789"
    s_abcdefghijklmnopqrstuvwxyz_LB_P_RB_00104010 = "abcdefghijklmnopqrstuvwxyz{|}"

    if len(input_str) != 0xe:
        exit(1)

    for local_7c in range(0xe):
        iVar1 = input_str[local_7c]

        if (iVar1 < 0) or (0x19 < iVar1):
            if (iVar1 < -0x20) or (-7 < iVar1):
                if (-0x32 < iVar1) and (iVar1 < -0x27):

                    pvVar2.append(s_0123456789_00104030[iVar1 + 0x31])

            else:

                pvVar2.append( s_ABCDEFGHIJKLMNOPQRSTUVWXYZ_00104040[iVar1 + 0x20])

        else:

            pvVar2.append(s_abcdefghijklmnopqrstuvwxyz_LB_P_RB_00104010[iVar1])


    return pvVar2

#local_68 = [-0x15, 0x14, 2, 10, 0xffffffd0, 0x12, 0xffffffcf, 0xd, 0xfffffff4, 0x11, 0xffffffd4, 8, 3, 0xffffffd2]
input_str = [-0x15, 0x14, 0x2, 0xa, -48, 0x12, -49, 0xd, -12, 0x11, -44, 0x8, 0x3, -46]

decoded_str = decode_string(input_str)
print(decoded_str)
```

Εικόνα 23: : Script decode_string.py

```
PS C:\Users\Marinos> & C:/Python311/python.exe c:/Users/Marinos/Desktop/decode_string.py
['L', 'u', 'c', 'k', '1', 's', '0', 'n', 'U', 'r', '5', 'i', 'd', '3']
```

Εικόνα 24: String εισόδου

