ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ανάπτυξη διεπαφής απομακρυσμένων δοκιμών διείσδυσης σε εργαστηριακό περιβάλλον

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιάσονα Αργιαντζή

321/2019014

Επιβλέπων: Δρ. Στεργιόπουλος Γεώργιος

Μέλη εξεταστικής επιτροπής: 1) Δρ. Καρύδα Μαρία

2)Δρ. Κοκολάκης Σπύρος

Σάμος , 8 Οκτωβρίου ,2024

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Ιάσονα Αργιαντζή που την εκπόνησε .

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου , κύριο Στεργιόπουλο Γεώργιο, για την επίβλεψη της διπλωματικής μου εργασίας και την καθοδήγησή καθ' όλη την διάρκεια της συγγραφής. Ευχαριστώ τον καθηγητή μου για το βοηθητικό υλικό που μου παρείχε κατά την διάρκεια της εργασίας, καθώς και τον χρόνο που αφιέρωσε , δίνοντας μου την δυνατότητα να ολοκληρώσω την εργασία. Θέλω να ευχαριστήσω όλους μου τους καθηγητές ξεχωριστά. Ακόμη, θα ήθελα να ευχαριστήσω την οικογένεια μου που στάθηκε δίπλα μου σε όλο αυτό το ταξίδι και με υποστήριζε συνέχεια για κάθε μου απόφαση. Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου στην Σάμο ,καθώς έκαναν την φοιτητική μου εμπειρία ακόμα πιο όμορφη.

Περίληψη

Το θέμα της παρούσας διπλωματικής εργασίας, αφορά τον αυτοματοποιημένο χειρισμό των Virtual Machines (VMs) μέσω της εφαρμογής (Oracle Virtual Box) και την πρόσβαση έτοιμων σχεδιασμένων VMs για τον πειραματισμό και την εκμάθηση του εκάστοτε χρήστη .

Με σκοπό ο χρήστης να αποκτήσει πρόσβαση, έχει φτιαχτεί μια RESTful υπηρεσία ιστού , όπου μπορεί να συνδεθεί από την διεπαφή login. Από εκεί, προσφέρεται η δυνατότητα να λάβει τα διαθέσιμα (VMs) .

Τα VMs, είναι επίτηδες ευπαθή (Web Applications). Ο χρήστη, καλείται να επιλύσει διάφορα θέματα ασφάλειας που μπορεί να προκύψουν από ένα (Web Application) όπως (SQL injection , XSS attacks , κ.α.) και να εκμεταλλευτεί αυτού του είδους τα exploits . Τέλος, τα (Web Applications) παρέχονται έτοιμα από το OWASP. Συγκεκριμένα, είναι το WebGoat[1] και το Juice Shop[2].

Λέξεις-κλειδιά :Virtual Box , αυτοματοποιημένη διαδικασία VBoxManage , Python Flask , REST , WebGoat OWASP , Juice Shop OWASP , Vulnerable Applications.

1.WebGoat (OWASP)[https://owasp.org/www-project-webgoat/]

2.Juice Shop (OWASP) [https://owasp.org/www-project-juice-shop/]

Abstract

The subject of this dissertation concerns the automated management of Virtual Machines (VMs) through the Oracle VirtualBox application, as well as the access to pre-configured VMs for experimentation and learning by the user.

In order to grant access to the user, a RESTful web service has been created, allowing the user to connect via a login interface, from which they will be able to access the available VMs.

The VMs are deliberately vulnerable (Web Applications), and the user is called upon to solve various security issues that may arise from a Web Application, such as SQL injection, XSS attacks, etc., and to exploit these kinds of vulnerabilities. Finally, the Web Applications are provided ready-to-use from OWASP, specifically [1]WebGoat and [2]Juice Shop.

Keywords: VirtualBox, automated process VBoxManage, Python Flask, REST, WebGoat OWASP, Juice Shop OWASP, Vulnerable Applications.

1.WebGoat (OWASP)[https://owasp.org/www-project-webgoat/]

2.Juice Shop (OWASP) [https://owasp.org/www-project-juice-shop/]

Περιεχόμενα

Περίληψη....4
Abstract....5
1. Εισαγωγή....9
1.1 Πληροφορίες για τα συστήματα της διπλωματικής....9
1.2 Σκοπός της Διπλωματικής....10
1.3 Δομή Διπλωματικής....11
2. Εργαλείο VM....12
2.1 Τι είναι το VM....12
2.2 Με ποιους τρόπους μπορεί να χρησιμοποιηθεί το VM....13
3 Εργαλείο Flask....14
3.1 Εισαγωγή....14
3.2 Τι είναι μια RESTful υπηρεσία....15
3.3 Πώς είναι ένα Response/Structure Format , Standards Methods και Status Code στο RESTful API....16
3.4 Ποια εργαλεία χρησιμοποιήθηκαν....18
4 Θεωρία για τα εργαλεία που χρησιμοποιήθηκαν....19
4.1 Εισαγωγή στο WebGoat....19
4.1.1 Ποιες επιθέσεις μπορούν να πραγματοποιηθούν στο WebGoat....19
4.2 Εισαγωγή στο Juice Shop....20
4.2.1 Ποιες επιθέσεις μπορούν να πραγματοποιηθούν στο Juice Shop....20
4.3 RESTful API μέσω python (Flask)....21
4.3.1 Θετικά για την χρήση Flask....21
4.3.2 Χρήση του Flask και ένα απλό παράδειγμα....22
4.3.3 Flask vs Django....23
5 Βασικές Επιθέσεις....24
5.1 SQL Injections....24
5.2 Broken Access Control....26
5.3 Cryptography failures....26
5.4 Cross-Site Scripting (XSS)....27
5.5 Broken Authentication....28
6 Υλοποίηση της εργασίας....29
6.1 VM (WebGoat , JuiceShop) Setup....29

6.2 Python Flask Web Application (Setup / Code) 33
6.2.1 Λειτουργία App (Login / Start VM) 37
6.2.2 Δημιουργία SnapShots 44
7 Μελλοντικές αναβαθμίσεις 46
8 Βιβλιογραφία 47

Περιεχόμενα εικόνων

Εικόνα 1 Αρχιτεκτονική VM 12
Εικόνα 2 Παράδειγμα Flask 22
Εικόνα 3 Παράδειγμα Flask Webpage 23
Εικόνα 4 Web Goat Περιβάλλον (VM) 31
Εικόνα 5 Περιεχόμενο README 31
Εικόνα 6 Εικόνα Firefox – Webgoat 32
Εικόνα 7 Βασική Πλατφόρμα Web Goat 32
Εικόνα 8 Ενεργοποίηση venv 34
Εικόνα 9 Αρχιτεκτονική (folder - flask) 34
Εικόνα 10 Main(app.py) 36
Εικόνα 11 Στοιχεία app.py 37
Εικόνα 12 ΦΟΡΜΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ REST ΜΗΝΥΜΑΤΑ 37
Εικόνα 13 Σύνδεση με Βάση Δεδομένων 38
Εικόνα 14 Κώδικας login 39
Εικόνα 15 login(401) μηνύματα και αποτελέσματα 39
Εικόνα 16 Login(200) Μήνυμα και αποτελέσματα 40
Εικόνα 17 Περίπτωση που ο χρήστη δεν ακολουθεί τισ οδηγίες 42
Εικόνα 18 Ξεκινάει ο χρήστης το vm με το (.ova) 42
Εικόνα 19 Δημιουργία Snap Shot 44
Εικόνα 20 Αποτέλεσμα από το SnapShot 45

Περιεχόμενα Πινάκων

Πίνακας 1 Αρκτικόλεξο 8
Πίνακας 2 Status Code [5] 17
Πίνακας 3 Χαρακτηριστικά VM(WebGoat) 29

ΠΙΝΑΚΑΣ 1 ΑΡΚΤΙΚΟΛΕΞΟ

VM	Virtual Machine
XSS	Cross-Site Scripting
DDoS	Distributed Denial of Service
VMM	virtual machine monitor
API	Application Program Interface
URI	Uniform Resource Locator
SDKs	Software Development Kits
CSRF	Cross-Site Request Forgery
IDOR	Insecure Direct Object References
XXE	XML External Entity
CORS	Cross-Origin Resource Sharing
VENV	Virtual Environment

1. Εισαγωγή

1.1 Πληροφορίες για τα συστήματα της διπλωματικής

Με την εκρηκτική ανάπτυξη του Διαδικτύου στα τέλη της δεκαετίας του 1990 και τις αρχές της δεκαετίας του 2000, η χρήση των ιστοσελίδων και των διαδικτυακών υπηρεσιών έγινε ευρέως διαδεδομένη. Ωστόσο, μαζί με τη ραγδαία αυτή εξέλιξη, αναπτύχθηκαν και οι πρώτες απειλές για την ασφάλεια των συστημάτων και των δεδομένων που διακινούνται μέσω του διαδικτύου . Οι επιθέσεις σε web επίπεδο άρχισαν να εμφανίζονται σχεδόν ταυτόχρονα με την επέκταση της χρήσης του web και αποτελούν ένα σημαντικό κομμάτι της ιστορίας της ασφάλειας πληροφοριακών συστημάτων.

Η άνοδος των δυναμικών επιθέσεων έγινε την περίοδο (2000 – 2010) , όπου εμφανίστηκαν επιθέσεις , SQL Injection , XSS επιθέσεις και DDos επιθέσεις. Τέλος από την περίοδο 2010 μέχρι και σήμερα οι επιθέσεις έχουν γίνει ποιο σύνθετες , επιθέσεις όπως **ransomware** , **phishing** .

Για την αντιμετώπιση των παραπάνω επιθέσεων θα έπρεπε να εκπαιδευτούν τα άτομα που θέλουν να ασχοληθούν με την κυβερνοασφάλεια με ασφαλή και πάνω από όλα νόμιμο τρόπο , έτσι έρχεται η χρήση του (**VM**) .

Η έννοια των **VM** άρχισε να αναπτύσσεται τη δεκαετία του 1960, όταν η IBM εργάστηκε πάνω στην τεχνολογία χρονικού διαμερισμού (time-sharing). Αυτή η τεχνική επέτρεπε σε πολλούς χρήστες να χρησιμοποιούν ταυτόχρονα έναν υπολογιστή, μοιράζοντας τη χρήση των πόρων του συστήματος. Το IBM CP-40 (1967) ήταν το πρώτο λειτουργικό σύστημα που υποστήριξε τη λειτουργία πολλαπλών εικονικών περιβαλλόντων, τα οποία μπορούσαν να τρέχουν ανεξάρτητα στον ίδιο φυσικό υπολογιστή. Με την ίδρυση της **VMware** το 1998. Η VMware ανέπτυξε την τεχνολογία που επέτρεπε στους χρήστες των προσωπικών υπολογιστών να εκτελούν πολλαπλά λειτουργικά συστήματα σε ένα μόνο μηχάνημα. Το **VMware Workstation** (1999) ήταν το πρώτο εμπορικό προϊόν που επέτρεπε την εκτέλεση πολλαπλών VM σε προσωπικούς υπολογιστές, ανοίγοντας τον δρόμο για τη διάδοση της εικονικοποίησης και στους servers.

1.2 Σκοπός της Διπλωματικής

Με την ραγδαία εξέλιξη της τεχνολογίας, μέσω της οποίας προστίθενται ειδικά στον τομέα του Web Application ,περισσότερες λειτουργίες και υπηρεσίες , είναι αναγκαίο ένας άνθρωπος που θέλει να ακολουθήσει την επιστήμη της κυβερνοασφάλειας να ξεκινήσει από βασικές έννοιες και από βασικούς τρόπους ελέγχου για την εύρεση βασικών λαθών που μπορεί να υπάρχει σε μια Ιστοσελίδα . Πρέπει, να ενημερώνεται συνεχώς καθώς ένας βασικός κανόνας που υπάρχει στην ασφάλεια είναι όσο πιο πολλές υπηρεσίες υποστηρίζουν ένα Web Application , (δηλαδή έχει μεγάλη επιφάνεια για να εντοπίσει μια ευπάθεια του συστήματος), άλλο τόσο αυξάνει την πιθανότητα επίθεσης στο σύστημα από έναν επιτιθέμενο εκμεταλλευόμενος ένα exploit ή ακόμα χειρότερα να υπάρξει επίθεση (0-day attack).

Για έναν άνθρωπο, που θέλει να μάθει τους τρόπους με τους οποίους μπορεί να προστατεύει έναν χρήστη από διαδικτυακή επίθεση , πρώτα θα πρέπει να γνωρίζει τον τρόπο με τον οποίο γίνεται . Επομένως, για να δοκιμάσει τις γνώσεις του και πρακτικά να μπορεί να δημιουργήσει ένα εικονικό σενάριο μέσα σε ένα VM , μπορεί μέσα να υπάρχει ένα ψεύτικο Web Application επίτηδες πειραγμένο για την εκμάθηση κάποιον γνωστών μέχρι και πολύ δύσκολων επιθέσεων.

Με αυτόν τον τρόπο και ο χρήστης που θέλει να πειραματιστεί είναι ασφαλισμένος νομικά , καθώς δεν επιχειρεί επίθεση σε πραγματικές υπηρεσίες και προστατεύει και τον ίδιο του τον υπολογιστή. Αυτό συμβαίνει, καθώς ακόμη και να καταστραφεί το VM ή πάθει οποιαδήποτε ζημία, έχει την δυνατότητα να κλείσει το VM και ο βασικός του υπολογιστής να παραμείνει άθικτος.

Με σκοπό να στηθεί όμως όλο αυτό το σύστημα απαιτείται εμπειρία , καθιστώντας έτσι δύσκολη την εφαρμογή για άτομα που μόλις μπαίνουν σε αυτόν τον κλάδο (φοιτητές , μαθητές) . Θα ήταν πιο εύκολο και για αυτούς να υπάρχει ήδη μια έτοιμη πλατφόρμα, στην οποία το μόνο που θα πρέπει να κάνουν είναι να την κατεβάσουν και στην συνέχεια να πειραματιστούν πάνω σε αυτήν.