```
mrns@mrns-virtual-machine:~/Desktop/RE_Challenge$ ./re_challenge
Can you guess the number?
 number=0
You entered=0
==== Well Done ! Here's a HINT for you ====
some strings are hidden
Tell me... What would you say to someone who is lucky ?
Luck1s0nUr5id3
Here is the flag
flag{Luck1s0nUr5id3}
```

Εικόνα 25: Reverse engineering challenge flag

Web Exploitation Challenge

Στόχος

Στόχος του συγκεκριμένου challenge είναι να εκπαιδεύσει τον διαγωνιζόμενο να αναγνωρίζει ευπάθειες τύπου file inclusion και να κατανοήσει τον τρόπο με τον οποίο μπορεί, κάποιος κακόβουλος, να τις εκμεταλλευτεί με στόχο την απομακρυσμένη εκτέλεση κώδικα (Remote Code Execution).

Σενάριο

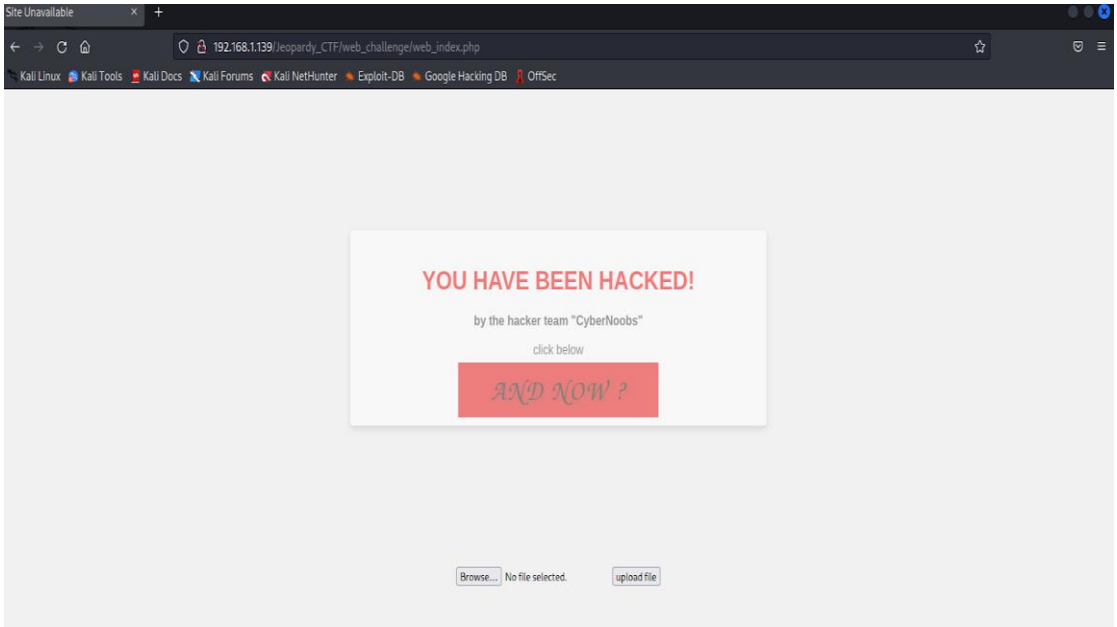
Στο συγκεκριμένο challenge (*Site Compromised*) ο διαγωνιζόμενος κατέχει τον ρόλο ενός επαγγελματία κυβερνοασφάλειας (ethical hacker) από τον οποίο ζητείται να επαναφέρει σε λειτουργία το web site της εταιρείας CyberCom. Συγκεκριμένα, το website της εταιρείας CyberCom έχει δεχτεί επίθεση από μια ομάδα κυβερνοεγκληματιών οι οποίοι έχουν αποκτήσει παράνομη πρόσβαση στον web server που φιλοξενεί το website της εταιρείας μπλοκάροντας την εύρυθμη λειτουργία του . Σκοπός του διαγωνιζόμενου είναι να ανακτήσει το flag το οποίο βρίσκεται hardcoded στο website της CyberCom. Το challenge δίνει και κάποια βοηθητικά στοιχεία (hints 1-3):

1. Το format του flag.
2. Ο διαγωνιζόμενος θα πρέπει να αποκτήσει πρόσβαση στον compromised server.
3. Ο διαγωνιζόμενος θα πρέπει να ψάξει κάποιο μήνυμα που έχουν αφήσει οι hackers.

Εικόνα 26: Σενάριο web exploitation challenge

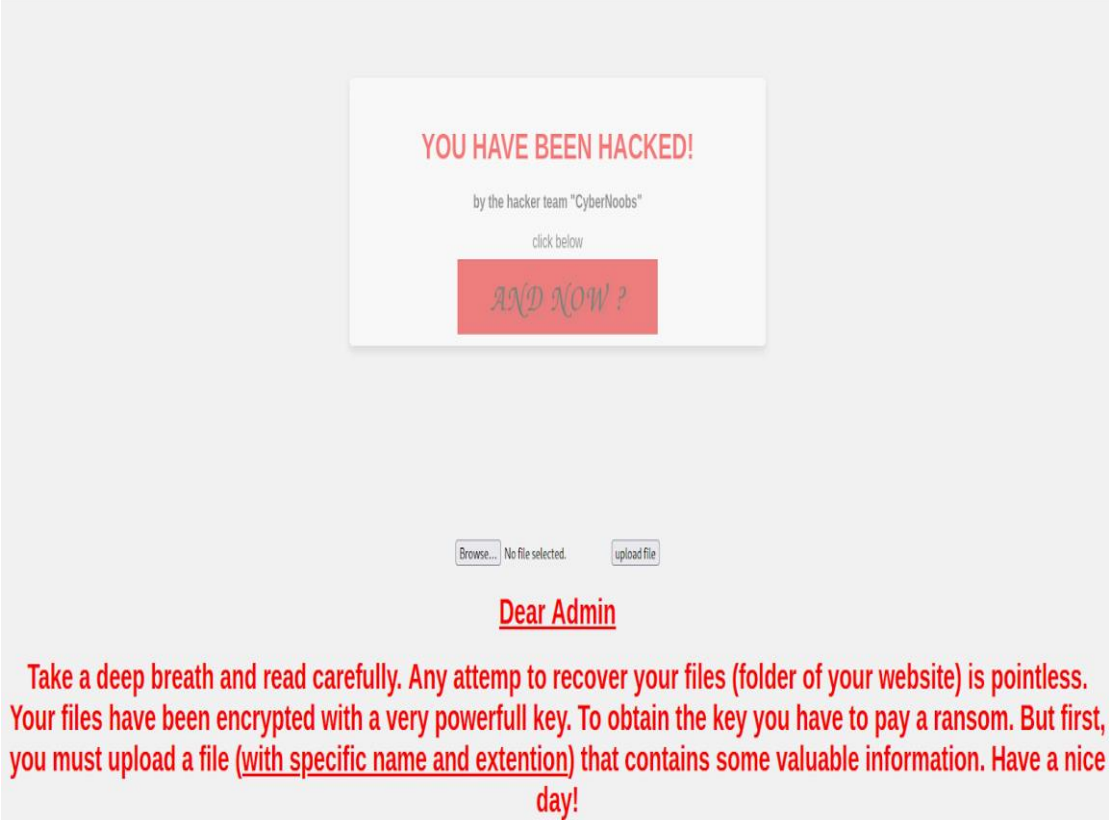
Ενδεικτική Λύση

Ξεκινάμε το challenge κάνοντας κλικ στον σύνδεσμο (URL) ο οποίος μας ανακατευθύνει στην αρχική σελίδα του compromised website.



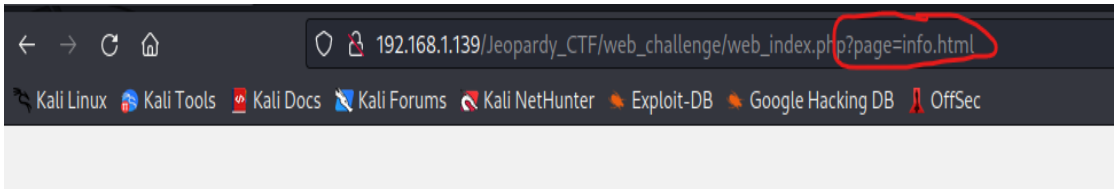
Εικόνα 27: Αρχική σελίδα compromised website

Λαμβάνοντας υπόψη μας το hint 3 αναζητούμε κάποιο μήνυμα που έχουν αφήσει οι hackers για εμάς. Κάνοντας κλικ στο κόκκινο κουμπί (*"AND NOW ?"*) λαμβάνουμε ένα μήνυμα τό οποίο μας αποθαρρύνει να χακάρουμε το compromised website καθώς ο κώδικας του website της CyberCom είναι αποθηκευμένος και προστατευμένος με έναν πολύ ισχυρό κωδικό. Προκειμένου να λάβουμε τον κωδικό θα πρέπει να πληρώσουμε κάποια "λύτρα" αφού πρώτα ανεβάσουμε κάποιες χρήσιμες πληροφορίες που θα μας ζητηθούν σε συγκεκριμένο είδος αρχείου.



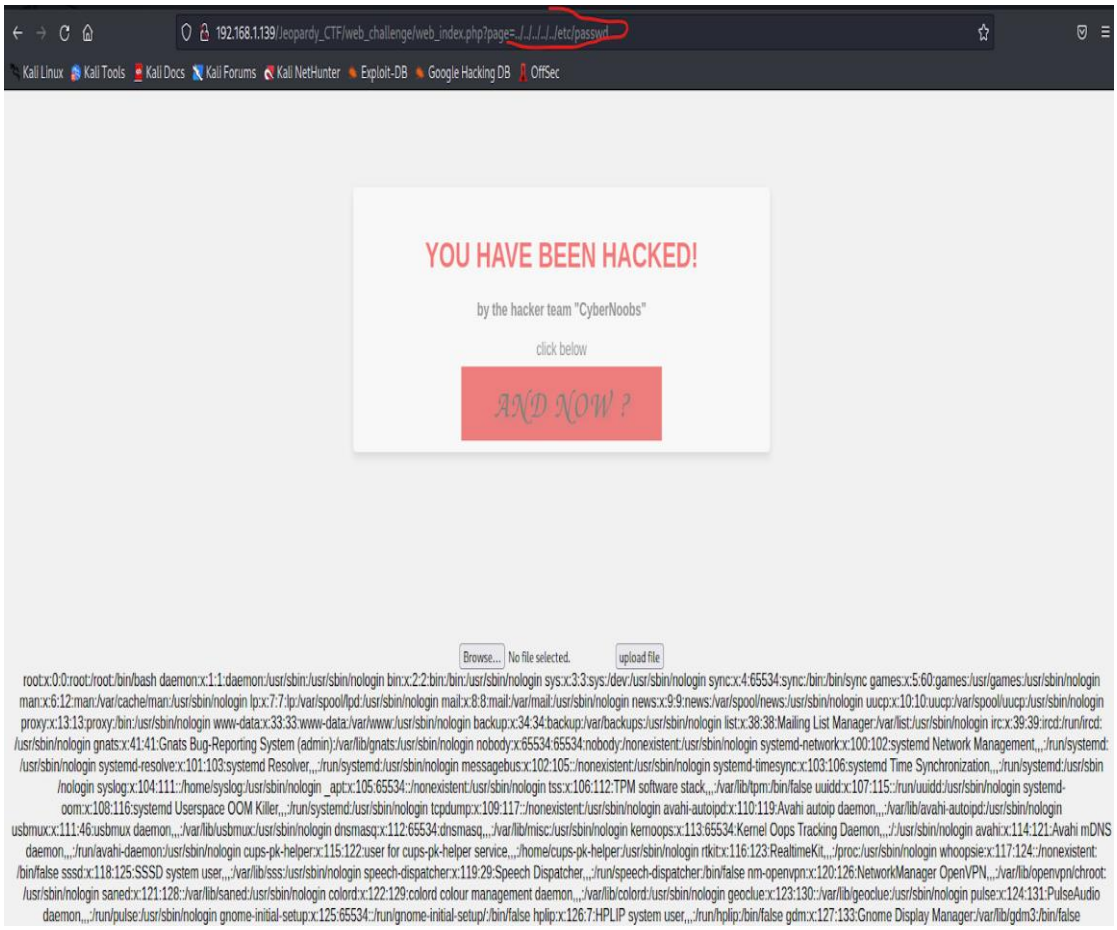
Εικόνα 28: Μήνυμα από τους hackers (hint).

Με μια πιο προσεκτική ματιά στο url παρατηρούμε πως το compromised website είναι ευάλωτο σε file inclusion ευπάθεια.



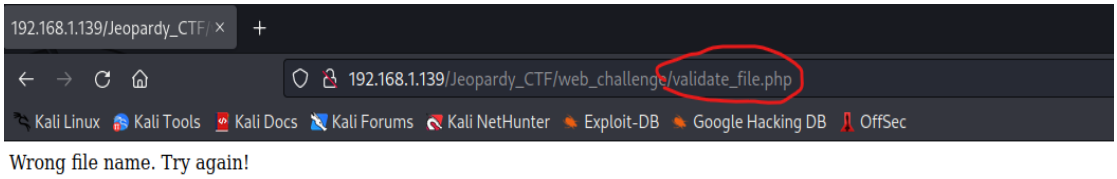
Εικόνα 29: URL compromised website

Αντικαθιστώντας στο URL όπου ?page=info.html με ?page=../../../../../etc/passwd μπορούμε να διαπιστώσουμε πως το website είναι ευάλωτο σε file inclusion ευπάθεια.



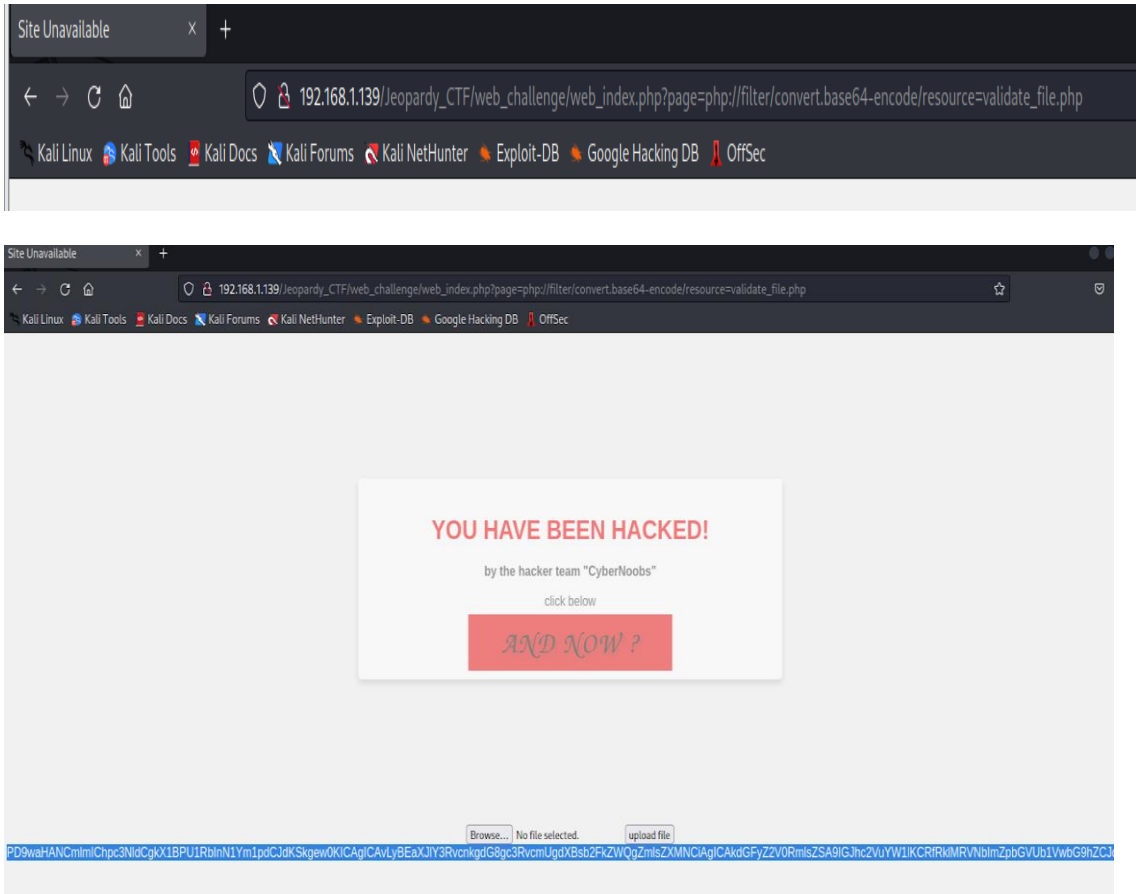
Εικόνα 30: Αναγνώριση file inclusion ευπάθειας

Επόμενο βήμα είναι να μελετήσουμε τη συμπεριφορά του upload μηχανισμού. Κάνοντας upload ένα τυχαίο αρχείο μας εμφανίζεται το μήνυμα της παρακάτω εικόνας.



Εικόνα 31: Μήνυμα εσφαλμένου τύπου αρχείου

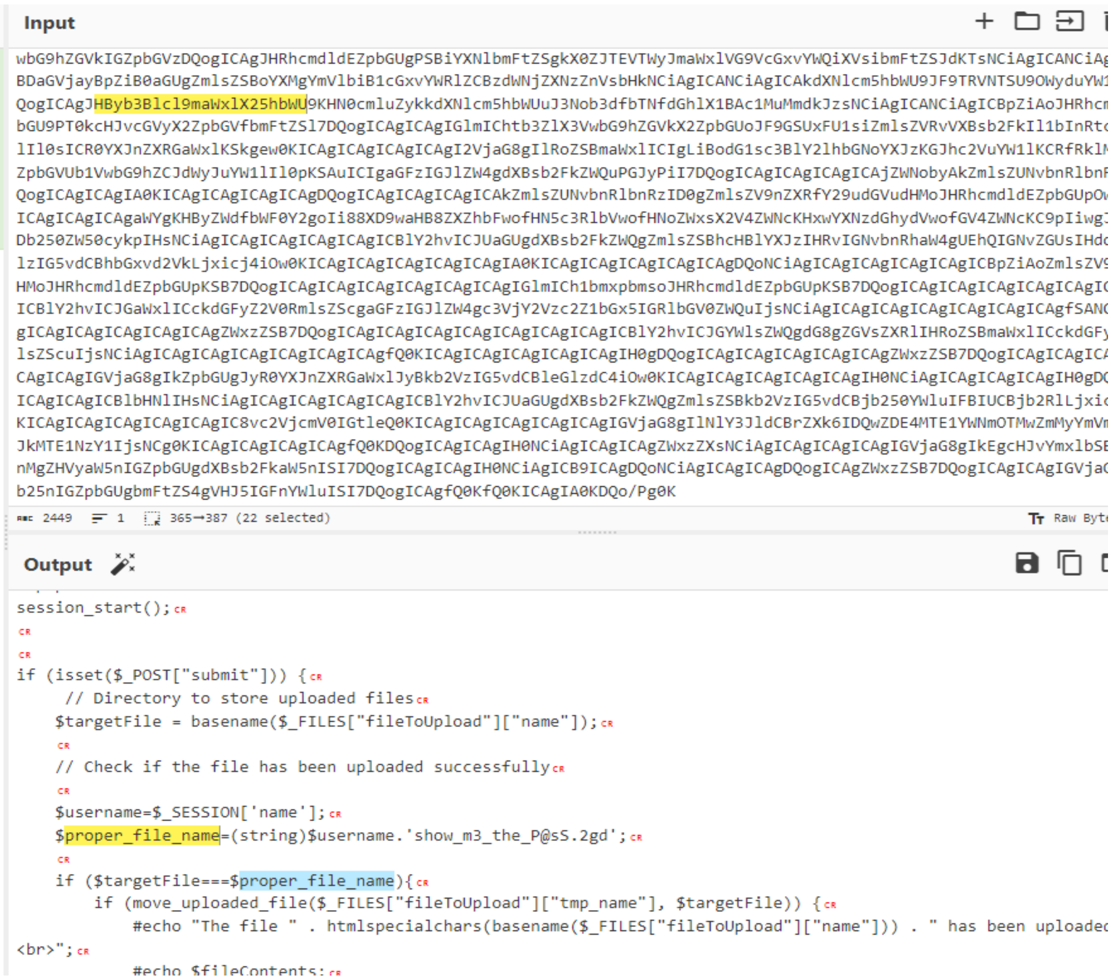
Παρατηρώντας το URL μπορούμε να καταλάβουμε πως το αρχείο το οποίο επρόκειτο να κάνουμε upload στον server υπόκειται σε κάποιου είδους έλεγχο μέσω του αρχείου *validate.php*. Προκειμένου να δούμε τους μηχανισμούς ελέγχου που έχουν υλοποιηθεί θα πρέπει με κάποιο τρόπο να διαβάσουμε το περιεχόμενο-κώδικα του αρχείου *validate.php*. Γενικά, δεν έχουμε τη δυνατότητα να διαβάσουμε τον php κώδικα. Μπορούμε, όμως , να επιχειρήσουμε να ανακτήσουμε την base64 κωδικοποίηση του μέσω της php εντολής php://filter/convert.base64-encode/resource=<php_file> οπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 32: Ανάκτηση base64 κωδικοποίησης αρχείου validate_file.php

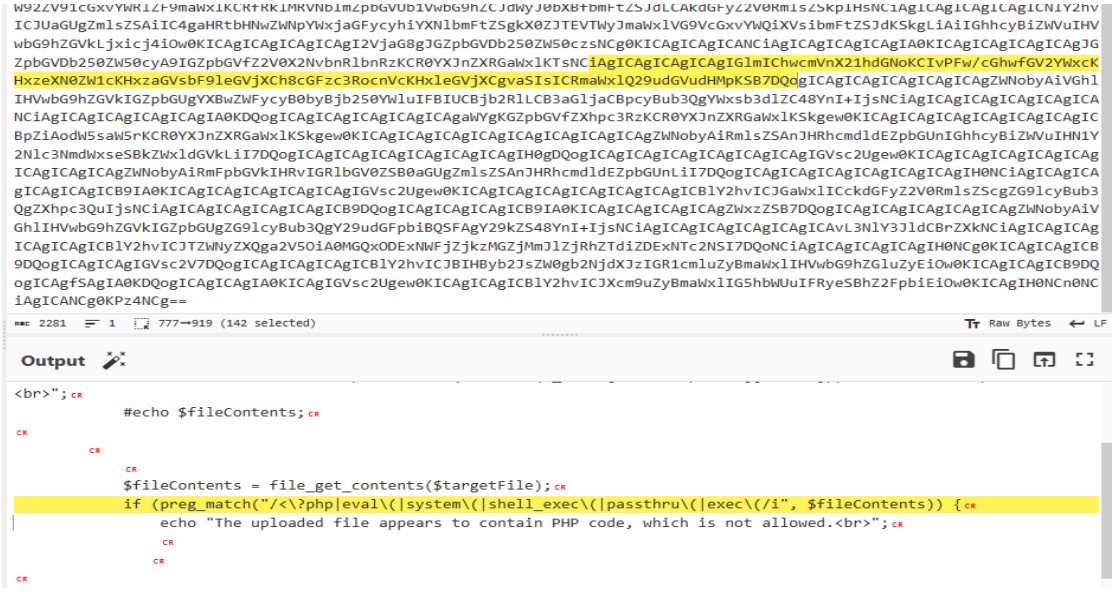
Στη συνέχεια μπορούμε να διαβάσουμε το περιεχόμενο του αρχείου *validate_file.php* αποκωδικοποιώντας με τη βοήθεια κάποιου base64 decoder (που μπορούμε να βρούμε online σε site όπως το ***cyberchef*** [21]).Έχοντας, πλέον, στη διάθεσή μας τον php κώδικα μπορούμε να εξάγουμε τις εξής πληροφορίες:

1. Το αρχείο που πρέπει να ανεβάσουμε πρέπει να έχει συγκεκριμένη ονομασία "<username>*show_m3_the_P@sS.2gd*".



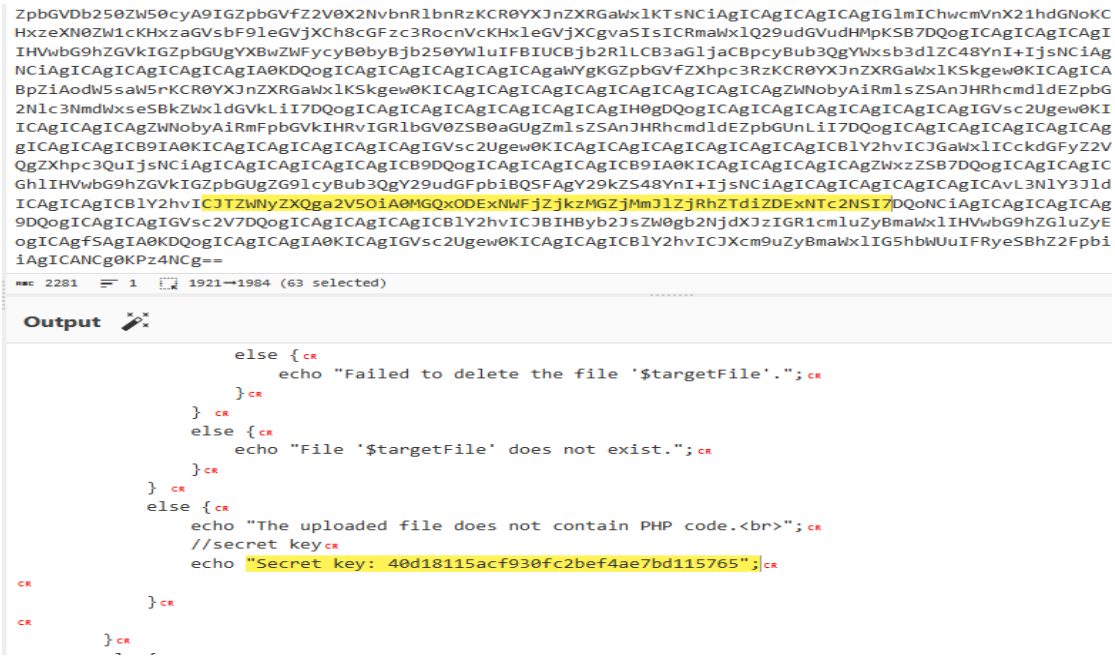
Εικόνα 33: Αποδεκτό όνομα αρχείου προς upload

2. Το αρχείο *validate_file.php* περιέχει φίλτρα που αποτρέπουν την εκτέλεση php συναρτήσεων με τις οποίες μπορούμε να εκτελέσουμε shellcode καθώς, επίσης, γίνεται έλεγχος για το αν το αρχείο περιέχει οποιονδήποτε php κώδικα.



Εικόνα 34: Υλοποίηση φίλτρων (sanitization)

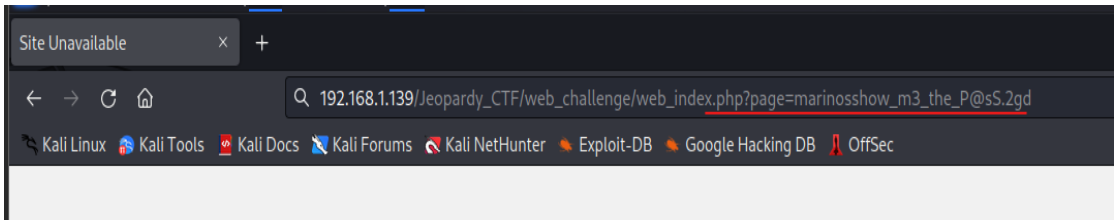
3. Τέλος, μπορούμε να διακρίνουμε τον μυστικό κωδικό-κλειδί το οποίο μπορούμε να χρησιμοποιήσουμε για να διαβάσουμε τον πηγαίο κώδικα του CyberCom website ωστε να λάβουμε το flag του συγκεκριμένου challenge.



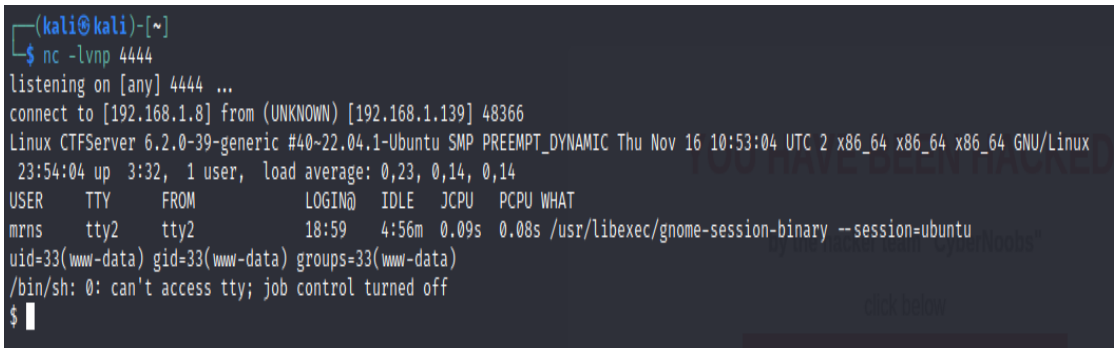
Εικόνα 35: : Κωδικός-κλειδί αποκρυπτογράφησης

Όπως ανακαλύψαμε γίνεται χρήση μηχανισμών που αποτρέπουν τη απομακρυσμένη εκτέλεση php κώδικα. Παρόλα αυτά μπορούμε να παρακάμψουμε αυτού του είδους φίλτρα και να εκτελέσουμε php κώδικα στον compromised server σκεπτόμενοι ως εξής:

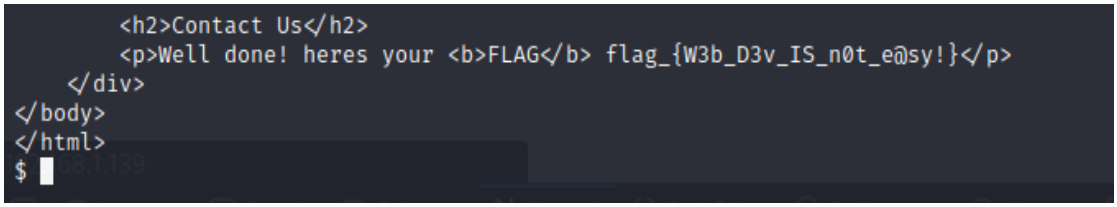
1. Κωδικοποιούμε σε base64 php κώδικα που περιέχει ένα reverse shell payload (τέτοιου είδους κώδικα μπορούμε να βρούμε και σε site όπως το pentestmonkey [22]).
2. Αποθηκεύουμε την base64 αναπαράσταση του php reverse shell στο αρχείο "<ctf_player_username>*show_m3_the_P@sS.2gd* ".
3. Ανεβάζουμε το αρχείο στον compromised server.
4. Ανοίγουμε έναν listener με την εντολή ***nc -lvnp <port>***.