1.3 Δομή Διπλωματικής

Η διπλωματική εργασία αποτελείται από (8) κεφάλαια συνολικά. Το πρώτο κεφάλαιο περιλαμβάνει εισαγωγικές πληροφορίες ,για τις τεχνολογίες που χρησιμοποιήθηκαν και μια ιστορική αναδρομή για το καθένα ξεχωριστά . Στο δεύτερο κεφάλαιο εντείνεται παραπάνω η έννοια του VM , πως λειτουργεί στον υπολογιστή ένα VM και κάποιες βασικές λειτουργίες και χρήσεις που μπορεί να πραγματοποιήσει ένας χρήστης σε αυτό . Έπειτα στο τρίτο κεφάλαιο αναφέρεται εισαγωγικές πληροφορίες για το Flask την εξέλιξη του και τι εργαλείο είναι , καθώς και μια αναφορά , στο πως λειτουργεί ένα RESTful API , τι είναι και κάποιες βασικές έννοιες όπως το Status Code , Response/Structure Format και στο τέλος γίνεται μια αναφορά για τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της διπλωματικής εργασίας . Στην συνέχεια στο τέταρτο κεφάλαιο αναφέρονται τα έτοιμα ευπαθή Web Applications (Web Goat,Juice Shop) ,εισαγωγικές πληροφορίες για το καθένα αντίστοιχα , τι επιθέσεις μπορεί να πραγματοποιήσει ο χρήστης και τι νέες τεχνικές επίθεσης μπορεί να μάθει από αυτές , αναφορά ξανά στο Flask πλεονεκτήματα της χρήσης του , ένα παράδειγμα κατανοήσεις για το πώς γράφεται μια εφαρμογή Flask και στο τέλος του κεφαλαίου μια σύγκριση Flask μεταξύ Django και για πιο λόγο καταλήξαμε στην υλοποίηση της διπλωματικής με Flask. Στο πέμπτο κεφάλαιο αναφέρονται κάποιες βασικές επιθέσεις που μπορούν να γίνουν είτε στο Web Goat είτε στο Juice Shop και μερικές εικόνες για κάποιες επιθέσεις , που δείχνουν το περιβάλλον που θα βλέπει ο χρήστης και πως θα μπορεί να περιηγηθεί σε αυτό , γίνεται αναλυτική αναφορά σε επιθέσεις (SQL Injection , XSS Attacks , Broken Access Control , Broken Authentication , Cryptography failures) . Στο έκτο κεφάλαιο γίνεται μια επίδειξη , πως έγινε το Set up για τα (Web Goat , Juice Shop) , οι ετοιμασίες και εντολές που είναι απαραίτητες για την σωστή εκκίνηση του Flask App καθώς και μια εικόνα για την επίδειξη της δομής των Folders της εφαρμογή και κάποια πρακτικά παραδείγματα για τις λειτουργίες που μπορεί να κάνει το Flask . Στο έβδομο κεφάλαιο υπάρχουν ιδέες για μελλοντική εξέλιξη αυτής της διπλωματικής εργασίας και τέλος στο όγδοο κεφάλαιο είναι η βιβλιογραφία .

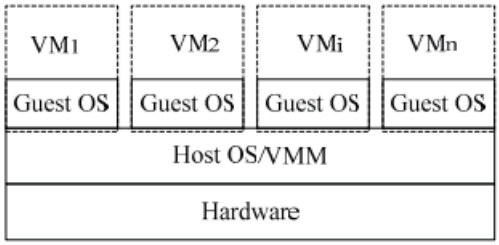
2. Εργαλείο VM

2.1 Τι είναι το VM

Με τη χρήση της τεχνολογίας VM, το υπολογιστικό σύστημα μπορεί να συνδυάσει διάφορους τύπους δεδομένων, λογισμικών και υλικών πόρων και να διαθέσει αυτούς τους πόρους για την εξυπηρέτηση διαφορετικών εργασιών. Επιπλέον, η τεχνολογία εικονικοποίησης διαχωρίζει τη διαχείριση του υλικού και του λογισμικού και προσφέρει σημαντικά χαρακτηριστικά, όπως απομόνωση απόδοσης, ενοποίηση διακομιστών και ζωντανή μετανάστευση συστημάτων. Παράλληλα, μπορεί να δημιουργήσει φορητά περιβάλλοντα για τα σύγχρονα υπολογιστικά συστήματα, διευκολύνοντας έτσι την ευελιξία και την αποδοτικότητα των πόρων.

Πιο συγκεκριμένα, η αρχιτεκτονική όπως απεικονίζεται στην εικόνα 1 , πολλαπλές εικονικές μηχανές (VMs) μοιράζονται την ίδια "φυσική μηχανή" ή κεντρικό σύστημα (host) . Στο χαμηλότερο επίπεδο, ακριβώς πάνω από το επίπεδο του υλικού, ο πυρήνας του λειτουργικού συστήματος (VMM) προσφέρει κατανομή πόρων στις εικονικές μηχανές . Σε κάθε εικονική μηχανή, διάφορες εργασίες ή υπηρεσίες εκτελούνται πάνω από το "guest" λειτουργικό σύστημα, το οποίο παρέχει το σύνηθες σύνολο υψηλού επιπέδου αφαιρέσεων, όπως πρόσβαση σε αρχεία και υποστήριξη δικτύου, για τις εφαρμογές που εκτελούνται στις εικονικές μηχανές. Τέλος, μια εικονική μηχανή (VM) είναι μια λογική μηχανή που έχει σχεδόν την ίδια αρχιτεκτονική με μια πραγματική κεντρική μηχανή, επιτρέποντας την εκτέλεση ενός λειτουργικού συστήματος μέσα σε αυτήν[3]

ΕΙΚΟΝΑ 1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ VM



3 Yunfa Li, Wanqing Li, Congfeng Jiang (2010). A Survey of Virtual Machine System: Current Technology and Future Trends : Grid and Service Computing Lab, School of Computer Science and Technology Hangzhou Dianzi University, 310018

2.2 Με ποιους τρόπους μπορεί να χρησιμοποιηθεί το VM

Οι εικονικές μηχανές (VMs) μπορούν να χρησιμοποιηθούν με διάφορους τρόπους, προσφέροντας ευελιξία και αποδοτικότητα σε μας εφαρμογές και περιβάλλοντα. Μερικοί από μας βασικούς τρόπους χρήσης περιλαμβάνουν:

i. **Δοκιμή και Ανάπτυξη Λογισμικού**: Οι προγραμματιστές μπορούν να χρησιμοποιούν VMs για να δοκιμάζουν εφαρμογές σε διαφορετικά λειτουργικά συστήματα και περιβάλλοντα χωρίς να απαιτείται ξεχωριστό υλικό.

ii. **Ενοποίηση Διακομιστών**: Ένα VM επιτρέπει την εκτέλεση πολλών διακομιστών σε έναν μόνο φυσικό κεντρικό υπολογιστή, μειώνοντας το κόστος υλικού και ενέργειας.

iii. **Ασφάλεια και Απομόνωση**: Τα VMs παρέχουν απομόνωση μεταξύ των εικονικών περιβαλλόντων, καθιστώντας μας κατάλληλες για δοκιμές ασφαλείας ή επικίνδυνων εφαρμογών χωρίς να διακινδυνεύετέ το κύριο σύστημα.

iv. **Δημιουργία Φορητών Περιβαλλόντων**: Οι εικονικές μηχανές επιτρέπουν τη δημιουργία πλήρως διαμορφωμένων και φορητών υπολογιστικών περιβαλλόντων που μπορούν να μεταφερθούν και να εκτελεστούν σε διαφορετικά συστήματα.

v. **Εκπαίδευση και Εκμάθηση:** Οι VMs είναι ιδανικές για εκπαιδευτικά περιβάλλοντα, επιτρέποντας μας μαθητές να πειραματιστούν με διαφορετικά λογισμικά ή λειτουργικά συστήματα χωρίς φόβο για καταστροφή του φυσικού συστήματος.

Εμείς στην συγκεκριμένη εργασία εστιάσαμε στα (ii , v) , καθώς θέλουμε ο εκπαιδευόμενος να είναι σε ένα ασφαλές περιβάλλον για να κάνει μας δοκιμές μας . Βέβαια θα πρέπει ο χρήστης να έχει ένα αρχείο που έχει κατάληξη (.ova) . Εδώ μας βοηθάει η υπηρεσία, RESTful API, που στήσαμε.

3 Εργαλείο Flask

3.1 Εισαγωγή

Το Flask είναι ένα από τα πιο δημοφιλή και ευρέως χρησιμοποιούμενα micro-frameworks για την ανάπτυξη web εφαρμογών με τη γλώσσα προγραμματισμού Python. Δημιουργήθηκε για να προσφέρει ευελιξία και απλότητα, ενώ παράλληλα επιτρέπει στους προγραμματιστές να έχουν τον πλήρη έλεγχο της αρχιτεκτονικής των εφαρμογών τους. Στη συνέχεια παρουσιάζεται μια ιστορική αναδρομή της εξέλιξης του Flask, καθώς και ο ρόλος του στη διαμόρφωση του οικοσυστήματος ανάπτυξης web εφαρμογών με Python. Η ιστορία του Flask ξεκινά με την ανάπτυξη δύο άλλων έργων που έθεσαν τα θεμέλια για το Flask: το **Werkzeug** και το **Jinja2**.

- **Werkzeug**: Είναι μια βιβλιοθήκη για τη διαχείριση αιτημάτων και αποκρίσεων HTTP. Σχεδιάστηκε αρχικά για να προσφέρει εργαλεία για web frameworks και εφαρμογές, καθιστώντας τη διαδικασία ανάπτυξης πιο εύκολη και ευέλικτη.

- **Jinja2**: Είναι ένας μηχανισμός απόδοσης προτύπων (templating engine) για τη Python, που επιτρέπει στους προγραμματιστές να δημιουργούν δυναμικά HTML templates και να ενσωματώνουν δεδομένα με ευέλικτο τρόπο

Από το 2015 και μετά, το Flask εδραιώθηκε ως ένα από τα κορυφαία frameworks στην Python για ανάπτυξη web εφαρμογών. Σε αυτή την περίοδο, έγιναν σημαντικές βελτιώσεις στη σταθερότητα και την απόδοση του Flask, με πολλές νέες εκδόσεις να βελτιώνουν τον τρόπο με τον οποίο το framework διαχειρίζεται τα αιτήματα και την επέκταση του λογισμικού. Σήμερα, το Flask παραμένει ένα από τα κορυφαία micro-frameworks για Python, ιδιαίτερα δημοφιλές σε προγραμματιστές που χρειάζονται ταχύτητα και ευελιξία στη δημιουργία web εφαρμογών και API.

3.2 Τι είναι μια RESTful υπηρεσία

Μια RESTful API (Application Programming Interface) είναι διεπαφή προγραμματισμού εφαρμογών, που επιτρέπει σε διαφορετικά κομμάτια λογισμικού να επικοινωνούν μεταξύ τους . Συγκεκριμένα, οι APIs υπηρεσίες παρέχουν έναν τρόπο για να εκτελείται συγκεκριμένος κώδικας ή να γίνεται χρήση υπηρεσιών όπως:

- Βιβλιοθήκες κώδικα (Libraries): Μια βιβλιοθήκη μπορεί να περιέχει σύνολα συναρτήσεων ή διαδικασιών που μπορεί κάποιος να χρησιμοποιήσει μέσω μιας API. Η API καθορίζει πώς θα γίνει η πρόσβαση σε αυτές τις λειτουργίες.

- SDKs (Software Development Kits): Τα SDKs περιέχουν εργαλεία και βιβλιοθήκες που βοηθούν τους προγραμματιστές να δημιουργούν εφαρμογές για μια συγκεκριμένη πλατφόρμα. Οι APIs σε αυτά τα SDKs προσδιορίζουν πώς να χρησιμοποιηθεί η πλατφόρμα ή οι βιβλιοθήκες για την ανάπτυξη εφαρμογών.

- Frameworks: Τα Frameworks παρέχουν ένα προκαθορισμένο περιβάλλον εργασίας για την ανάπτυξη εφαρμογών και περιλαμβάνουν APIs που προσφέρουν συγκεκριμένες λειτουργίες και δυνατότητες, όπως διαχείριση βάσεων δεδομένων ή διαχείριση της διεπαφής χρήστη.

Είναι ένας τρόπος για τις εταιρίες να προσφέρουν τα δεδομένα και τις υπηρεσίες τους στον χρήστη , χωρίς απαραίτητα ο χρήστης να γνωρίζει από προγραμματισμό[4]

4 Lauren Murphy , Tosin Alliyu , Andrew Macvean , Mary Beth Kery, Brad A. Myers (2017) , Preliminary Analysis of REST API Style Guidelines

3.3 Πώς είναι ένα Response/Structure Format , Standards Methods και Status Code στο RESTful API

Αφού ο χρήστης αρχίζει και εκμεταλλεύεται τις υπηρεσίες και τα δεδομένα του εκάστοτε API θα πρέπει να μεταφερθούν αυτά τα δεδομένα . Η οδηγίες που δόθηκαν για να ισχύει για όλες τις υπηρεσίες REST , θέτουν ότι τα δεδομένα θα μεταφέρονται και θα επεξεργάζονται μέσω ενός JSON αρχείου . Ο τρόπος εμφάνισης των ένθετων αντικειμένων, η μορφή των συνδεδεμένων στοιχείων, καθώς και οι κανόνες για τους τύπους αντικειμένων που πρέπει να περιλαμβάνονται σε μια απόκριση.

Αν και οι περισσότερες οδηγίες συμφωνούν σε θέματα όπως η χρήση του JSON για τη μορφοποίηση δεδομένων και η μορφή ημερομηνίας και ώρας, υπάρχουν διαφορές στους κανόνες για τους τύπους αντικειμένων που πρέπει να χρησιμοποιούνται σε μια απόκριση και στη μορφή των συνδεδεμένων στοιχείων, κάτι που μπορεί να προκαλέσει ασυνέπειες μεταξύ διαφορετικών API ή συστημάτων.

Εφόσον το Structure Format είναι σε JSON, πρέπει να οριστούν και άλλοι κανόνες για την λειτουργία και τις μεθόδους που μπορεί να τρέξει μια RESTful API υπηρεσία. Οι βασικές μέθοδοι είναι οι εξής:

- **GET** : Χρησιμοποιείται μόνο για να λάβει ο χρήστης τις πληροφορίες που θέλει από την REST υπηρεσία χρησιμοποιώντας το αντίστοιχο (URI).
- **POST** : Ο χρήστης στέλνει κάποια δεδομένα στον server, είτε για να δημιουργήσει, είτε για να ανανεώσει κάποια δεδομένα. Τα δεδομένα όπως προαναφέραμε το (body) τους είναι με αρχείο JSON.
- **PUT** : Αντικαθιστά τα ήδη υπάρχοντα δεδομένα, με νέα δεδομένα, τα οποία όρισε ο χρήστης . Η βασική του λειτουργία είναι να ανανεώνει (update) τα αντικείμενα του server.
- **DELETE** : Επιτρέπει την διαγραφή ενός συγκεκριμένου πόρου από τον server.