5. Εκμεταλλευόμενοι την file inclusion ευπάθεια εκτελούμε το php reverse shell payload (εικόνα 36) .
6. Πλοηγούμαστε στο path όπου βρίσκεται ο κώδικας του web exploitation challenge.
7. Τέλος, κάνουμε unzip το password protected zip αρχείο κάνοντας χρήση του κωδικού-κλειδί που βρήκαμε προηγουμένως (εντολή ***unrar x <password> <password_protected_file.zip>***) και βρίσκουμε το flag.



Εικόνα 36: εκτελούμε το php reverse shell payload



Εικόνα 37: Reverse shell



Εικόνα 38: Web exploitation challenge flag

Cryptography Challenge

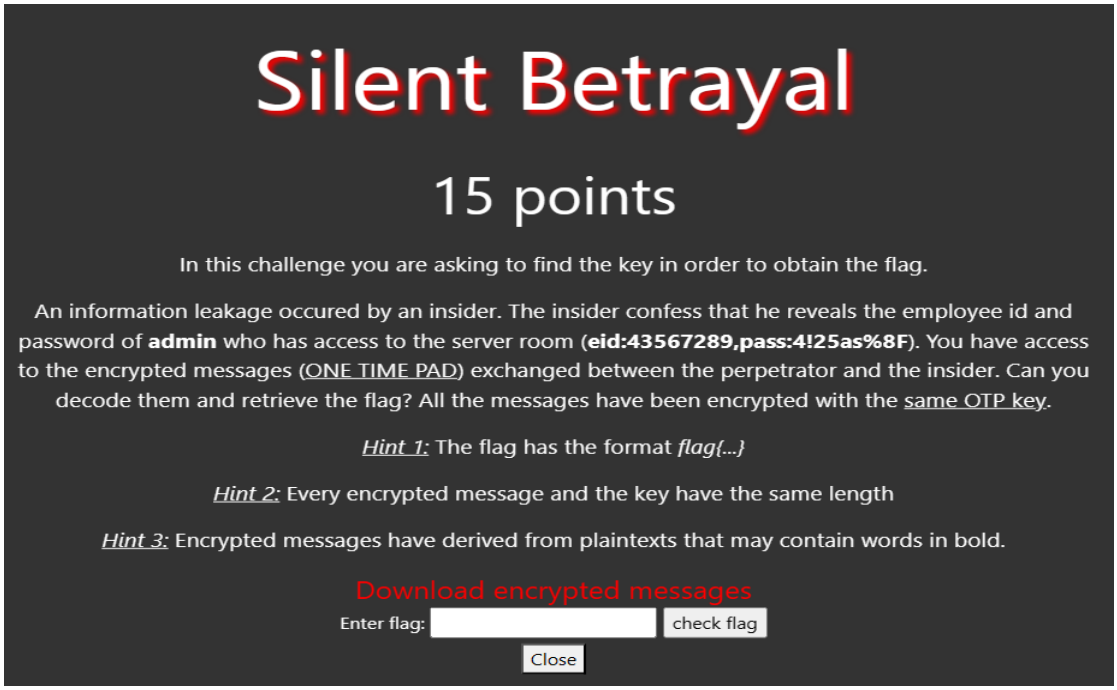
Στόχος

Στόχος του συγκεκριμένου challenge είναι να φέρει σε επαφή τον διαγωνιζόμενο με την τεχνική κρυπτανάλυσης - επίθεσης γνωστών μηνυμάτων (Known Plaintext Attack) την οποία καλείται, εκμεταλλευόμενος και τις αδυναμίες του κρυπταλγόριθμου One-Time-Pad, να εφαρμόσει προκειμένου να ανακτήσει το flag το οποίο περιέχεται σε κάποιο από τα ciphertexts.

Σενάριο

Στο συγκεκριμένο challenge (*Silent Betrayal*) ο συνεργός του δράστη φαίνεται να έχει διαρρεύσει τα στοιχεία (password, employee id) του administrator. Η διακίνηση των στοιχείων έχει γίνει μέσω μηνυμάτων τα οποία έχουν κρυπτογραφηθεί με τον αλγόριθμο One-Time-Pad κάνοντας χρήση του ίδιου κλειδιού. Ο διαγωνιζόμενος έχοντας στη διάθεσή του τα κρυπτογραφημένα μηνύματα, που έχουν ανταλλάξει ο "δράστης" με τον συνεργό του, καλείται, εφαρμόζοντας τεχνικές κρυπτανάλυσης, να βρει μέρος του κλειδιού αποκρυπτογράφησης το οποίο θα τον βοηθήσει να ανακτήσει το flag που περιέχεται σε κάποιο από τα ciphertexts του challenge.Το challenge δίνει κάποια βοηθητικά στοιχεία (hints 1-3) όπως:

1. Το format του flag.
2. Κάθε κρυπτοκείμενο έχει το ίδιο μήκος με το κλειδί κρυπτογράφησης.
3. Τα αρχικά μηνύματα (plaintexts), από τα οποία έχουν παραχθεί τα ciphertexts, περιέχουν μια ή περισσότερες λέξεις με έντονη γραφή.



Εικόνα 39: Σενάριο cryptography challenge

Ενδεικτική Λύση

Για την επίλυση του challenge, θα πρέπει, επιπλέον, να λάβουμε υπόψη μας πως ο One-Time-Pad αλγόριθμος αποτελεί έναν συμμετρικό αλγόριθμο κρυπτογράφησης. Στους αλγόριθμους αυτής της κατηγορίας χρησιμοποιείται το ίδιο κλειδί για την κρυπτογράφηση και την αποκρυπτογράφηση των μηνυμάτων. Επιπλέον, μας δίνεται η πληροφορία πως όλα τα ciphertexts έχουν προκύψει από το ίδιο κλειδί το οποίο καθιστά τη διαδικασία της αποκρυπτογράφησης ευκολότερη. Γνωρίζοντας την λειτουργία του κρυπταλγόριθμου και λαμβάνοντας υπόψη το hint 3 μπορούμε να εξάγουμε μέρος του κλειδιού εφαρμόζοντας την τεχνική *Crib Dragging* [23] . Με τη συγκεκριμένη τεχνική "σέρνουμε" ένα γνωστό σύνολο χαρακτήρων (στη συγκεκριμένη περίπτωσή τις λέξεις με άσπρη έντονη γραφή) στο κρυπτογραφημένο κείμενο με στόχο την αποκάλυψη μέρος του αρχικού μηνύματος. Τα παρακάτω python script εκτελεί τη διαδικασία που μόλις αναφέραμε.

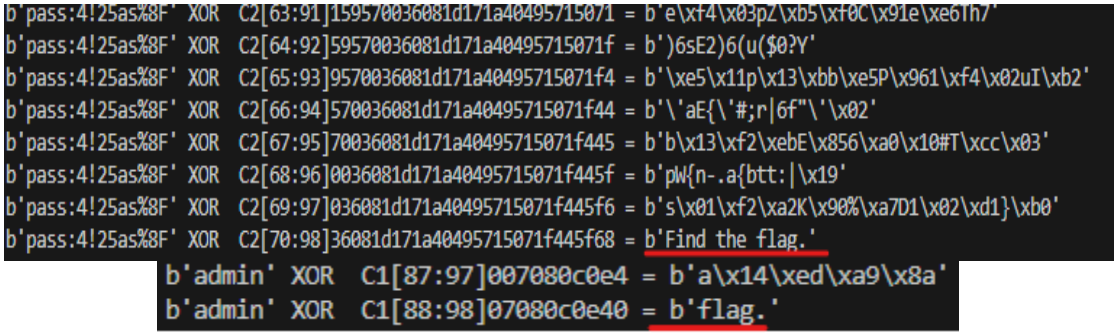
```python
import binascii

C1="0b0a17465228541d44453d4f014803490b095018071c450e45104a790e1b4443110f53351e011644540703000708 0c0e40"
C2="02031c0e1509005d01683c1d10001f061b4441060d45450201431a6a5a43175159570036081d171a40495715071f445f68"
C3="0b0a1c0201411553474c3808551446100111525400004c1b451f4238080e10173131013507491 07f245b1746230f554653"

cribs=['eid:43567289',"pass:4!25as%8F",'admin','flag{']

# Convert cribs into hex representation
cribs_hex_representation=[]
for crib in cribs:
    crib_hex=binascii.hexlify(crib.encode()).decode() # get hex representation of each 'crib'
    print(crib_hex)
    cribs_hex_representation.append(crib_hex)

# Perform 'crib dragging' technique XOR(ing) every 'crib' with every ciphertext (Ci)
ciphertexts_length=len(C1) # each ciphertexts has the same length
Cs=[C1,C2,C3]

for c in range(len(Cs)):

    for  hex_crib in cribs_hex_representation:
        for i in range(ciphertexts_length):
            if (len(Cs[c][i:len(hex_crib)+i]) < len(hex_crib)):
                break
            else:
                xor_result=bytes.fromhex(xor_hex_strings(hex_crib,Cs[c][i:len(hex_crib)+i]))
                ascii_hex_crib=bytes.fromhex(hex_crib)
                print(str(ascii_hex_crib)+" XOR "+" C"+str(c+1)+"["+str(i)+":"+str(len(hex_crib)+i)+"]"+ Cs[c][i:len(hex_crib)+i]+' = '+ str(xor_result))
```

Εικόνα 40: Crib dragging κώδικας python

```python
def xor_hex_strings(hex_str1, hex_str2):
    # Convert hex strings to integers
    int_value1 = int(hex_str1, 16)
    int_value2 = int(hex_str2, 16)

    # Perform XOR operation
    result = int_value1 ^ int_value2

    # Convert the result back to a hex string
    result_hex = hex(result)[2:]  # Remove '0x' from the beginning

    return result_hex
```

Εικόνα 41: Python κώδικας συνάρτησης xor_hex_strings()

Από τα αποτελέσματα που επιστρέφει ο παραπάνω κώδικας επικεντρωνόμαστε σε εκείνα που αναπαριστούν κάποια (αγγλική) φράση ή λέξη.



Εικόνα 42: Μέρος αποτελέσματος που επιστρέφει ο κώδικας

Το αποτέλεσμα, το οποίο παρουσιάζεται στην παραπάνω εικόνα, ενημερώνει τον χρήστη πως το **key[70:98]=”Find the flag.“**. Εξετάζοντας πιο προσεκτικά τα αποτελέσματα μπορούμε να εξάγουμε παρόμοιες πληροφορίες. Έτσι, συγκεντρώνουμε τις εξής πληροφορίες που μας βοηθούν να κατασκευάσουμε το κλειδί:

- **key[70:98]= ”Find the flag. “**
- **C2[70:98]=”pass:4!25as%8F”**
- **key[88:98] = ”flag”**
- **C1[88:98]=”admin”**
- **key[44:68] = ” key.You can”**
- **C2[44:68]=”eid:43567289”**
- **key[50:60] = ”y.You” (*επικαλύπτεται με το 5*)**
- **C3[50:60] = “flag{”**

Με βάση τα προηγούμενα, καταλήγουμε στο συμπέρασμα πως **key[44:68) || key[70:98] = “ key.You can?Find the flag.” *(1)*** (όπου ? άγνωστο key[68:70)) και πως το C3 περιέχει το flag του challenge (εφόσον περιέχει το string ‘flag{’). Άρα, μπορούμε, εφαρμόζοντας την πράξη **C3[44:98]) XOR (key[44:68) || 00 || key[70:98],** να λάβουμε μέρος του C3 που περιέχει το flag (το key[68:70) μπορούμε να το αντικαταστήσουμε με δύο μηδενικά). Έτσι λαμβάνουμε το παρακάτω αποτέλεσμα.

Εικόνα 43: Μέρος του flag

Το μόνο που απομένει είναι να βρούμε τον χαρακτήρα που βρίσκεται στη θέση του byte \x01 προκειμένου να έχουμε στην κατοχή μας το πλήρες flag. Αν παρατηρήσουμε την ***(1)*** μπορούμε να αντικαταστήσουμε το **?** με **’ ‘** (κενό) και να έχουμε το υποψήφιο **candidate_part_of_key = key[44:68) || key[70:98] = “ key.You can Find the flag.”** Στη συνέχεια, μπορούμε να βρούμε όλους τους δυνατούς συνδυασμούς (brute force) για τους οποίους ισχύει **candidate_part_of_key XOR partial_flag = C3[44:98]**. Τα προηγούμενα υλοποιούνται στο παρακάτω python script (*brute_force.py*).

```python
def xor_strings(s1, s2):
    return ''.join([chr(ord(a) ^ ord(b)) for a, b in zip(s1, s2)])


candidate_part_of_key= " key.You can Find the flag." #replace '?' with ' '
partial_flag = "lp flag{0tP_?sn't_P3rfEc4!}"
target_hex = '4c1b451f4238080e10173131013507491 07f245b1746230f554653' # C3[44:98]

#for star_char in range(32,127):
for question_char in range(257):

    result_hex = xor_strings(candidate_part_of_key, partial_flag.replace('?', chr(question_char))).encode('utf-8').hex()
        #print(result_hex)
    if result_hex == target_hex:
        print('" " ', chr(question_char))
```

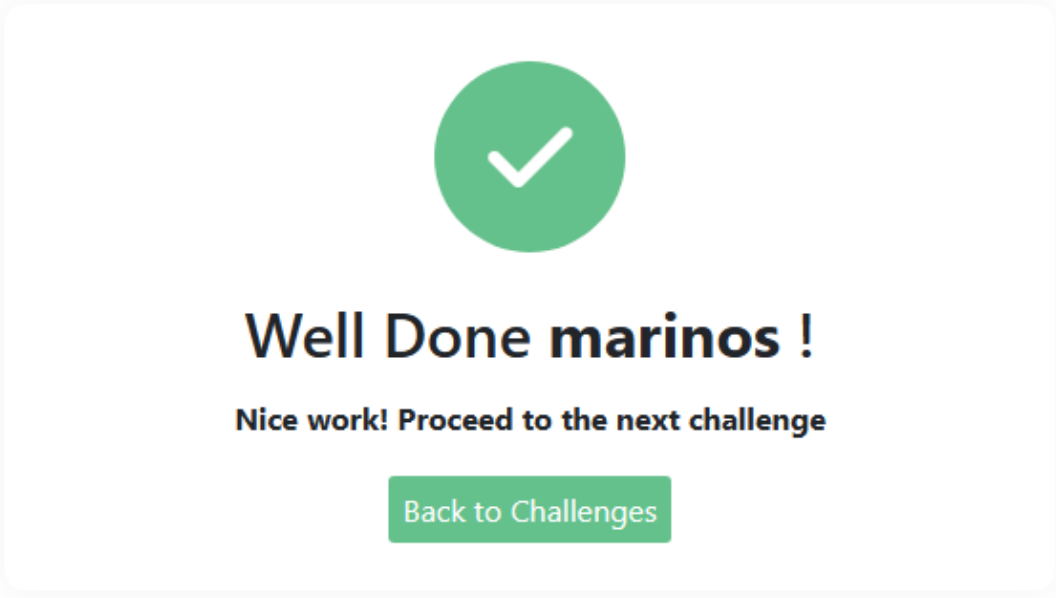
Εικόνα 44: brute_force.py script