Τέλος, θέλουμε και μηνύματα ενημέρωσης από την μεριά του server, με σκοπό να ελέγχεται καλύτερα η μετάδοση δεδομένων και η κατάστασή τους (Status Code). Παρακάτω, είναι ο πίνακας για τα βασικά status code που μπορεί να συναντήσει κάποιος όταν φτιάχνει έναν RESTful API Server.

Πίνακας 2 Status Code [5]

Νούμερο	Status	Περιγραφή
200	OK	Η περίπτωση που ο χρήστης ζητά ένα GET request και λαμβάνει τα δεδομένα επιτυχώς
201	CREATED	Όταν ο χρήστης εισάγει δεδομένα με την μέθοδο POST και εκχωρούνται επιτυχώς
304	NOT MODIFIED	Ενημερώνει τον Client ότι η απάντηση που έλαβε δεν έχει αλλάξει με αποτέλεσμα να συνεχίζει να χρησιμοποιεί τα ίδια cashed version από την απάντηση
400	BAD REQUEST	Περίπτωση λάθους , όπου ο client έστειλε λάθος μέθοδο (π.χ post ενώ έπρεπε να είναι GET)
401	UNAUTHORIZED	Περίπτωση login όπου ο client προσπαθεί να πάρει πρόσβαση σε πόρους με λανθασμένα credentials
404	NOT FOUND	Περίπτωση που ο client , προσπαθεί να λάβει πόρους από ένα (URL) , όπου το API δεν το αναγνωρίζει.
500	INTERNAL SERVER ERROR	Ο server αντιμετωπίζει ένα πρόβλημα που δεν ξέρει πως να το αντιμετωπίσει.

5 HTTP response status codes [https://developer.mozilla.org/en-US/docs/Web/HTTP/Status]

3.4 Ποια εργαλεία χρησιμοποιήθηκαν

Για την ολοκλήρωση της παρούσας εργασίας χρειαστήκαμε τα εξής εργαλεία:

Python

Flask : Βιβλιοθήκη για την διαχείριση του RESTful API και τον ορισμό των URL που θα απευθύνεται το backend.

mysql.connector :Βιβλιοθήκη για την επικοινωνία της εφαρμογής μου με την βάση δεδομένων.

Threading : Βιβλιοθήκη για την διαχείριση και εκτέλεση διεργασιών για πολλαπλούς χρήστες

Blueprint : Στην **python**, την χρησιμοποιούμε για να ομαδοποιήσουμε και να δημιουργήσουμε πιο μικρές λειτουργικές ενότητες, με σκοπό την εύκολη ανάγνωση του κώδικα. Είναι ιδιαίτερα χρήσιμο σε περίπτωση που το project αρχίζει και μεγαλώνει και θες να ομαδοποιήσεις τις λειτουργίες του. Στην προκειμένη εργασία έχουμε δύο αρχεία που ξεχωρίζουν το (WEB.py , VBoxManage.py) .

Frontend-Backend: Για το μέρος τις ιστοσελίδας χρησιμοποιήθηκαν (html,css,javascript) για το frontend και για το backend για την επικοινωνία μεταξύ χρήστη και RESTful API (javascript).

Βάση δεδομένων (SQL)

Είναι στο (localhost database) και τρέχει **MySQL/MariaDB**

Η βάση δεδομένων περιέχει μία οντότητα (user), όπου με την σειρά της περιέχει (id_user,username,password). Ο κωδικός του user, κρυπτογραφείτε με SHA2. Αυτή, αποτελεί μια οικογένεια αλγορίθμων κατακερματισμού (hashing) και πιο συγκεκριμένα όσο αφορά την βάση δεδομένων, χρησιμοποιούμε τον SHA2-256 .

Virtual Box (VM)

- Εγκατάσταση (.iso) Desktop τελευταίας έκδοσης από το εξής link .

- Έπειτα στο περιβάλλον μέσα στο VM έγιναν οι απαραίτητες εγκαταστάσεις μέσω εντολών docker ακολουθώντας πιστά τα βήματα που αναγραφόταν στο OWASP. Τα βήματα για Web Goat : (link) τα βήματα για Juice Shop : (link).

4 Θεωρία για τα εργαλεία που χρησιμοποιήθηκαν

4.1 Εισαγωγή στο WebGoat

Το Web Goat είναι ένα, επίτηδες, ευπαθές application. Ο βασικός του σκοπός, είναι ο πειραματισμός στην εύρεση ευπαθειών που μπορεί να βρεθούν σε ένα java-based application . Μέσω αυτού του application, μπορεί να εξασκηθεί ο χρήστης στις ικανότητες του για τον τομέα του application layer και πάνω από όλα με νόμιμο τρόπο, δεν επιτίθεται σε μία πραγματική υπηρεσία.

4.1.1 Ποιες επιθέσεις μπορούν να πραγματοποιηθούν στο WebGoat

Όπως προαναφέραμε το Web Goat είναι ένα Web Application γραμμένο σε java-based applications , σε αυτό μπορούν να πραγματοποιηθούν οι εξής επιθέσεις.

- **SQL Injection** : Είναι μια επίθεση, στην οποία ένας κακόβουλος χρήστης είτε μπορεί να προσπαθεί να πάρει πρόσβαση στην βάση δεδομένων ενός Web Application ,είτε να διαγράψει τα δεδομένα, είτε ακόμη και να τα αλλοιώσει. Θα αναφερθούμε πιο αναλυτικά παρακάτω για αυτήν την επίθεση

- **XSS Επίθεση**: Αποτελεί μια επίθεση, στην οποία γίνεται εισαγωγή κακόβουλου κώδικα JavaScript σε μια εφαρμογή ιστού, με απώτερο σκοπό να εκτελεστεί στον περιηγητή του χρήστη και να αποκτηθεί πρόσβαση σε ευαίσθητα δεδομένα.

- **CSRF Επίθεση** : Είναι μια επίθεση, όπου ένας κακόβουλος χρήστης εκμεταλλεύεται την ταυτότητα ενός άλλου χρήστη, με σκοπό να εκτελέσει ανεπιθύμητες ενέργειες στην εφαρμογή.

- **IDOR** : Επίθεση, που βασίζεται στην πρόσβαση σε ευαίσθητα αντικείμενα ή δεδομένα μέσω μη εξασφαλισμένων αναφορών, όπως ID σε URLs .

- **Broken Authentication and Session Management** : Σενάρια, όπου η διαχείριση ταυτότητας και **συνεδρίας** δεν είναι σωστά εξασφαλισμένη,

επιτρέποντας σε κακόβουλους χρήστες να αποκτήσουν πρόσβαση σε λογαριασμούς άλλων.

- **Command Injection** : Επίθεση, όπου ο επιτιθέμενος εισάγει εντολές συστήματος σε μια εφαρμογή που δεν ελέγχει σωστά την είσοδο του χρήστη.

- **XXE Επίθεση** : Επίθεση σε εφαρμογές που επεξεργάζονται XML δεδομένα, επιτρέποντας στον επιτιθέμενο να εκμεταλλευτεί εξωτερικές οντότητες

- **Insecure Deserialization** : Επίθεση, η οποία αφορά την εκμετάλλευση δεδομένων που επανεγγράφονται από μη αξιόπιστες πηγές.

- **Security Misconfigurations** : Σφάλματα στις ρυθμίσεις ασφαλείας που μπορεί να αφήσουν εκτεθειμένη την εφαρμογή σε διάφορες επιθέσεις.

4.2 Εισαγωγή στο Juice Shop

Το Juice Shop επιλέχθηκε , γιατί προσομοιώνει αρκετά καλά ένα Web Application του τώρα . Είναι και αυτό ένα, επίτηδες, ευπαθές Web Application και μπορεί να χρησιμοποιηθεί για εκπαιδευτικούς λόγους , για CTFs και πολλά άλλα.

Οι ευπάθειες που περιέχει το Juice Shope βασίζεται σε όλο το **OWASP Top Ten** [6] , μαζί φυσικά και με άλλες ευπάθειές που μπορείς να συναντήσεις σε ένα πραγματικό σενάριο.

Το Application Juice Shop έχει γραφτεί σε (Node.js , Angular και Express).

4.2.1 Ποιες επιθέσεις μπορούν να πραγματοποιηθούν στο Juice Shop

Οι ευπάθειες που μπορεί να εντοπίσει κάποιος στο Juice Shop, είναι παρόμοιες με του Web Goat . Αυτό που αλλάζει, είναι πως παρέχει επιπλέον και αυτές τις παραπάνω επιθέσεις.

- **NoSQL Injection** : Επιθέσεις σε βάσεις δεδομένων NoSQL (όπως MongoDB) , με σκοπό την απόκτηση ή αλλοίωση δεδομένων.

- **Directory Traversal** : Εκμετάλλευση της δυνατότητας πλοήγησης στο σύστημα αρχείων του διακομιστή, με σκοπό την πρόσβαση σε αρχεία εκτός του προβλεπόμενου καταλόγου.
- **Broken Access Control** : Επιθέσεις, που στοχεύουν στο να αποκτήσει κανείς πρόσβαση σε περιοχές της εφαρμογής που δεν θα έπρεπε να είναι προσβάσιμες ,χωρίς την κατάλληλη εξουσιοδότηση.

- **Business Logic Vulnerabilities** : Εκμετάλλευση λογικών σφαλμάτων στη ροή της εφαρμογής για να αποκτήσει κάποιος πλεονέκτημα ή να παρακάμψει κανόνες.

- **CORS Misconfiguration** : Εκμετάλλευση λανθασμένων ρυθμίσεων CORS για να αποκτήσει ο εκάστοτε χρήστης ,πρόσβαση σε δεδομένα από διαφορετική προέλευση.

4.3 RESTful API μέσω python (Flask)

Επιλέχτηκε το Flask της python για το RESTful API , διότι αποτελεί ένα ελαφρύ και ευέλικτο web framework . Είναι ιδανικό για εφαρμογές που ξεκινούν από μικρές και μπορούν να επεκταθούν σε μεγαλύτερες, πιο πολύπλοκες εφαρμογές.

Η αρχιτεκτονική του Flask, ακολουθεί μια microframe αρχιτεκτονική. Πιο συγκεκριμένα, αυτό σημαίνει ότι προσφέρει τις βασικές λειτουργίες που χρειάζεται μια Web εφαρμογή, χωρίς όμως να επιβάλλει συγκεκριμένη δομή ή επιπλέον βιβλιοθήκες.

4.3.1 Θετικά για την χρήση Flask

- **Απλή και Εύχρηστη**: Το Flask, είναι γνωστό για την απλότητά του και την ευκολία χρήσης, που το καθιστούν ιδανικό για αρχάριους αλλά και έμπειρους προγραμματιστές.
- **Modular Design**: Επιτρέπει την κατασκευή εφαρμογών με αρθρωτή δομή, κάνοντας ευκολότερη την ανάπτυξη και συντήρηση πολύπλοκων εφαρμογών.
- **Επέκταση με Plugins**: Παρέχει τη δυνατότητα επέκτασης με τη χρήση plugins ή άλλων εξωτερικών βιβλιοθηκών για λειτουργίες όπως η διαχείριση βάσεων δεδομένων, ο έλεγχος ταυτότητας και η φόρτωση αρχείων.

6. OWASP Top Ten [https://owasp.org/www-project-top-ten/]

- **Routing**: Παρέχει ενσωματωμένο σύστημα δρομολόγησης (routing) για τον καθορισμό των διαδρομών (URLs) και των αντίστοιχων views (λειτουργιών που εκτελούνται για κάθε διαδρομή).

- **Ενσωματωμένος Development Server**: Το Flask, έρχεται με έναν ενσωματωμένο development server για την εύκολη εκτέλεση και δοκιμή εφαρμογών τοπικά, κατά την ανάπτυξη.

4.3.2 Χρήση του Flask και ένα απλό παράδειγμα

Ο κώδικας της python με την βιβλιοθήκη Flask σχεδιάστηκε, έτσι ώστε να διευκολύνει αρκετά τον προγραμματιστή στην κατανόηση του κώδικα. Στο παρακάτω παράδειγμα έχουμε στήσει έναν server όπου με την μέθοδο GET επιστρέφει στον χρήστη ένα hello world.