```
PS C:\Users\Marinos> & C:/Python311/python.exe c:/Users/Marinos/Desktop/brute_force.py
" "  !
```

Εικόνα 45: : Αποτέλεσμα brute_force.py script

Εκτελώντας το brute_force.py βρίσκουμε πως ο χαρακτήρας που λείπει είναι το θαυμαστικό (!). Έτσι, μένει να ελέγξουμε αν ο χαρακτήρας (!) που βρήκαμε είναι ο σωστός τοποθετώντας το υποψήφιο flag στο αντίστοιχο πεδίο του challenge.

Εικόνα 46: Υποψήφιο flag crypto challenge



Εικόνα 47: Μήνυμα επιτυχούς ολοκλήρωσης crypto challenge

Steganography Challenge

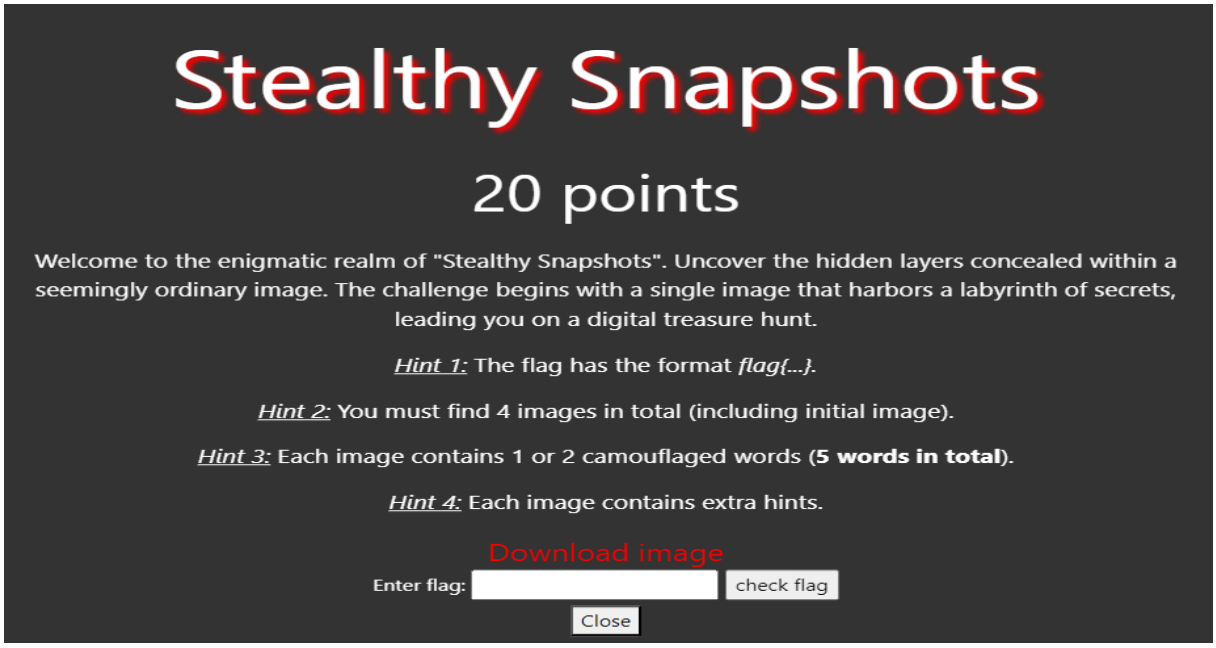
Στόχος

Στόχος του συγκεκριμένου challenge είναι να φέρει σε επαφή τον διαγωνιζόμενο με βασικά εργαλεία στεγανογραφίας και εξόρυξης αρχείων τα οποία καλείται να χρησιμοποιήσει προκειμένου να εξάγει πληροφορίες οι οποίες βρίσκονται κρυμμένες σε κάθε εικόνα. Επιπλέον, το challenge, απαιτεί από τον διαγωνιζόμενο να εφαρμόσει απλές password cracking τεχνικές προκειμένου να ανακτήσει το flag του challenge.

Σενάριο

Στο challenge (*Stealthy Snapshots*) ο διαγωνιζόμενος καλείται να αποκαλύψει τα κρυφά επίπεδα (κρυμμένες εικόνες) που κρύβονται μέσα σε μια, φαινομενικά, συνηθισμένη εικόνα. Το challenge ξεκινάει με μια εικόνα η οποία φιλοξενεί έναν "λαβύρινθο" από μυστικά τα οποία οδηγούν σε ένα κυνήγι θησαυρού με έπαθλο το flag του challenge. Το challenge δίνει κάποια βοηθητικά στοιχεία (hints 1-4) όπως:

1. Το format του flag.
2. Πρέπει να βρεθούν συνολικά 4 εικόνες (μαζί με την αρχική εικόνα).
3. Κάθε εικόνα περιέχει 1 ή 2 κρυμμένες λέξεις.
4. Κάθε εικόνα περιέχει επιπλέον χρήσιμες πληροφορίες.



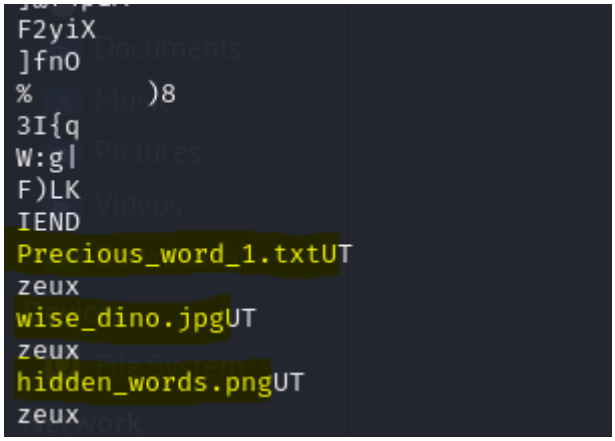
Εικόνα 48: Σενάριο steganography challenge

Ενδεικτική Λύση

Αρχικά, έχοντας υπόψη το hint 4,ανοίγουμε την εικονα μας με κάποιον image viewer. Παρατηρώντας την εικόνα, αρχικά με γυμνό μάτι, δεν είμαστε σε θέση να διακρίνουμε κάποιου είδους πληροφορία η οποία θα μας φανεί χρήσιμη .

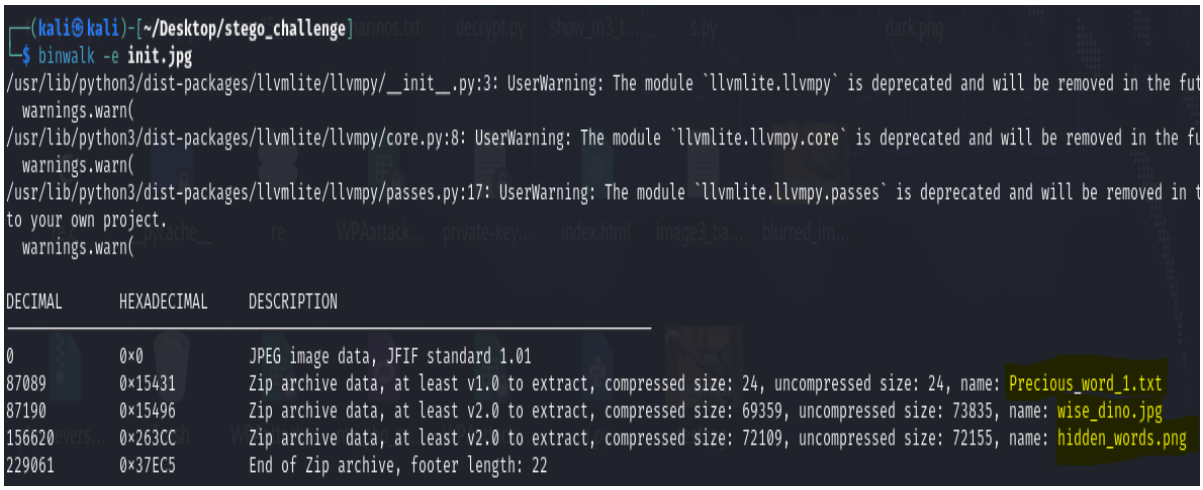
Εικόνα 49: Αρχική εικόνα (init.jpg).

Επόμενο βήμα, είναι να χρησιμοποιήσουμε την εντολή *strings* με την οποία μπορούμε να διαβάσουμε συμβολοσειρές (strings) οι οποίες περιέχονται (embedded) στην εικόνα init.jpg.

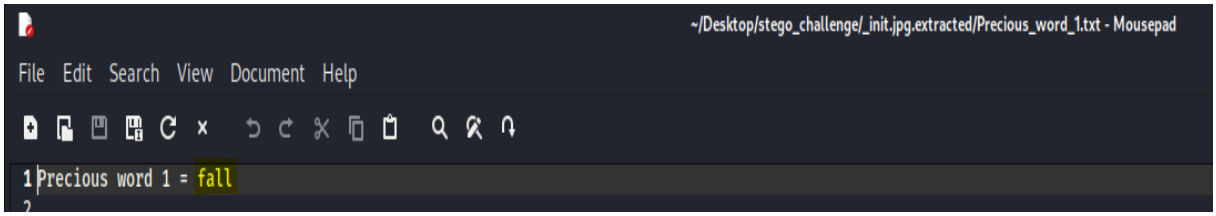


Εικόνα 50: : Embedded strings

Παρατηρώντας τα αποτελέσματα που επιστρέφει η εντολή μπορούμε να διαπιστώσουμε πως η εικόνα init.jpg "κρύβει", τουλάχιστον, δύο επιπλέον εικόνες (wise_dino.jpg και hidden_words.png) και ένα αρχείο txt στο οποίο περιέχεται μια από τις τέσσερις κρυμμένες λέξεις (hint 2 και hint 3). Προκειμένου να εξάγουμε όλα τα αρχεία που περιέχονται στην εικόνα init.jpg θα χρησιμοποιήσουμε το εργαλείο *binwalk* [24].



Εικόνα 51: Αποτελέσματα binwalk



Εικόνα 52: Πρώτη κρυμμένη λέξη

Έχοντας στην κατοχή μας την πρώτη κρυμμένη λέξη (“**fall**”) μπορούμε να εξετάσουμε την εικόνα hidden_words.png ανοίγοντάς την, αρχικά, σε έναν image viewer. Η εικόνα από μόνη της δεν μας παρέχει κάποια πληροφορία που μπορούμε να διακρίνουμε με γυμνό μάτι. Παρόλα αυτά, εικάζουμε, βασιζόμενοι στην ονομασία της πως ίσως “κρύβει” παραπάνω από μια λέξεις.

Εικόνα 53: Εικόνα hidden_words.png

Μπορούμε να αντλήσουμε κάποια πληροφορία από τα metadata της εικόνας με τη βοήθεια του εργαλείου *exiftool*. Στα αποτελέσματα που επιστρέφονται το ενδιαφέρον μας στρέφεται στο “Comment” tag το οποίο περιέχει το hint *“poor some light on me”*. Το τελευταίο, φανερώνει έμμεσα πως πρέπει να επεξεργαστούμε την εικόνα προκειμένου να μας εμφανιστεί κάποιου είδους πληροφορία.

```
┌──(kali㉿kali)-[~/Desktop/stego_challenge/_init.jpg.extracted]
└─$ exiftool hidden_words.png
ExifTool Version Number         : 12.55
File Name                       : hidden_words.png
Directory                       : .
File Size                       : 72 kB
File Modification Date/Time     : 2023:12:13 14:55:07-05:00
File Access Date/Time           : 2024:01:19 12:22:52-05:00
File Inode Change Date/Time     : 2024:01:19 11:36:55-05:00
File Permissions                : -rw-------
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 303
Image Height                    : 405
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Gamma                           : 2.2
Pixels Per Unit X               : 11811
Pixels Per Unit Y               : 11811
Pixel Units                     : meters
Comment                         : Hint: pour some light on me
Image Size                      : 303x405
Megapixels                      : 0.123
```

Εικόνα 54: Αποτελέσματα ExifTool (hidden_words.png)

Στη συγκεκριμένη περίπτωση μπορούμε να χρησιμοποιήσουμε εργαλεία όπως το *stegsolve* [25] η websites όπως το *Aperi'Solve* [26] προκειμένου να επεξεργαστούμε την εικόνα και να αντλήσουμε τυχόν χρήσιμη πληροφορία Όπως φαίνεται και παρακάτω, η εικόνα hidden_words.png περιέχει επιπλέον δύο κρυμμένες λέξεις ("**aces**","**species**").

Εικόνα 55: Επεξεργασμένη hidden_words.png

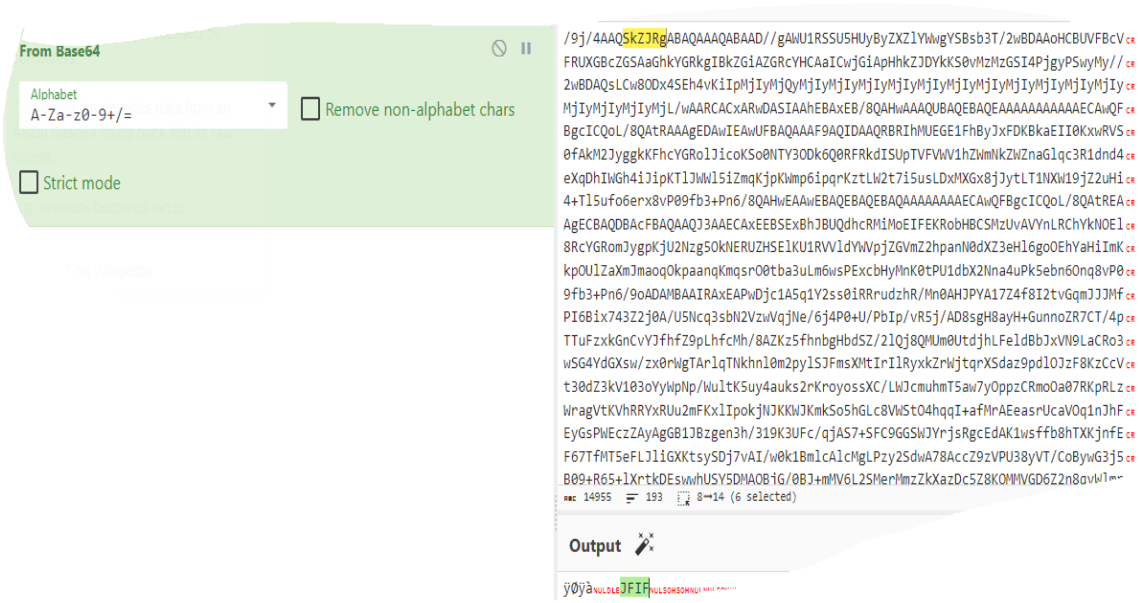
Η επόμενη εικόνα (wise_dino.jpg) παραπέμπει τον διαγωνιζόμενο να δώσει ιδιαίτερη έμφαση "στα σχόλια". Χρησιμοποιώντας το εργαλείο *ExifTool* παρατηρούμε πως στο "Comment" tag υπάρχει ένα, αρκετά μεγάλου μήκους, string σε κωδικοποίηση base64 το οποίο, αν το αποκωδικοποιήσουμε, καταλαβαίνουμε πως αποτελεί κάποια εικόνα (ένδειξη JFIF). Επιπλέον,

μπορούμε να διακρίνουμε μια ακόμα κρυμμένη λέξη (“**sunt**”) η οποία βρίσκεται στο tag “Description‘.

Εικόνα 56: Εικόνα wise_dino.jpg