ΕΙΚΟΝΑ 2 ΠΑΡΑΔΕΙΓΜΑ FLASK

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run()
```

Ο χρήστης, μπορεί να έχει πρόσβαση σε αυτό το αποτέλεσμα πηγαίνοντας σε έναν φυλλομετρητή (π.χ Firefox) . Πατώντας το (http://loaclhost:5000/), επιστρέφει σε μορφή JSON αρχείο, το μήνυμα (“Hello, World!”).

Μέσω του Flask, ο προγραμματιστής έχει την δυνατότητα να επιστρέφει στον χρήστη αντί για ένα JSON αρχείο ,αρχεία (html,css,,javascript, κ.α). Αυτό μπορεί να πραγματοποιηθεί εύκολα μέσω της εντολής : (**render_template('Test.html')**).

ΕΙΚΟΝΑ 3 ΠΑΡΑΔΕΙΓΜΑ FLASK WEBPAGE

```
@blp.route("/")
def website():
    return render_template('Website.html')
```

Με βάση την παραπάνω εικόνα, ο χρήστης μπορεί να έχει πρόσβαση στον πόρο (Website.html) γράφοντας την εξής διεύθυνση : (http://localhost:5000/).

Όπως ανέφερα και στα παραδείγματα παραπάνω, υπάρχουν πολλά εργαλεία από την βιβλιοθήκη (flask) για να δημιουργήσει ένας προγραμματιστής ένα application με βάση αυτό . Όμως, υπάρχει και το Django που παρέχει ακόμα περισσότερες λειτουργίες.

4.3.3 Flask vs Django

Πριν την σύγκριση, θα πρέπει να αναφερθώ στα βασικά πλεονεκτήματα που έχει ένα Web Application γραμμένο σε Django. Αρχικά, είναι αρκετά ευέλικτό με αποτέλεσμα, κάποιες διεργασίες να μην κινδυνεύουν να φτάσουν σε κρίσιμο σημείο . Με τον όρο , κρίσιμο σημείο, εννοούμε να μην ανταποκρίνεται ο server και να επιστρέφει μήνυμα (500) . Οι εκδόσεις που έχει το Django, υποστηρίζουν αρκετά (URL formats). Το Django, δημιουργεί τα δικά του templates, ενώ το flask θα πρέπει να αναφερθεί σε τρίτο και πιο συγκεκριμένα (Jinja 2 Template) . Ακόμη, υπάρχουν συνεχόμενες ενημερώσεις για νέα έκδοση ,τουλάχιστον 2 φορές τον χρόνο . Τέλος, το Authorization API γίνεται από το ίδιο το Django και όχι από τρίτους όπως κάνει το Flask. Συνεπώς, καταλήγουμε στο ότι δεν υπάρχει πραγματικά καλύτερη επιλογή, αλλά ότι αφήνεται στην κρίση του προγραμματιστή και φυσικά εξαρτάται άμεσα και από το είδος του προγράμματος που θέλει και πρέπει να «τρέξει». Σε γενικό πλαίσιο, το Flask προτιμάται για μικρές εφαρμογές. Ενώ, σε μεγάλα Project χρησιμοποιείται κυρίως το Django . Για την συγκεκριμένη εργασία το Web Application που στήσαμε, είναι σχετικά ελαφρύ.

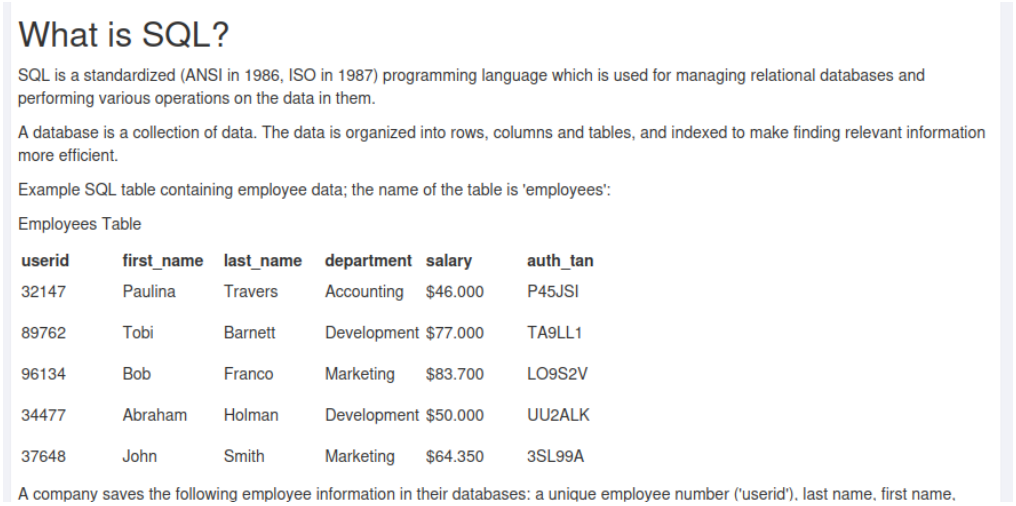
7. Nuruldelmia Idris a, Cik Feresa Mohd Foozy a,b,1,*, Palaniappan Shamala a,b (April 2020),A Generic Review of Web Technology: DJango and Flask

5 Βασικές Επιθέσεις

5.1 SQL Injections

Το SQL Injection είναι μια τεχνική επίθεσης σε βάσεις δεδομένων, όπου ένας κακόβουλος χρήστης "εισάγει" κακόβουλο κώδικα SQL σε μια εφαρμογή, με στόχο να εκμεταλλευτεί τα κενά ασφαλείας στην επεξεργασία των δεδομένων από την εφαρμογή. Μπορεί να λάβει τα δεδομένα μιας βάσεις να τα παραμετροποιήσεις , ακόμα και να διαγράψει ολόκληρα tables .

Για το WebGoat στην υποκατηγορία (SQL Injection (Intro)) , βάζει τον χρήστη να καταλάβει πως να χειρίζεται βασικές εντολές SQL. Για παράδειγμα παρακάτω δίνεται ένας πίνακας βάσης δεδομένων (employee) και ζητάει από τον χρήστη να επιστρέψει το “department” του “Bob Franco”.



What is SQL?

SQL is a standardized (ANSI in 1986, ISO in 1987) programming language which is used for managing relational databases and performing various operations on the data in them.

A database is a collection of data. The data is organized into rows, columns and tables, and indexed to make finding relevant information more efficient.

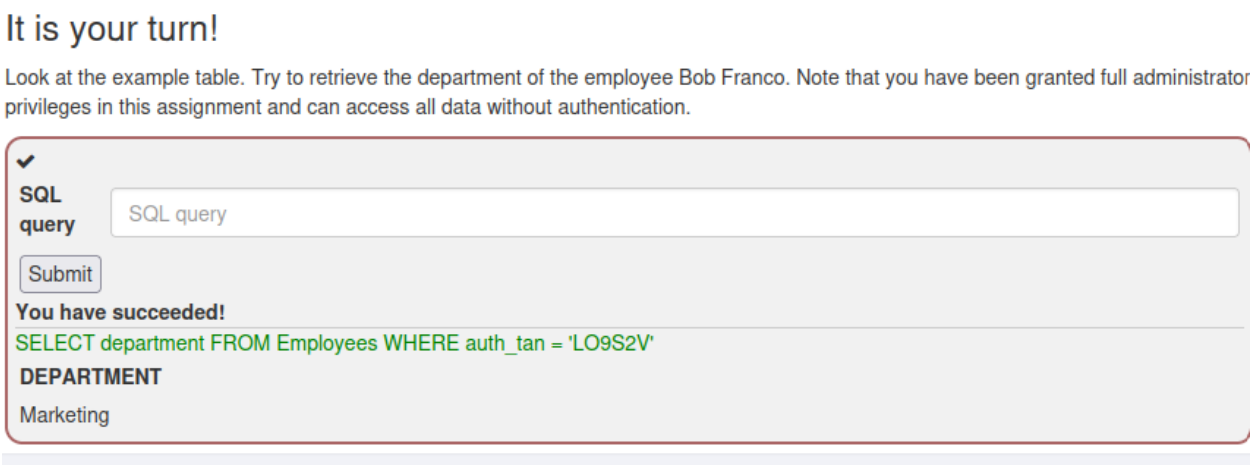
Example SQL table containing employee data; the name of the table is 'employees':

Employees Table

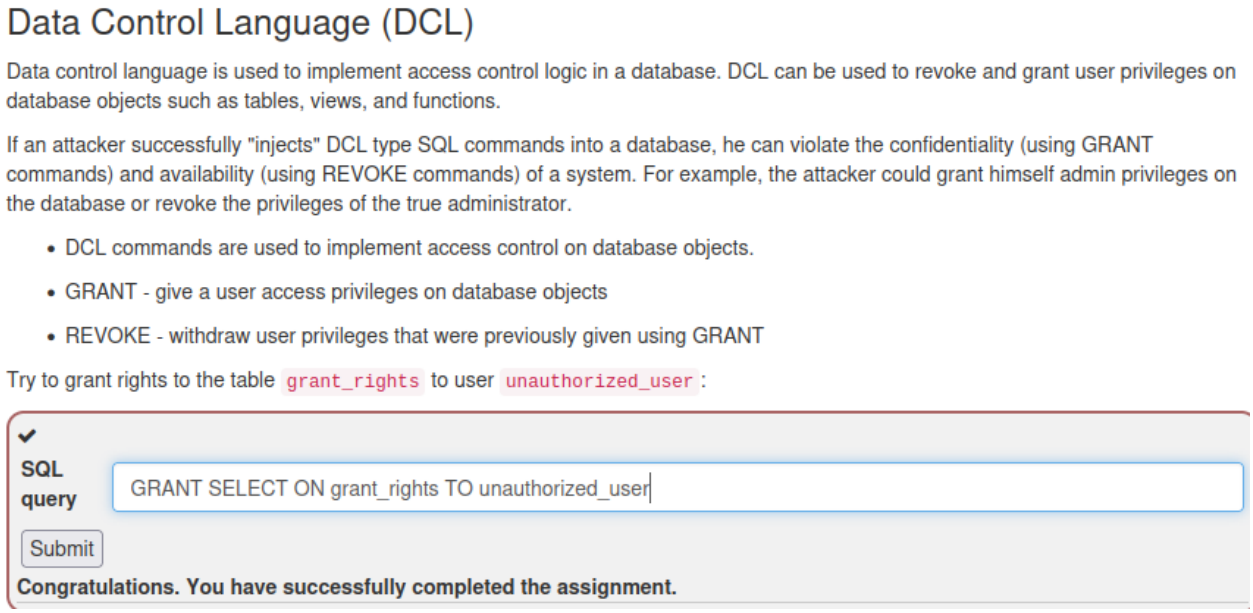
userid	first_name	last_name	department	salary	auth_tan
32147	Paulina	Travers	Accounting	$46.000	P45JSI
89762	Tobi	Barnett	Development	$77.000	TA9LL1
96134	Bob	Franco	Marketing	$83.700	LO9S2V
34477	Abraham	Holman	Development	$50.000	UU2ALK
37648	John	Smith	Marketing	$64.350	3SL99A

A company saves the following employee information in their databases: a unique employee number ('userid'), last name, first name,

Λύση σε αυτό το πρόβλημα είναι η εντολή είναι η παρακάτω εικόνα.



Ερώτημα που αλλάζεις τα δικαιώματα για έναν χρήστη (unauthorized_user).



Μέσα υπάρχει και ο κλασικός τρόπος επίθεσης (1==1) βλέπουμε παρακάτω τον κώδικα προσπαθούμε να πάρουμε όλα τα στοιχεία των users από την βάση δεδομένων.

SELECT * FROM users WHERE username = '' OR '1'='1';

5.2 Broken Access Control

Το Broken Access Control είναι μια κατηγορία ευπαθειών ασφαλείας που συμβαίνει όταν ένα σύστημα αποτυγχάνει να περιορίσει ή να ελέγξει σωστά την πρόσβαση χρηστών ή συσκευών σε πόρους ή λειτουργίες, οδηγώντας σε μη εξουσιοδοτημένη πρόσβαση.

Ποιους τύπους επιθέσεων συναντάμε στον τομέα του Broken Access Control .

1. **Bypassing Authentication** : Όταν ένας χρήστης μπορεί να παρακάμψει τη διαδικασία σύνδεσης και να αποκτήσει πρόσβαση σε προστατευμένες περιοχές.

2. **Privilege Escalation** : Όταν ένας χρήστης με χαμηλά προνόμια (π.χ. κανονικός χρήστης) καταφέρνει να εκτελέσει ενέργειες που θα έπρεπε να είναι διαθέσιμες μόνο σε έναν χρήστη με δικαιώματα root.

3. **Insecure Direct Object References (IDOR):** Όταν ένας χρήστης μπορεί να έχει πρόσβαση σε δεδομένα που δεν του ανήκουν απλά αλλάζοντας το ID ή άλλο στοιχείο αναφοράς σε ένα URL ή φόρμα

Έστω μια επίθεση IDOR , οι διαχειριστές μπορούν να δημιουργούν και να διαγράφουν χρήστες . Αν ένας χρήστης μέσω του URL μπορεί να καλέσει ενέργεια διαγραφείς (π.χ. /admin/delete?id=1) , χωρίς να υπάρχουν έλεγχοι για την ταυτότητά του και τα δικαιώματά του, τότε υπάρχει **Broken Access Control**.

5.3 Cryptography failures

Τα Cryptography failures , αναφέρονται σε καταστάσεις όπου η κρυπτογραφία δεν προστατεύει επαρκώς τα δεδομένα, λόγω κακής σχεδίασης, υλοποίησης ή χρήσης των κρυπτογραφικών τεχνικών.

Βασικά αίτια που μπορούν να οδηγήσουν σε μια τέτοια ευπάθεια είναι .

Χρήση κρυπτογράφησης ECB (Electronic Codebook) mode: Το ECB mode για συμμετρική κρυπτογράφηση μπλοκ είναι επικίνδυνο γιατί κρυπτογραφεί ίδια μπλοκ δεδομένων με τον ίδιο τρόπο, καθιστώντας τα ευπαθή σε pattern αναλύσεις.

Αποθήκευση κλειδιών σε μη ασφαλή μέρη: Αν τα κρυπτογραφικά κλειδιά αποθηκεύονται σε plain text ή σε μέρη που δεν προστατεύονται επαρκώς (π.χ., κώδικας πηγαίου κώδικα ή μη ασφαλείς βάσεις δεδομένων), οι επιτιθέμενοι μπορούν εύκολα να τα κλέψουν.

Μεταφορά ευαίσθητων δεδομένων χωρίς κρυπτογράφηση: Πολλά συστήματα αποτυγχάνουν να κρυπτογραφήσουν την επικοινωνία μεταξύ των πελατών και των servers, επιτρέποντας σε έναν επιτιθέμενο να καταγράψει ή να αλλάξει τα δεδομένα κατά τη μεταφορά (π.χ., σε HTTP αντί για HTTPS).

5.4 Cross-Site Scripting (XSS)

Το **Cross-Site Scripting** είναι μια επίθεση που επιτρέπει σε έναν επιτιθέμενο να εισάγει κακόβουλο κώδικα JavaScript (ή άλλες γλώσσες scripting) σε μια εφαρμογή ιστού, με στόχο την εκτέλεση αυτού του κώδικα στους browsers των ανυποψίαστων χρηστών. Ο κακόβουλος κώδικας μπορεί να οδηγήσει σε κλοπή cookies,συλλογή προσωπικών δεδομένων.

Οι βασικοί τρόποι επίθεσης XSS attacks είναι οι ακόλουθοι.

- **Stored XSS (Αποθηκευμένο XSS):** Ο κακόβουλος κώδικας αποθηκεύεται μόνιμα στον server (π.χ. σε μια βάση δεδομένων) και εκτελείται όταν οι χρήστες επισκέπτονται την επηρεαζόμενη σελίδα.

- **Reflected XSS (Αντανακλώμενο XSS):** Ο κακόβουλος κώδικας περιλαμβάνεται σε μια παράμετρο URL και εκτελείται όταν το θύμα επισκεφτεί αυτό το URL.

- **DOM-based XSS**: Η ευπάθεια βρίσκεται στον client-side κώδικα JavaScript, όταν η εφαρμογή χειρίζεται δεδομένα χωρίς κατάλληλη επικύρωση ή φιλτράρισμα.

5.5 Broken Authentication

Αυτή η επίθεση αναφέρεται σε αδυναμίες στις διαδικασίες αυθεντικοποίησης και διαχείρισης συνεδριών μιας εφαρμογής, οι οποίες μπορεί να επιτρέψουν σε επιτιθέμενους να παρακάμψουν τις διαδικασίες ασφαλείας και να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε λογαριασμούς χρηστών ή συστήματα.

Επιθέσεις που μπορεί να γίνουν για να εκμεταλλευτούν την ευπάθεια Broken Authentication.

Επιθέσεις Brute Force

- Αδυναμία να ανιχνευτούν πολλές αποτυχημένες προσπάθειες σύνδεσης, που επιτρέπει στους επιτιθέμενους να δοκιμάσουν κωδικούς πρόσβασης μέχρι να βρουν τον σωστό.

Προβλήματα στη Διαχείριση Συνεδριών (Sessions)

- Χρήση μη ασφαλών cookies ή μη ρυθμισμένων flags για την προστασία τους (π.χ., HttpOnly, Secure).

- Ακατάλληλη ή ελλιπής αποσύνδεση (logout) που μπορεί να επιτρέψει σε επιτιθέμενους να ανακτήσουν συνεδρίες.

Λάθη στην Υλοποίηση της Αυθεντικοποίησης

- Σφάλματα στον κώδικα που επιτρέπουν σε επιτιθέμενους να παρακάμψουν τη διαδικασία αυθεντικοποίησης.

6 Υλοποίηση της εργασίας

6.1 VM (WebGoat , JuiceShop) Setup

Σαν πρώτο βήμα χρειάστηκε να κατεβάσουμε το (.iso file) μέσω αυτού του (link) , έπειτα όρισα τα χαρακτηριστικά κάθε συστήματος όπως βλέπετε στον παρακάτω πίνακα.

ΠΙΝΑΚΑΣ 3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ VM(WEBGOAT)

Λειτουργικό Σύστημα	Ubuntu (64-bit)
Βασική Μνήμη	4048MB
Επεξεργαστές	3
Μνήμη Γραφικών	16 MB
Εικονικό Μέγεθος	75 GB
Πραγματικό Μέγεθος	7,90 GB

Παρόμοια χαρακτηριστικά υπάρχουν και στο (Juice Shop) . Αφού ξεκινήσει να «τρέχει» το VM, μπήκα στο περιβάλλον ubuntu . Προσοχή! Στην περίπτωση που δεν γίνει το installation που αναφέρει το σύστημα να κάνετε στην αρχή , μπορείτε να πειραματιστείτε με το λειτουργικό σύστημα και να τρέξετε κάποιες εντολές . Όμως, μετά το κλείσιμο του VM, δεν θα έχει αποθηκευτεί καμία αλλαγή , διότι το VM βρίσκεται σε κατάσταση live mode . Δηλαδή, δεν πραγματοποιεί καμία εγγραφή στον δίσκο. Οπότε, πραγματοποιήθηκε η εγκατάσταση μαζί με δημιουργία λογαριασμού (username , password). Τα στοιχεία (username , password) ,θα μπορεί να τα βρει ο χρήστης όταν θα περιμένει την εγκατάσταση του .ova αρχείου. Εκεί, θα εμφανιστεί ένα pop up window που θα έχει μέσα αυτά τα στοιχεία. Αφού ολοκληρωθεί η εγκατάσταση του Ubuntu, πηγαίνουμε στο terminal και σαν πρώτο βήμα εγκαθιστούμε το Docker με την παρακάτω εντολή.

- **Sudo apt install docker.io**

Αφού ολοκληρωθεί η εγκατάσταση , ακολουθώντας τα βήματα που μας δίνονται από το OWASP παίρνουμε το image του (WebGoat) μέσω docker εντολής .

- **Sudo docker pull webgoat/webgoat**

Αυτή η εντολή θα κατεβάσει το image του και θα είναι έτοιμο τα τρέξει το Application με την εντολή.

- **Sudo docker run -d – p 127.0.0.1:8080:8080 –name webgoat webgoat/webgoat**

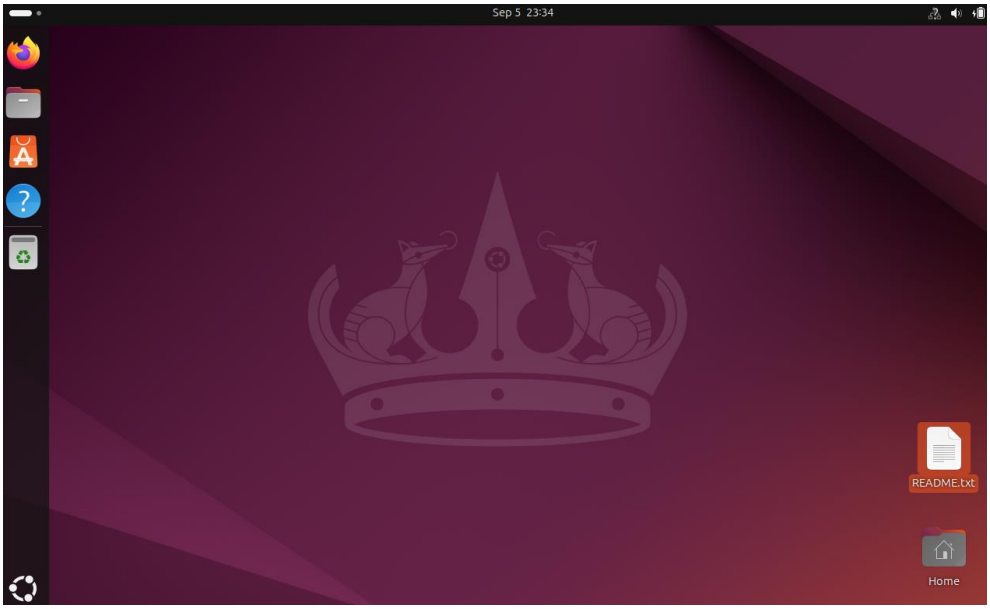
Η εντολή – name λειτουργεί σαν “shortcut” , δηλαδή αντί ο χρήστης να τρέχει αυτήν την εντολή με αυτόν τον τρόπο για να αρχίσει και να σταματήσει το Application θα χρειαστεί μόνο αυτές τις δυο εντολές

- **Sudo docker start webgoat**
- **Sudo docker stop webgoat**

Ο τρόπος εκκίνησες του WebGoat, όπως και για το Juice Shop , αναφέρεται σε ένα αρχείο (READ_ME.txt) , το οποίο αναφέρει αναλυτικά τα βήματα για να τρέξει κάποιος, που δεν έχει εμπειρία, αυτό το πρόγραμμα με ασφάλεια . Στην συνέχεια, αφού το Web Application «τρέχει» κανονικά, ο χρήστης μπορεί να έχει πρόσβαση σε αυτό πηγαίνοντας στο Firefox και πατώντας το URL: (**http://localhost:8080/WebGoat)** , ή μπορεί να πατήσει το εικονίδιο που του έχω βάλει στους σελιδοδείκτες. Σημαντική σημείωση, αποτελεί πως ο χρήστης ,δεν πρέπει να είναι συνδεδεμένος στο ίντερνετ και θα πρέπει να περιμένει λίγο να φορτώσει ,αφού «τρέξει» την εντολή εκκίνησης. Παρακάτω υπάρχουν εικόνες για το README που υπάρχει στο σύστημα και τι βλέπει ο χρήστης όταν συνδέεται επιτυχώς στο Application.

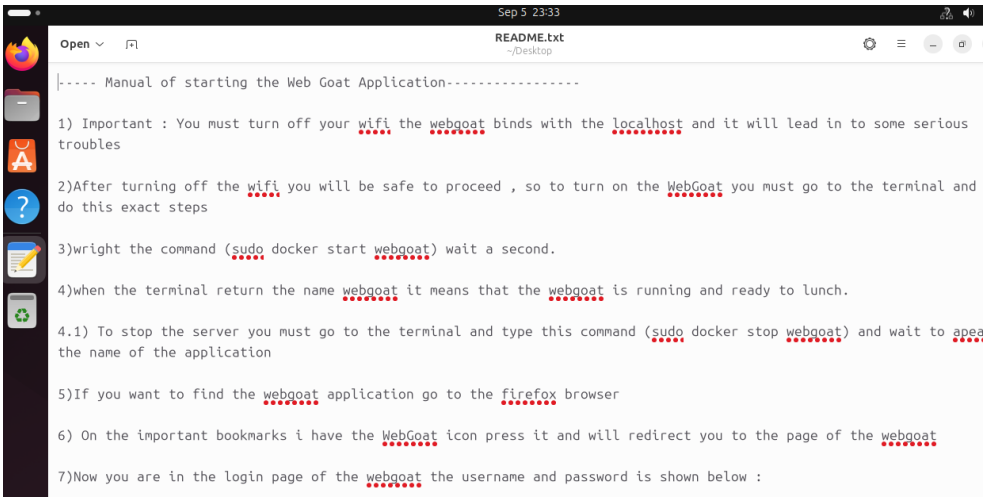
Καθώς μπαίνει στο περιβάλλον ο χρήστης, θα διακρίνει το (README.txt) αρχείο κάτω δεξιά όπως είναι στην εικόνα.

ΕΙΚΟΝΑ 4 WEB GOAT ΠΕΡΙΒΑΛΛΟΝ (VM)



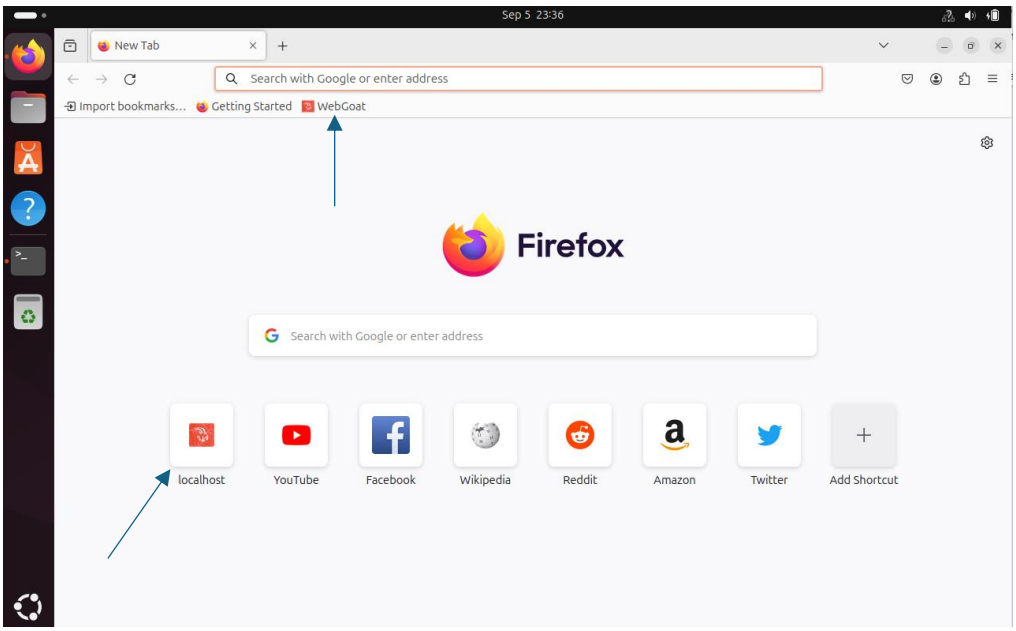
Και αφού το ανοίξει θα ακολουθήσει τις αναλυτικές οδηγίες για την σωστή λειτουργία της μηχανής.

ΕΙΚΟΝΑ 5 ΠΕΡΙΕΧΟΜΕΝΟ README



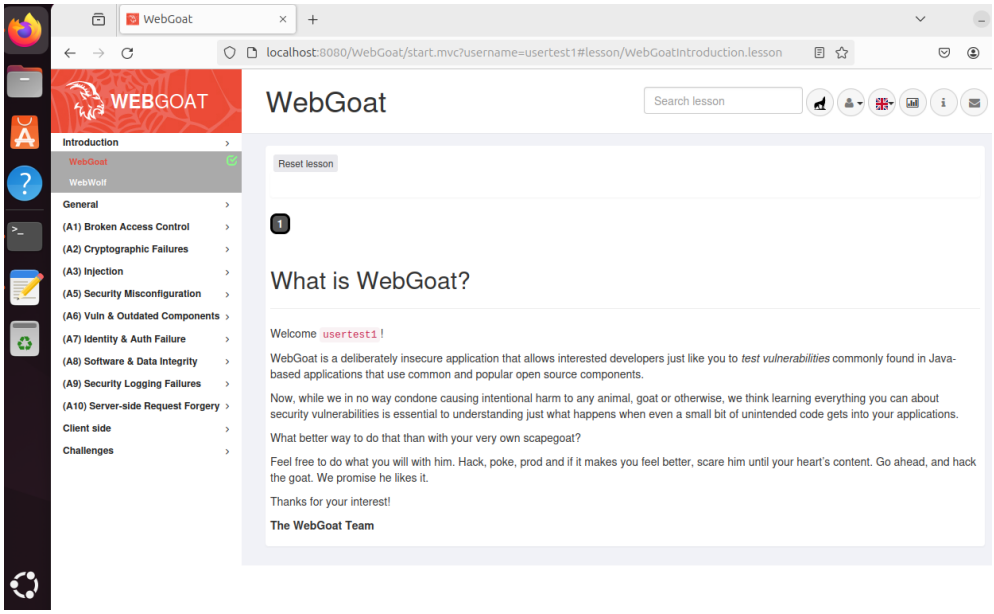
Εφόσον ο χρήστης, ακολούθησε σωστά τα βήματα θα πρέπει να μπει στο Firefox και να επιλέξει το εικονίδιο του Web Goat που είναι καρφιτσωμένο στους σελιδοδείκτες.

Εικονα 6 Εικονα Firefox – Webgoat



Και αφού πατήσει τα συγκεκριμένα εικονίδια τον μεταφέρει στην πλατφόρμα (login) . Ακολούθως, προχωρά στην εκχώρηση των στοιχείων που υπάρχουν στο README αρχείο . Παρακάτω, είναι η βασική σελίδα του Web Goat και από αριστερά τα μαθήματα που μπορεί να μάθει ο συγκεκριμένος χρήστης .

Εικονα 7 Βασικη Πλατφορμα Web Goat



6.2 Python Flask Web Application (Setup / Code)

Πριν ξεκινήσω την δημιουργία κώδικα (python) για την εφαρμογής μου, δημιούργησα ένα (venv) περιβάλλον. Η δημιουργία ενός (venv) περιβάλλοντος για το project είναι αρκετά σημαντική για τους παρακάτω λόγους:

- **Απομόνωση** : Στο συγκεκριμένο project, αλλά πολύ πιθανόν και σε άλλα project που απαιτούν flask κώδικα, χρειάζονται βιβλιοθήκες και πακέτα τρίτων . Η δημιουργία ενός (vevn) περιβάλλοντος, βοηθά τον χρήστη να ξεχωρίσει και να απομονώσει αυτές τις βιβλιοθήκες , με αποτέλεσμα να μην έρχονται σε σύγκρουση με άλλα project που έχουν δημιουργηθεί σε flask.

- **Ασφάλεια** : Όταν ο προγραμματιστής εργάζεται σε ένα (venv) περιβάλλον, απομονώνει τις βιβλιοθήκες που χρησιμοποιεί . Αυτό έχει ως αποτέλεσμα την μείωση του ρίσκου για μία μη ασφαλή βιβλιοθήκη και τον περιορισμό επιρροής άλλων μερών του συστήματος . Τέλος, η απομόνωση βοηθά στην αποφυγή τυχαίας εγκατάστασης κακόβουλου λογισμικού σε παγκόσμιο επίπεδο.

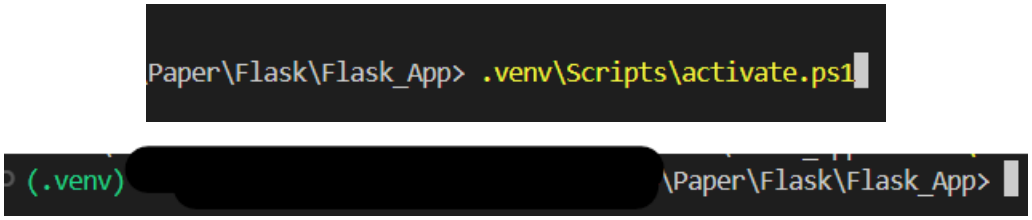
Με βάση αυτούς τους δυο βασικούς λόγος δημιούργησα ένα folder για να εκχωρήσω το project μου . Ακολούθησα τα παρακάτω βήματα.
Πρώτα, δημιούργησα ένα αρχείο (.venv) , καθώς εκεί θα πραγματοποιηθεί η εγκατάσταση του (Virtual Environment) και θα υπάρχουν όλα τα πακέτα που θα χρειαστώ για το project . Θα μπορούσα, να δημιουργήσω ένα αρχείο χωρίς την τελεία (.) στην αρχή του ονόματος. Όμως έτσι, θα έχανα μια βασική διαφορά και κατά την προσωπική μου άποψη απαραίτητη . Με την δημιουργία λοιπόν, φακέλου που αρχίζει με τελεία (.), ουσιαστικά καταφέρνω να γίνεται απόκρυψη από Version Control. Δηλαδή ,όταν ξεκινάει το αρχείο σου με αυτόν τον τρόπο (.venv) είναι κρυφό σε πολλά συστήματα αρχείων, γεγονός που βοηθά να μην το βλέπει συχνά όταν περιηγείσαι στους φακέλους.

Η δημιουργία ενός venv αρχείου, πραγματοποιείται με τον παρακάτω τρόπο στο (Virtual Code).

- **python 3.10 -m venv .venv**

Αφού εκτελεστεί αυτή η εντολή ,εφόσον χρησιμοποιούμαι (Virtual Code), πατάμε (Ctrl + Shift + p για **Windows ,** Command + Shift + p για **MAC**). Αναζητάμε python Interpreter και επιλέγουμε το “path” που εργαζόμαστε. Τέλος, με το shortcut (Ctrl + J για **Windows** και Command + J για **MAC**) ,μπαίνουμε κατευθείαν σε venv περιβάλλον .Σε περίπτωση που δεν μπορέσει ο χρήστης να ενεργοποιήσει το venv, μπορεί να πληκτρολογήσει στο terminal την εντολή που φαίνεται στην παρακάτω εικόνα.

ΕΙΚΌΝΑ 8 ΕΝΕΡΓΟΠΟΙΗΣΗ VENV



Το πράσινο (.venv), υποδηλώνει πως ο χρήστης είναι μέσα στο Virtual Environment και από εδώ και έπειτα κατεβάζουμε όλες τις απαραίτητες βιβλιοθήκες για την λειτουργία αυτού του προγράμματος.

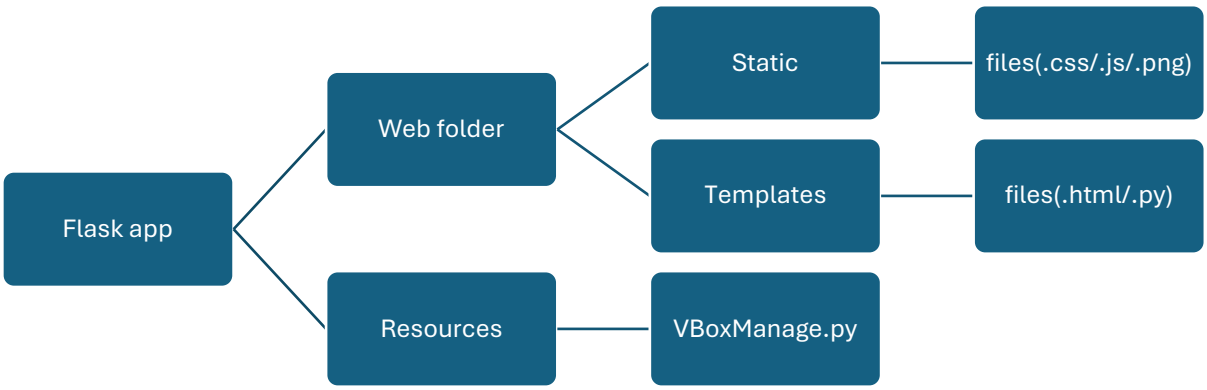
Εντολές και βιβλιοθήκες που χρησιμοποιήθηκαν

- **pip install flask**
- **pip install mysql-connector-python**
- **pip install flask-smorest (BluePrint)**

Αφού τελειώσαμε τις εγκαταστάσεις, προχωράμε στην υλοποίηση του κώδικα.

Παρακάτω είναι μια εικόνα πως χωρίζονται τα αρχεία στο project μας και τι λειτουργία που έχει το καθένα.

ΕΙΚΌΝΑ 9 ΑΡΧΙΤΕΚΤΟΝΙΚΗ (FOLDER - FLASK)



Με βάση την Εικόνα 9 , δημιουργήθηκαν δύο βασικά folders, με κοινό σκοπό την δημιουργία του Flask app . Πιο συγκεκριμένα το Web folder , περιέχει άλλα δύο folders (static , Template) , τα οποία και είναι απαραίτητα για την σωστή λειτουργία του flask . Το folder Template περιέχει όλα τα αρχεία (.html) και τον κώδικα python που δημιούργησα (WebSite.py). Το Flask ,χρησιμοποιεί το Jinja2 template engine για να κάνει rendering δυναμικό περιεχόμενο σε αυτά τα αρχεία..

Το Folder Static, περιέχει όλα τα στατικά αρχεία. Αυτά τα αρχεία, δεν αλλάζουν δυναμικά και είναι απαραίτητα για το στυλ και τη λειτουργικότητα της σελίδας (πχ CSS , Javascript , png) . Για να συμπεριλαμβάνεις αυτά τα αρχεία στην html μέσω του flaks ,χρησιμοποιείς την παρακάτω εντολή.

- **{{ url_for('static', filename='style.css') }}**

Τέλος, ο φάκελος Resources , περιέχει όλες τις λειτουργίες που μπορεί να κάνει ο χρήστης μέσω (GET,POST) request από το αρχείο (VBoxManage.py) . Βάση των ενεργειών του χρήστη ,“επικοινωνεί” κατευθείαν με το συγκεκριμένο αρχείο.

Οπότε και με την βοήθεια του Blueprint, που βοηθάει στην ομαδοποιήσει των ενεργειών , το όνομα της κάθε κατηγορίας καταγράφεται με τον εξής τρόπο:

- Για το Web folder (**blp = Blueprint("Website",__name__,description = "Operations on Website")**)

- Για το Resources folder (**blp = Blueprint("Users",__name__,description = "Operations on username / password")**)

Εφόσον τα έχουμε ομαδοποιημένα, μπορούμε στο main πρόγραμμα (app.py) να τα συμπεριλάβουμε όπως παρατηρούμε με τον παρακάτω κώδικα.

ΕΙΚΌΝΑ 10 MAIN(APP.PY)