```
└─$ exiftool wise_dino.jpg
ExifTool Version Number         : 12.55
File Name                       : wise_dino.jpg
Directory                       : .
File Size                       : 74 kB
File Modification Date/Time     : 2023:12:13 14:39:39-05:00
File Access Date/Time           : 2024:01:19 12:22:52-05:00
File Inode Change Date/Time     : 2024:01:19 11:36:55-05:00
File Permissions                : -rw-------
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
XMP Toolkit                     : Image::ExifTool 12.55
Description                     : Precious_word_2=sunt
Comment                         : /9j/4AAQSkZJRgABAQAAAQABAAD//gAWU1RSSU5HUyByZXZlYWwgYSBsb3T/2wBDAAoHC
IpMjIyMjQyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy.MjIyMjIyMjIyMjL/wAARCACxARwDASIAAhEBAxEB/8QAHwAA
fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqK
wQFBgcICQoL/8QAtREA.AgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl.8RcYGR
wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP0.9fb3+Pn6/9oADAMBAAIRAxEAPwDjc1A5q1Y2ss0iRRrudzhR/Mn0AHJPYA17Z4
fZ9pLhfcMh/8AZKz5fhnbgHbdSZ/2lQj8QMUm0UtdjhLFeldBbJxVN9LaCRo3.wSG4YdGXsw/zx0rWgTArlqTNkhnl0m2pylSJFmsXM
KpRLz.WragVtKVhRRYxRUu2mFKxlIpokjNJKKWJKmkSo5hGLc8VWStO4hqqI+afMrAEeasrUcaVOq1nJhF.EyGsPWEczZAyAgGB1JBz
78AccZ9zVPU38yVT/CoBywG3j5.B09+R65+lXrtkDEswwhUSY5DMAOBjG/0BJ+mMV6L2SMerMmzZkXazDc5Z8KOMMVGD6Z2n8qvWlmr
hgMcbRjkdOcVoxxOLqTdjcu0c5wpK9BnjIAx+J9ahSzcgEJ.njrjv3pqyE7nr/gXwuLSPzZAPPkXn/pmh58se/TJ9RjtXQ3N+F70zUb
/oa8oOoy7drN/n+tXY7v93tLEgDk/wA6lykONKnvY7K7SO6j3R/fTJUeo7p/h7/WsiEc.Vk+FtRwRzjJ/WutubRWUyRnPdk9PUj29q5
gV53r2objnr8.xGM9P8jFThKbnJ9hTlZG0jITw2T7VpWwrktJck9fw9PbmuwtlziumtHkCErq5dQU/ZTo0qbFcbki.7kKLinsaTNNJq
Y437E2jOBycDH/Ac1JqdzEFZVbgNkIOmQ.FXdkdc8ntjHvT7bLPxkIpUtgdcDO0D3Yn8KaLLzCDjru25HUAhc46/eJAB/u16OnUw6EW
nJ+YMBxgAn1FZTleWppBWRxl2WBxv3EMXYjBEhbBIz/d7c+h9aZDJO.ByQAeVxjBU9D1qZlTYdrqSp27ySN7KvJJPGM56elZTJIMAIC
yTjJHB5Ht/jWhKwxtHIPX6Vj24JPNdDDp6lQVJDfWsZHRGLktClaHZIwTIX0z3wM/rW1pWulJdp.PBx+FYVu4WRlc/Nn8885qTV51YR
```

Εικόνα 57: Αποτελέσματα ExifTool (wise_dino.jpg)

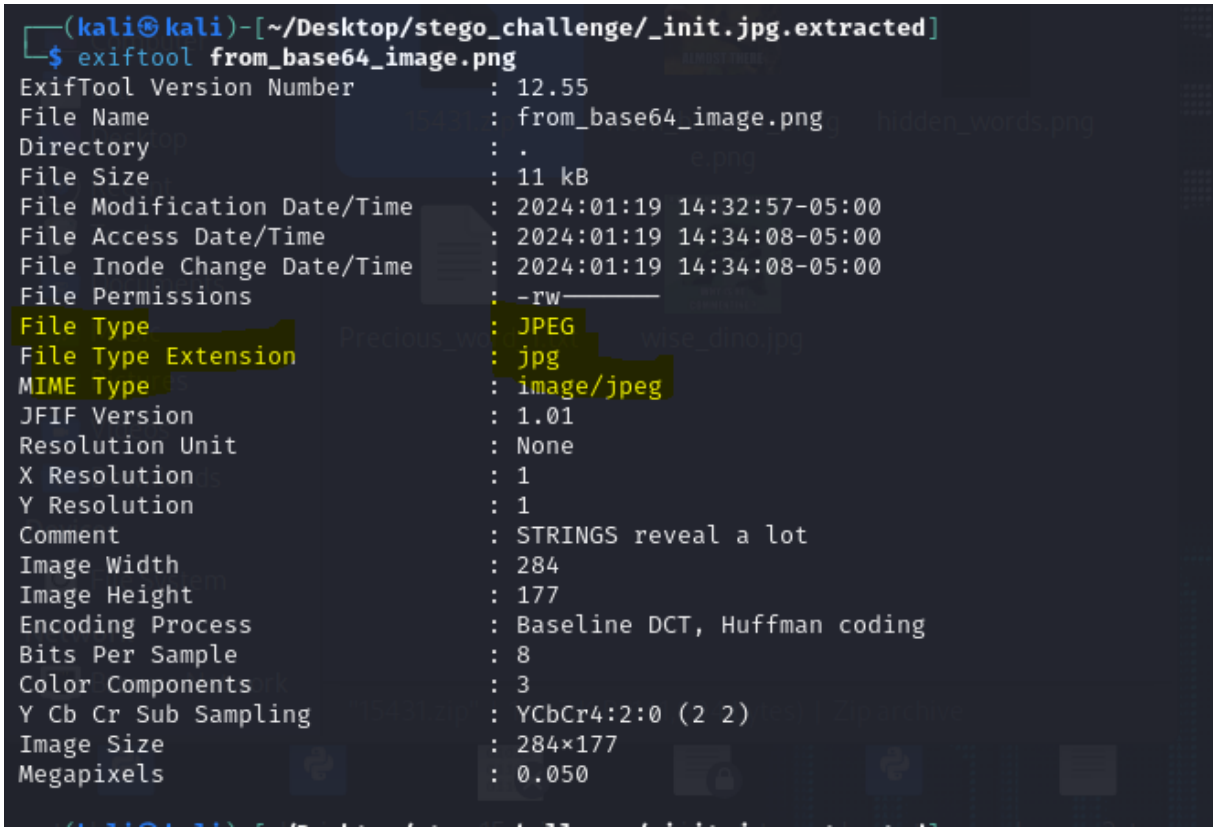


Εικόνα 58: Αποκωδικοποίηση base64 string του "Comment" tag (wise_dino.jpg)

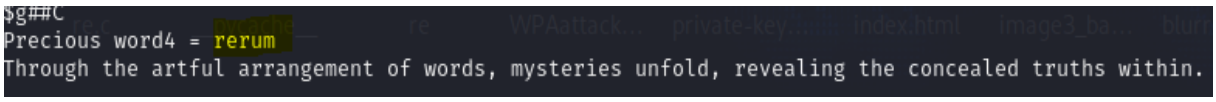
Στη συνέχεια μπορούμε να ανακτήσουμε την εικόνα (έστω from_base64_image.png) από το base64 string μεταβαίνοντας σε έναν base64-to- image-converter [27].

Εικόνα 59:Ανακτημένη εικόνα (from_base64_image.png)

Εξετάζοντας τα metadata της εικόνας με τη βοήθεια του exiftool, παρατηρούμε πως η εικονα μας είναι τύπου jpg και όχι png. Γι αυτό αλλάζουμε το file extension από png σε jpg. Επιπλέον, το "Comment" tag μας παραπέμπει να χρησιμοποιήσουμε την εντολή strings για να αντλήσουμε κάποιου είδους πληροφορία. Εκτελώντας το τελευταίο λαμβάνουμε μια επιπλέον κρυμμένη λέξη ("**rerum**") και ένα hint.

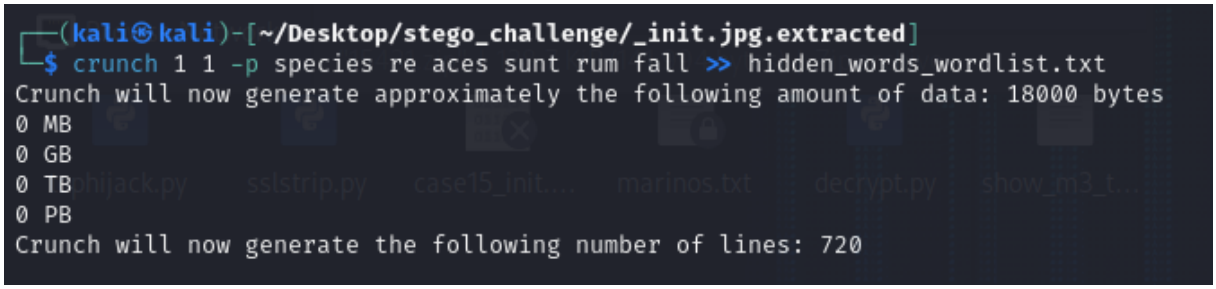


Εικόνα 60: : Αποτελέσματα ExifTool (from_base64_image.png)



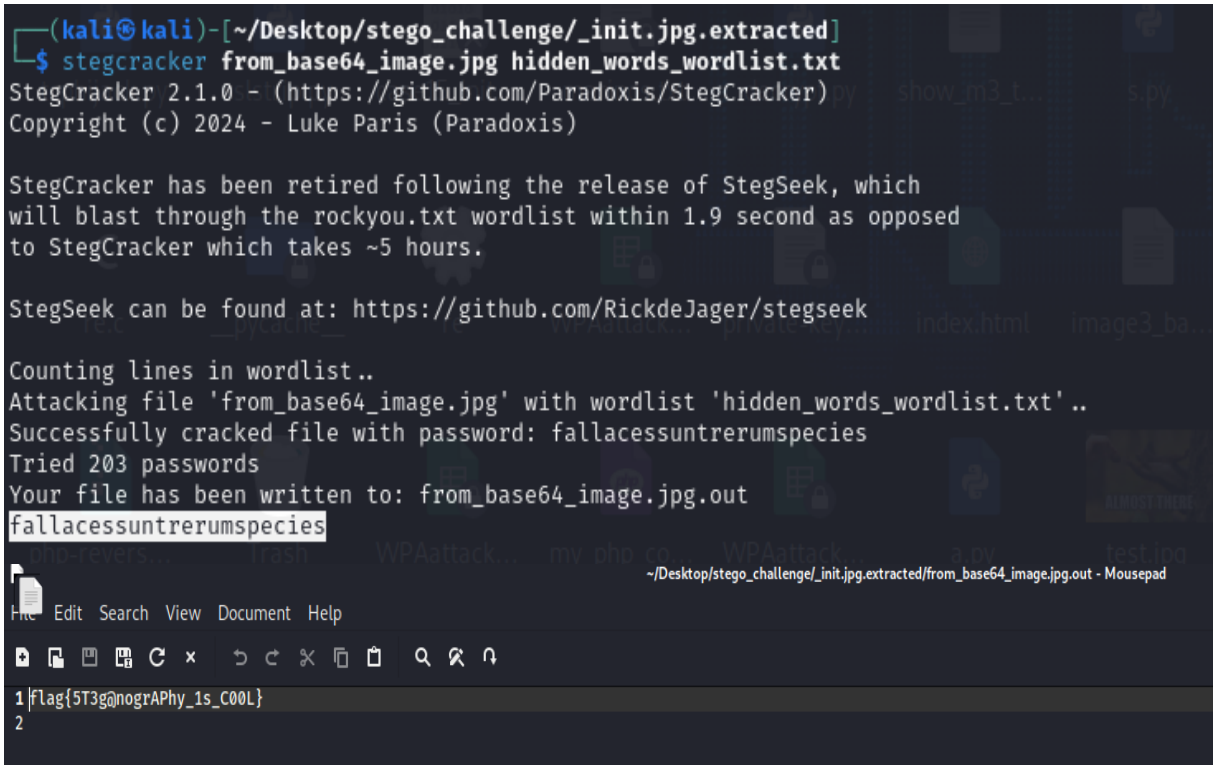
Εικόνα 61:: Αποτελέσματα strings (from_base64_image.png)

Η τελευταία πρόταση (“*Through the artful…”*) υπονοεί πως η εικόνα from_base64_image.png περιέχει το flag το οποίο είναι κρυμμένο και προστατεύεται με κωδικό. Ο κωδικός αυτός αποτελείται από τις κρυμμένες λέξεις τοποθετημένες στη σωστή σειρά. Παρόλα αυτά δεν μας δίνεται κάποια πληροφορία για την σειρά με την οποία πρέπει να τοποθετηθούν οι λέξεις. Το σκεπτικό μας για τη λύση του συγκεκριμένου προβλήματος είναι να δημιουργήσουμε μια wordlist η οποία περιέχει όλους τους πιθανούς συνδυασμούς των λέξεων που έχουμε βρει και στη συνέχεια να εκτελέσουμε κάποιου είδους dictionary επίθεση για να ανακτήσουμε το flag. Για τη δημιουργία της wordlist μπορούμε να χρησιμοποιήσουμε το εργαλείο *crunch* [28].



Εικόνα 62:Δημιουργία wordlist (hidden_words_wordlist.txt)

Τέλος, κάνοντας χρήση του *stegcracker* [29] σε συνδυασμό με την wordlist που δημιουργήσαμε προηγουμένως είμαστε σε θέση να ανακτήσουμε το flag.



Εικόνα 63:: Steganography flag challenge

Συμπεράσματα