```python
from flask import Flask
from resources.VBoxManage import blp as VboxManage
from templates.WebSite import blp as WEB

app = Flask(__name__)

app.register_blueprint(WEB)
app.register_blueprint(VboxManage)

if __name__ == '__main__':
    app.run()
```

Με την εντολή (**register_blueprint(όνομα του blueprint)**), μπορεί ο εκάστοτε χρήστης να εκχωρεί τις λειτουργίες του συγκεκριμένου blueprint στην εφαρμογή.

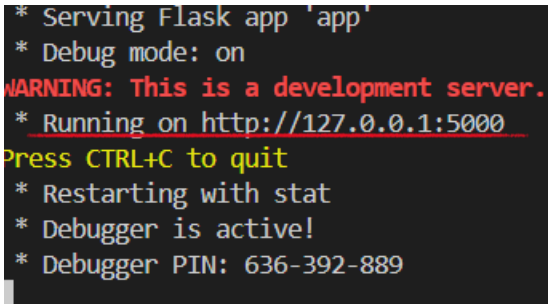
6.2.1 Λειτουργία App (Login / Start VM)

Εφόσον έχουμε ενεργοποιήσει το (.venv) , θα «τρέξουμε» το πρόγραμμα με την εντολή

- Flask run

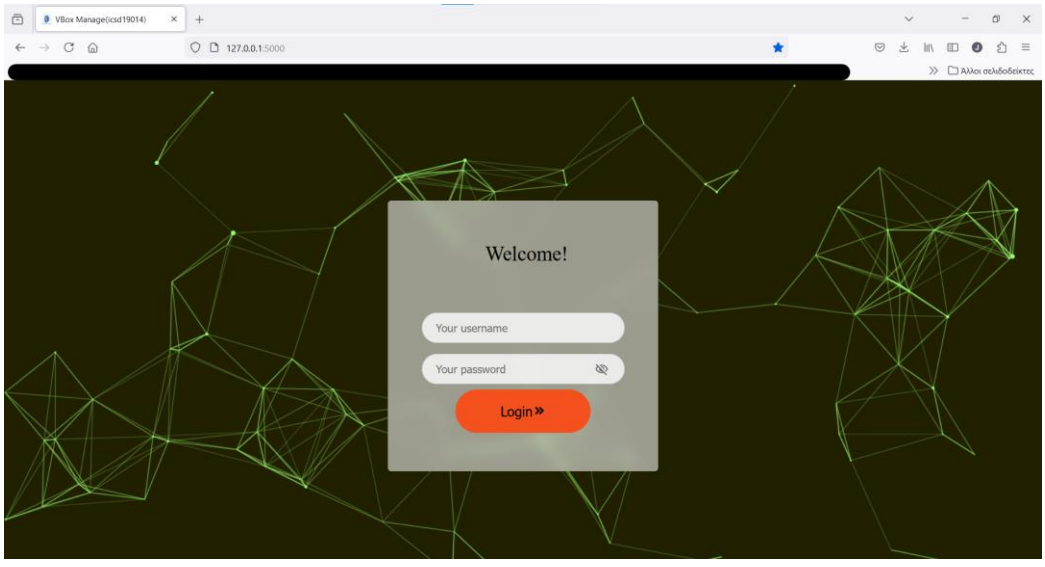
Αφού «τρέξει», θα δούμε στο terminal μας την πόρτα και την διεύθυνση που τρέχει η REST υπηρεσία μας .

ΕΙΚΟΝΑ 11 ΣΤΟΙΧΕΙΑ APP.PY



Τα πρώτα μηνύματα που βλέπουμε σαν απάντηση από την μεριά του REST είναι (200,304) . Αυτά, υποδηλώνουν πως όλα έχουν φορτώσει σωστά. Όσο αφορά τώρα τα αρχεία που είναι με μήνυμα 304 , ομοίως έχουν φορτώσει σωστά και μας δείχνει ουσιαστικά και μια επιπλέον πληροφορία ,ότι δεν έγινε καμία αλλαγή στον κώδικα σε αυτά τα αρχεία. Παρακάτω, είναι η φόρμα που βλέπει ο χρήστης και τα μηνύματα που παίρνει το REST.

ΕΙΚΟΝΑ 12 ΦΟΡΜΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ REST ΜΗΝΥΜΑΤΑ



```
127.0.0.1 - - [11/Sep/2024 12:19:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:19:50] "GET /static/Style.css HTTP/1.1" 304 -
127.0.0.1 - - [11/Sep/2024 12:19:50] "GET /static/scripts.js HTTP/1.1" 304 -
127.0.0.1 - - [11/Sep/2024 12:19:50] "GET /static/visibility_off.png HTTP/1.1" 304 -
127.0.0.1 - - [11/Sep/2024 12:19:50] "GET /static/cool-background.png HTTP/1.1" 304 -
127.0.0.1 - - [11/Sep/2024 12:19:54] "GET /static/visibility.png HTTP/1.1" 200 -
```

Login

Για την λειτουργία Login αρχικά, το πρόγραμμα μας πρέπει να επικοινωνήσει με την βάση δεδομένων που έχουμε στήσει. Η python χρειάζεται τα παρακάτω στοιχεία για να λάβει τα δεδομένα του χρήστη.

ΕΙΚΟΝΑ 13 ΣΥΝΔΕΣΗ ΜΕ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

```
def get_db_connection():
    try:
        connection = mysql.connector.connect(
            host='localhost',
            database='diplomatikh',
            user='root',
            passwo[redacted]
        )
        if [redacted].is_connected():
            return [redacted]
    except Error as e:
        print(f"Error while connecting to MySQL: {e}")
        return None
```

Στα πεδία (host = [την Ip της βάσης σου] , database = [το όνομα της βάσης σου] , user = [το username σου] , password = [και ο κωδικός σου για να μπεις στην βάση σου]).

Εφόσον έχει συνδεθεί σωστά στην βάση το πρόγραμμα , μπορούμε να συγκρίνουμε τα στοιχεία που δίνει ο χρήστης με την βάση. Ο κώδικας login είναι ο παρακάτω.

ΕΙΚΟΝΑ 14 ΚΩΔΙΚΑΣ LOGIN

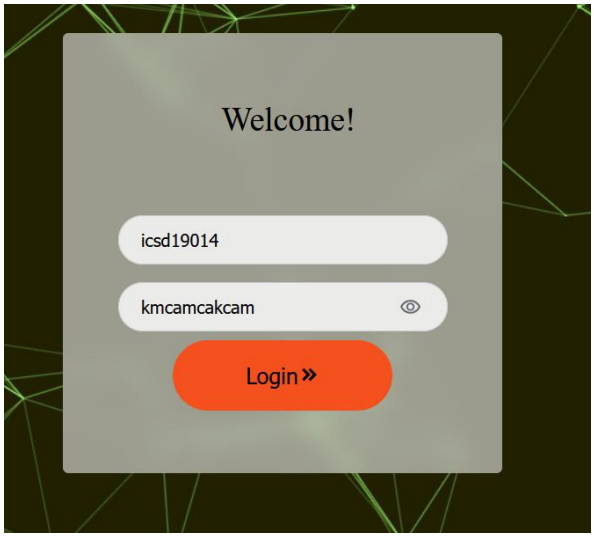
```python
@blp.route("/login",methods=["POST"])
class Users(MethodView):
    def post(self):
        username = request.json.get('username')
        password = request.json.get('password')

        connection = get_db_connection()
        if connection:
            cursor = connection.cursor(dictionary=True)
            cursor.execute("SELECT username,password1 FROM users WHERE username = %s", (username,))
            user = cursor.fetchone()
            cursor.close()
            connection.close()

            if user and user['password1'] == password:
                return jsonify({'success': True}) , 200
            else:
                return jsonify({'success': False}) ,401
        else:
            return jsonify({'success': False, 'error': 'Database connection failed'})
```

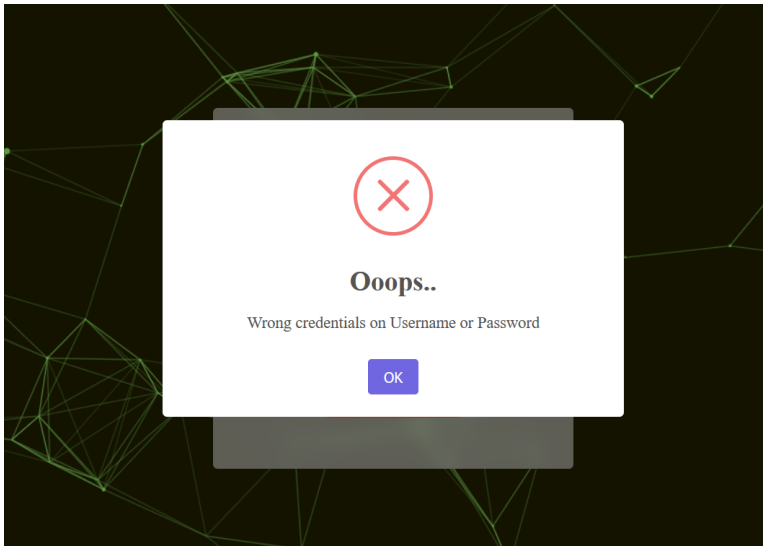
Μια μικρή εξήγηση στον κώδικα.

- Παίρνουμε από τον χρήστη το username/password που έβαλε και είναι σε μορφή JSON
- Εκτελούμε το παρακάτω SQL query και αναζητούμε στην βάση δεδομένων με βάση το username
- Τέλος, με την if αν υπάρχει user και για αυτόν τον user ο κωδικός είναι ίδιος ,επιστρέφει το μήνυμα 200 στον server, υποδηλώνοντας έτσι πως τα στοιχεία είναι σωστά και μπορεί να προχωρήσει. Αν τώρα, επιστρέψει 401, το μήνυμα για unauthorized , υποδηλώνει την περίπτωση λανθασμένων στοιχείων.

Παρακάτω είναι το αποτελέσματα login (λάθους/σωστού) και τα μηνύματα τους.

ΕΙΚΟΝΑ 15 LOGIN(401) ΜΗΝΥΜΑΤΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

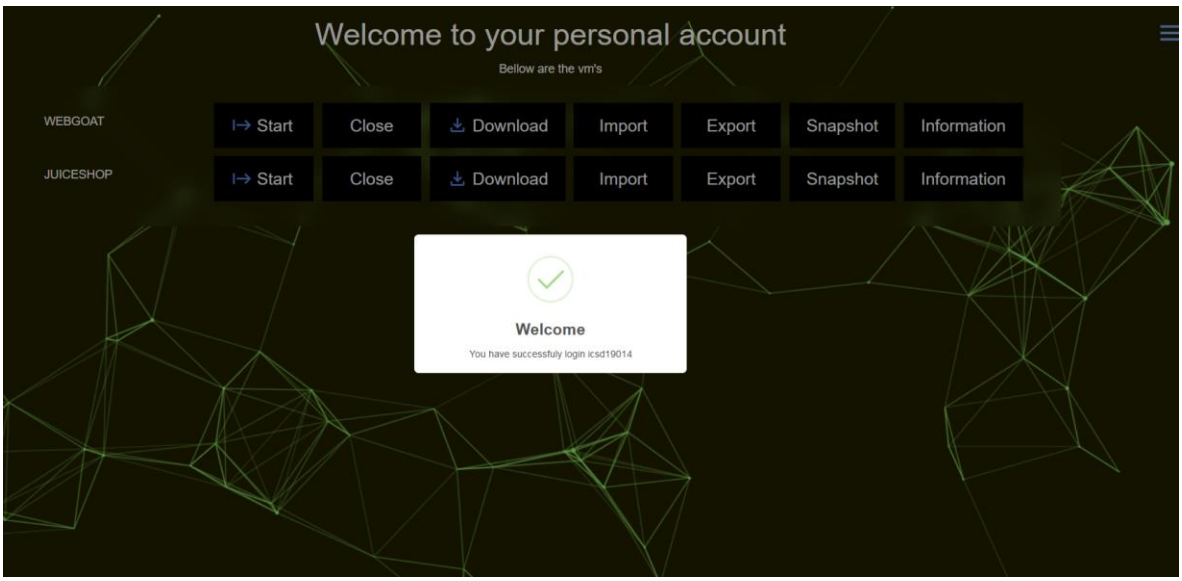


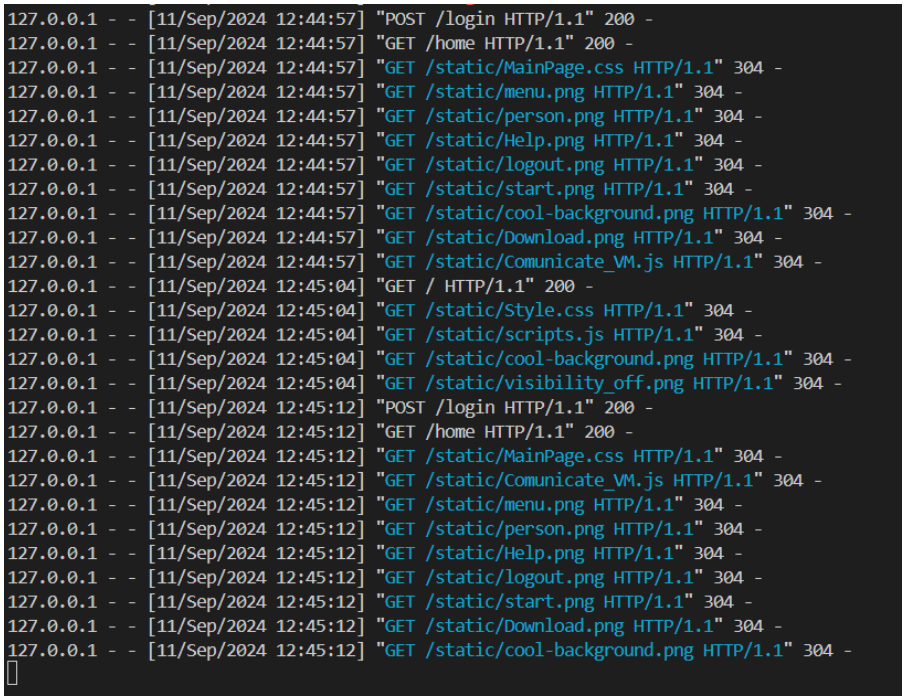