Μέσω της συγκεκριμένης διπλωματικής μας δόθηκε η ευκαιρία να δημιουργήσουμε challenges για έναν Jeopardy-style Capture the Flag διαγωνισμό όσο και να υλοποιήσουμε ένα υποτυπώδες σύστημα το οποίο "φιλοξενεί" έναν διαγωνισμό αυτού του είδους. Επιπλέον, μας δόθηκε η δυνατότητα να ασχοληθούμε και με πρακτικές οι οποίες συμβάλλουν στην προστασία (iptables) και παρακολούθηση (inotifywait) σημαντικών αρχείων (όπως credentials Β.Δ) που βρίσκονται αποθηκευμένα στον server ("JeopardyCTF server") ο οποίος "φιλοξενεί" τα challenges και τη web εφαρμογή. Μέσω των συγκεκριμένων challenges, αναδείξαμε θέματα τα οποία έχουν να κάνουν με την απόκρυψη δεδομένων σε μια εφαρμογή και τους τρόπους με τους οποίους μπορούμε να τα εξάγουμε (reverse engineering challenge), την συγκάλυψη και τον τρόπο ενσωμάτωσης αρχείων σε άλλα αρχεία μαζί με αντίστοιχες μεθόδους με τις οποίες μπορούμε να τα ανακαλύψουμε (steganography challenge), θίξαμε θέματα ασφαλείας τα οποία αφορούν εφαρμογές στον Παγκόσμιου Ιστού (web exploitation challenge) και θέματα κρυπτογραφίας.

Βιβλιογραφία

1. Svabensky, V. Celeda, P. Vykopal, J. Brisakova, S. “Cybersecurity knowledge and skills taught in capture the flag challenges” Comput. Secur. 2021, 102, 102154.Chung-Kuan Chen “CTF: Alternative Training for Offensive Security” .
2. K. Bock, G. Hughey, and D. Levin, “King of the hill: A novel cybersecurity competition for teaching penetration testing,” in 2018 USENIX Workshop on Advances in Security Education (ASE 18).
3. Available: https://aws.amazon.com/what-is/lamp-stack.
4. “What is LAMP Stack” [Online]. Available: https://aws.amazon.com/what-is/lamp-stack.
5. “Apache http server project” [Online]. Available: https://httpd.apache.org/.
6. MySQL official website [Online]. Available: https://www.mysql.com.
7. PHP official website [Online]. Available: https://www.php.net.
8. “The betfilter.org iptables project” [Online]. Available: https://www.netfilter.org/projects/iptables/index.html.
9. “inotifywait - Linux man page” [Online]. Available: https://linux.die.net/man/1/inotifywait.
10. “Apache http server project” [Online]. Available: https://httpd.apache.org/docs/2.4/howto/htaccess.html.
11. “ptrace - Linux manual page” [Online]. Available : https://man7.org/linux/man-pages/man2/ptrace.2.html.
12. “File Inclusion Vulnerability: What are they and how do they work?” [Online]: https://brightsec.com/blog/file-inclusion-vulnerabilities.
13. “Path Traversal” [Online]: https://owasp.org/www-community/attacks/Path_Traversal.

14. D.Rijmenants, "One-time pad" [Online]: Available https://www.ciphermachinesandcryptology.com/en/onetimepad.htm.
15. Zaidoon Kh. AL-Ani, A.A.Zaidan, B.B.Zaidan and Hamdan.O.Alanazi "Overview: Main Fundamentals for Steganography" JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617.
16. Steghide [Online] : Available https://github.com/StefanoDeVuono/steghide.
17. "ExifTool by Phil Harvey" [Online]. Available: https://exiftool.org/
18. ltrace(1) – Linux manual page [Online]. Available: https://man7.org/linux/man- pages/man1/ltrace.1.html.
19. ltrace(1) – Linux manual page [Online]. Available: https://man7.org/linux/man- pages/man1/strace.1.html.
20. "Ghidra Software Reverse Engineering Framework," [Online]. Available: https://github.com/NationalSecurityAgency/ghidra.
21. CyberChef website [Online]. Available: https://gchq.github.io/CyberChef/.
22. pentestmonkey [Online]. Available: https://pentestmonkey.net/.
23. "Toying with Cryptography: Crib Dragging" [Online]. Available: https://samwho.dev/blog/toying-with-cryptography-crib-dragging/.
24. Binwalk [Online]. Available: https://github.com/ReFirmLabs/binwalk.
25. stegsolve [Online]. Available: https://github.com/zardus/ctf-tools/blob/master/stegsolve/install
26. Aperi'Solve [Online]. Available: https://www.aperisolve.com/.
27. Code Beautify, Base64 to Image Converter [Online]. Available: https://codebeautify.org/base64-to-image-converter.
28. Crunch [Online]. Available: https://www.kali.org/tools/crunch/
29. Stegcracker [Online]. Available: https://github.com/Paradoxis/StegCracker.