```
127.0.0.1 - - [11/Sep/2024 12:40:50] "GET /static/visibility.png HTTP/1.1" 304 -
127.0.0.1 - - [11/Sep/2024 12:40:54] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [11/Sep/2024 12:41:23] "POST /login HTTP/1.1" 401 -
```

Παρακάτω ο χρήστης βάζει τα σωστά στοιχεία στην **Εικόνα 16** και επιστρέφει μήνυμα (OK – 200)

ΕΙΚΟΝΑ 16 LOGIN(200) ΜΗΝΥΜΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ





Εφόσον ο χρήστης μπήκε με σωστά στοιχεία στην βασική πλατφόρμα, μπορεί να κάνει αρκετές ενέργειες με τα (VMs) που θα του παρέχονται. Όμως, ο χρήστης θα πρέπει να ακολουθήσει μια διαδικασία .

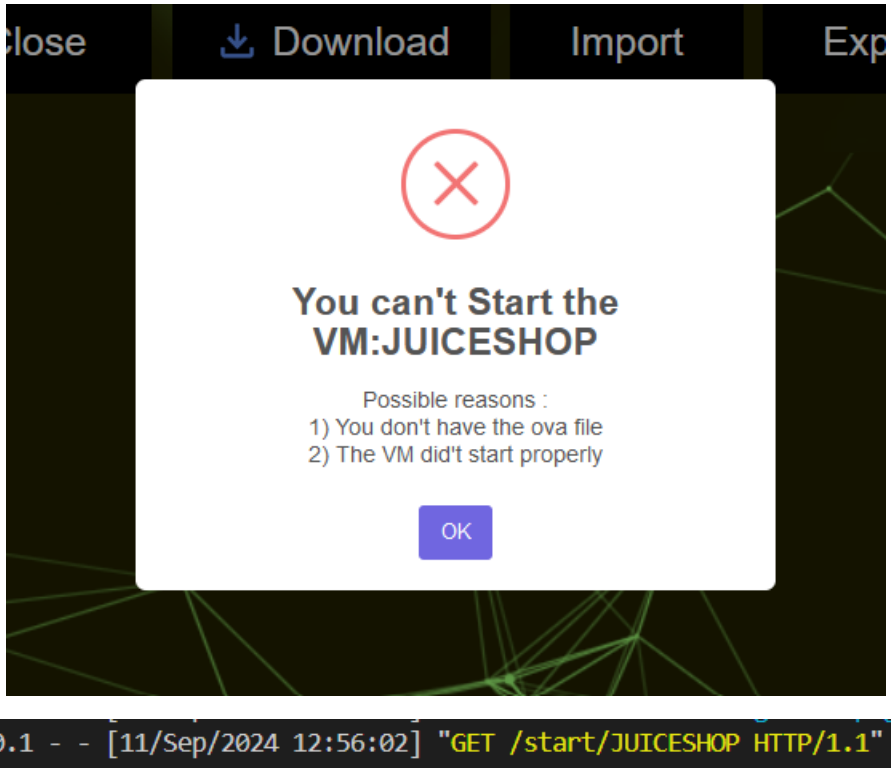
1. Πρώτα πατάει το (“Download”) για να λάβει το εκάστοτε vm.

2. Έπειτα, αφού ολοκληρωθεί η εγκατάσταση πρέπει να πατήσει το (“import”) για να βάλει το vm στο δικό του virtual box μπορεί να πάρει 2 με 3 λεπτά.

3. Εφόσον γίνουν όλα αυτά, μπορεί να πατήσει το κουμπί (“Start”) για να ξεκινήσει το vm του.

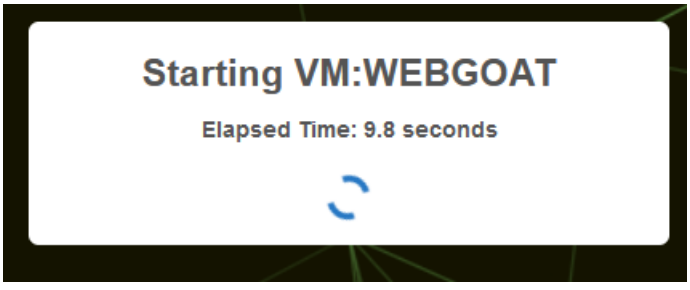
Αν δεν ακολουθήσει τα βήματα όπως είναι, δεν θα μπορεί να κάνει καμία ενέργεια και το πρόγραμμα θα εμφανίσει στον χρήστη συγκεκριμένα μηνύματα , γιατί δεν θα μπορεί να κάνει συγκεκριμένες ενέργειες.

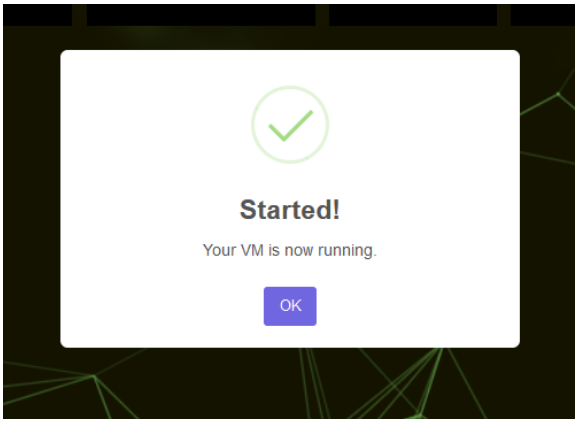
ΕΙΚΟΝΑ 17 ΠΕΡΙΠΤΩΣΗ ΠΟΥ Ο ΧΡΗΣΤΗ ΔΕΝ ΑΚΟΛΟΥΘΕΙ ΤΙΣ ΟΔΗΓΙΕΣ

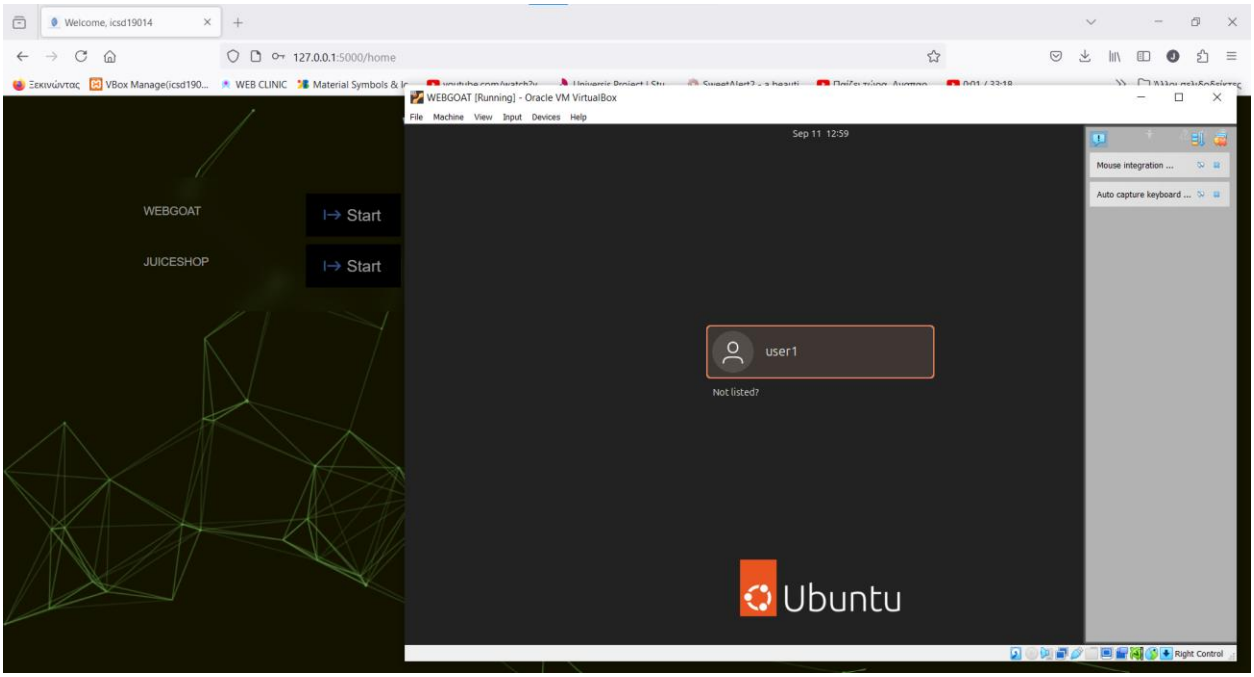


Στην περίπτωση που ξεκινήσει σωστά το VM ενημερώνει τον χρήστη πόσο χρόνο πήρε το VM να ξεκινήσει στον υπολογιστή του.

ΕΙΚΟΝΑ 18 ΞΕΚΙΝΑΕΙ Ο ΧΡΗΣΤΗΣ ΤΟ VM ΜΕ ΤΟ (.OVA)







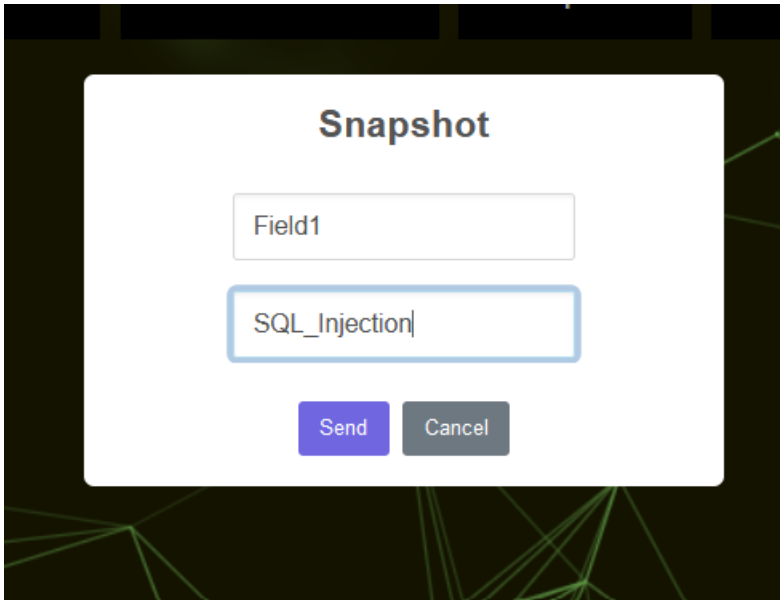
Παρακάτω είναι τα μηνύματα από την μεριά του Flask που με το URL("/status/<VM_name>") παίρνει την πληροφορία για τον χρόνο που χρειάστηκε στο VM για να ξεκινήσει στον χρήστη , καθώς στο τέλος επιστρέφει το Flask το μήνυμα για την σωστή εκκίνηση του VM.

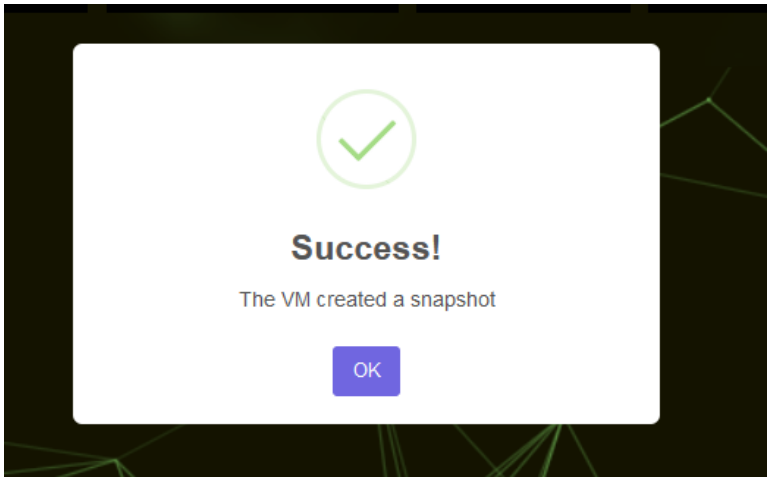
```
127.0.0.1 - - [11/Sep/2024 12:58:19] "GET /start/WEBGOAT HTTP/1.1" 200 -
Waiting for VM "WEBGOAT" to power on...
127.0.0.1 - - [11/Sep/2024 12:58:19] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:20] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:21] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:22] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:23] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:24] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:25] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:26] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:27] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:29] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:30] "GET /status/WEBGOAT HTTP/1.1" 200 -
127.0.0.1 - - [11/Sep/2024 12:58:31] "GET /status/WEBGOAT HTTP/1.1" 200 -
VM "WEBGOAT" has been successfully started.
```

6.2.2 Δημιουργία SnapShots

Εφόσον ο χρήστης φτάσει σε ένα σημείο του δίνεται και η δυνατότητα να δημιουργήσει ένα Snapshot στο VM , με αποτέλεσμα αν το VM του κολλήσει μπορεί να ανατρέξει στο Snapshot που είχε τραβήξει. Παρακάτω βλέπουμε πως μπορεί να τραβήξει Snapshots ο χρήστης με όνομα Field1 και περιγραφή SQL_Injection.

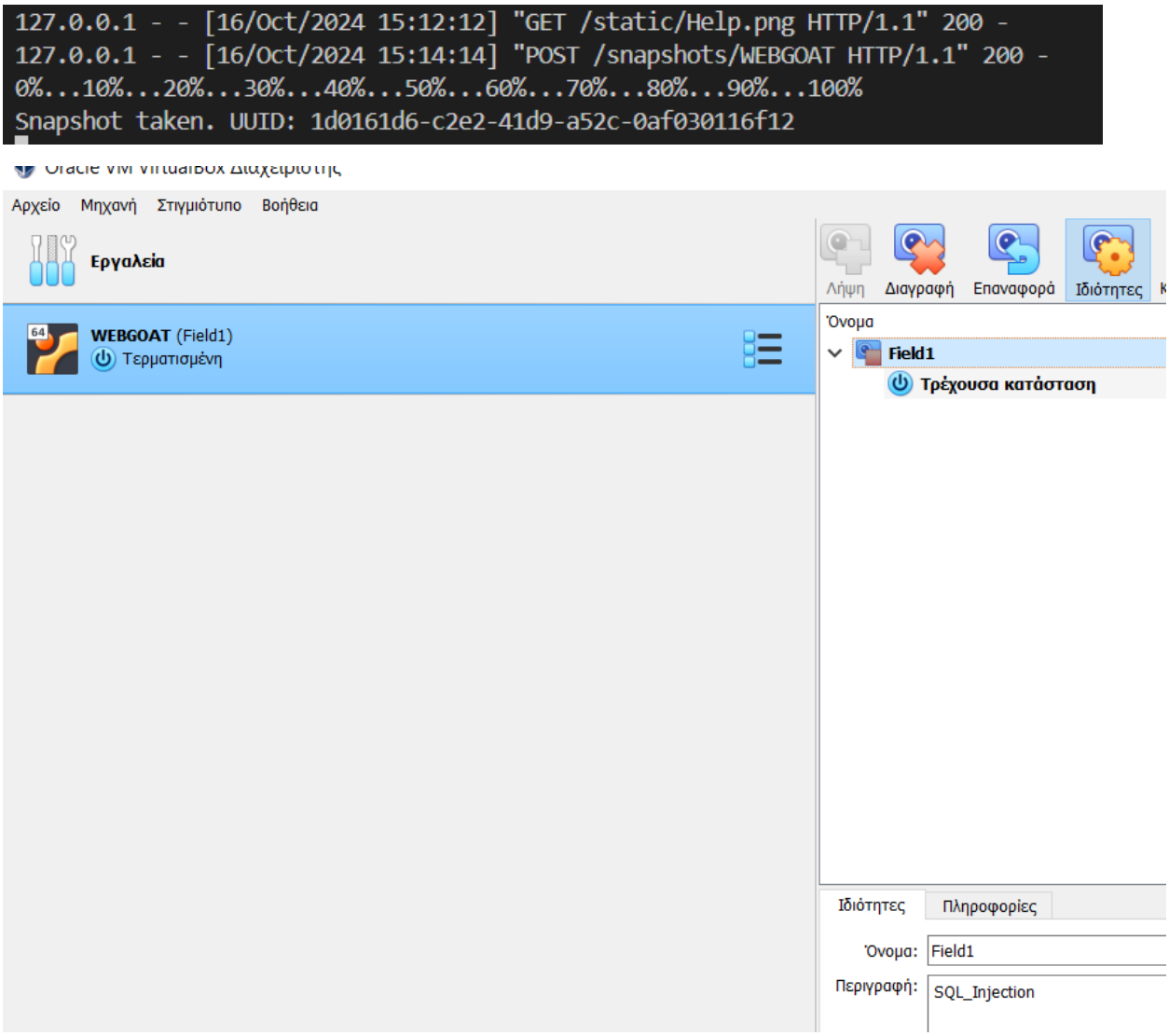
ΕΙΚΟΝΑ 19 ΔΗΜΙΟΥΡΓΙΑ SNAP SHOT





Τα μηνύματα του Flask και η μέθοδος POST στο URL (http://localhost:5000/snapshots/<VM_NAME>). Όπως βλέπουμε το Snap Shot έγινε με επιτυχία στο Web Goat .

ΕΙΚΟΝΑ 20 ΑΠΟΤΕΛΕΣΜΑ ΑΠΟ ΤΟ SNAPSHOT



7 Μελλοντικές αναβαθμίσεις

Ενέργειες που μπορούν να πραγματοποιηθούν μελλοντικά σε αυτό το project είναι οι εξής .

Δημιουργία χρήστη (Teacher) : Να βάλω και έναν δεύτερο ρόλο , με σκοπό να δημιουργεί αυτός κάποια VM που θα περιέχουν υπηρεσίες με ευπάθειες εκτός των defaults (WebGoat , Juice Shop).

"Download" κουμπί : Τα (.ova) αρχεία να είναι σε ένα Google Drive API , με στόχο την εύκολη παροχή VMs από πολλούς χρήστες και την διευκόλυνση του παραπάνω χρήστη **Teacher** , να μπορεί να διαγράφει και να βάζει τα VMs που θέλει μέσα στο Google Driver API.

Προφίλ : Να υπάρχει καταγραφή κατάστασης σκορ για το πόσα προβλήματα λύνουν οι χρήστες.

Λειτουργία VMs : Η παρούσα εργασία ανοίγει τα VMs τοπικά στον υπολογιστή του εκάστοτε χρήστη , σημαντικό θα ήταν η διαχείριση των VMs να μην γίνεται τοπικά , για να μην επιβαρύνεται ο υπολογιστής του χρήστη σε περίπτωση που δεν πληροί τις προδιαγραφές που ζητάει κάποιο αντίστοιχο VM.

8 Βιβλιογραφία

- WebGoat (OWASP)[https://owasp.org/www-project-webgoat/]
- Juice Shop (OWASP) [https://owasp.org/www-project-juice-shop/]
- Yunfa Li, Wanqing Li, Congfeng Jiang (2010). A Survey of Virtual Machine System: Current Technology and Future Trends : Grid and Service Computing Lab, School of Computer Science and Technology Hangzhou Dianzi University, 310018
- HTTP Response Status Codes [https://developer.mozilla.org/en-US/docs/Web/HTTP/Status]
- Lauren Murphy , Tosin Alliyu , Andrew Macvean , Mary Beth Kery, Brad A. Myers (2017) , Preliminary Analysis of REST API Style Guidelines
- OWASP Top Ten [https://owasp.org/www-project-top-ten/]
- Nuruldelmia Idris a, Cik Feresa Mohd Foozy a,b,1,*, Palaniappan Shamala a,b (April 2020),A Generic Review of Web Technology: DJango and Flask
- SQL Injection Bypassing WAF (OWASP) (https://owasp.org/www-community/attacks/SQL_Injection_Bypassing_WAF)