ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Αυτοματοποίηση ανάπτυξης υποδομών για την διαμόρφωση ασφαλείας συστημάτων Automation of infrastructure deployment for system security configurations
Ονοματεπώνυμο Φοιτητή	Ναλπαντίδης Κωνσταντίνος
Πατρώνυμο	Νικόλαος
Αριθμός Μητρώου	Π/ 11103
Επιβλέπων	Παναγιώτης Κοτζανικολάου, Καθηγητής

Ημερομηνία Παράδοσης Φεβρουάριος 2025

Copyright ©

Περίληψη

Στην παρούσα πτυχιακή εργασία γίνεται ανάλυση των τεχνολογιών αυτοματοποίησης υποδομών και τεχνολογιών container, με σκοπό την δημιουργία εσκεμμένα ευπαθών υποδομών για την μελέτη σεναρίων ασφάλειας και την εύρεση εργαλείων τα οποία βοηθούν στην ενίσχυση ασφάλειας αυτών. Αρχικά γίνεται ανάλυση των υφιστάμενων εργαλείων αυτοματισμού με σκοπό την επιλογή των καταλληλότερων για το επιθυμητό σενάριο. Στην συνέχεια παρουσιάζεται ο σχεδιασμός της υποδομής και η δημιουργία των επιμέρους τμημάτων της, ώστε να καταγραφούν τα χαρακτηριστικά τους και να είναι ευκολότερη η συγγραφή κώδικα για την αυτοματοποίηση τους. Αντίστοιχα αναλύεται η ασφάλεια στις web εφαρμογές και πραγματοποιείται σύγκριση γνωστών web εφαρμογών οι οποίες είναι εσκεμμένα ευπαθείς, με σκοπό τη χρήση τους σε μελέτες ασφάλειας. Βάση της ανάλυσης, επιλέγονται τόσο η εφαρμογή η οποία καλύπτει τις ανάγκες της εργασίας, όσο και του web application firewall με σκοπό την προστασία αυτής. Συμπερασματικά, η εργασία μελετά πως η αυτοματοποίηση υποδομών μπορεί να βοηθήσει στην γρηγορότερη δημιουργία υποδομών για ερευνητικούς σκοπούς και αν μπορούν να εφαρμοστούν και σε καθημερινά σενάρια όπως για παράδειγμα σε εταιρίες με ανάγκες διασφάλισης της υποδομής τους σε διαφορετικό επίπεδο κλίμακας.

Λέξεις Κλειδιά:

1. **Τεχνολογίες αυτοματοποίησης υποδομών**
2. **Τεχνολογίες container**
3. **Εσκεμμένα ευπαθείς υποδομές**
4. **Μελέτη σεναρίων ασφάλειας**
5. **Ενίσχυση ασφάλειας**
6. **Εργαλεία αυτοματισμού**
7. **Υποδομή**
8. **Συγγραφή κώδικα**
9. **Ασφάλεια web εφαρμογών**
10. **Εσκεμμένα ευπαθείς εφαρμογές**
11. **Web application firewall**
12. **Αυτοματισμός**
13. **Δημιουργία υποδομών**
14. **Ερευνητικοί σκοποί**
15. **Κλιμάκωση**

Abstract

This thesis analyzes infrastructure automation and container technologies to create deliberately vulnerable infrastructures, for the purpose of studying security scenarios and for identifying tools that enhance their security. Initially, an analysis of existing automation tools is conducted to select the most suitable ones for the desired scenario. Then, the infrastructure design and the creation of its individual components are presented to document their characteristics and facilitate the writing of code for their automation. Similarly, web application security is analyzed, and a comparison of well-known deliberately vulnerable web applications is carried out to determine their usability in security studies. Based on this analysis, both the application that meets the project's needs and the web application firewall for its protection are selected. In conclusion, this study examines how infrastructure automation can accelerate the creation of infrastructures for research purposes and whether it can also be applied to everyday scenarios, such as companies needing to secure their infrastructure at different scale levels.

Key Words:

1. **Infrastructure automation technologies**
2. **Container technologies**
3. **Intentionally vulnerable infrastructures**
4. **Study of security scenarios**
5. **Security enhancement**
6. **Automation tools**
7. **Infrastructure**
8. **Code development**
9. **Web application security**
10. **Intentionally vulnerable applications**
11. **Web application firewall**
12. **Automation**
13. **Infrastructure creation**
14. **Research purposes**
15. **Scaling**

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

1. Εισαγωγή και Βασικοί Ορισμοί

1.1. Υποδομή ως κώδικας (Infrastructure as Code / IaC)

Το αντικείμενο μελέτης της εργασίας αυτής είναι η χρήση τεχνολογιών αυτοματισμού για την δημιουργία και ρύθμιση υποδομών σε δοκιμαστικά και ερευνητικά σενάρια καθώς και η χρήση αυτών, ως βάση για χρήση σε εφαρμογές σε παραγωγικό περιβάλλον.

Σε συνδυασμό με την αυτοματοποίηση των υπό μελέτη σεναρίων, θα διεξαχθούν δοκιμαστικές επιθέσεις σε εσκεμμένα ευπαθείς εφαρμογές με στόχο την μελέτη της αποτελεσματικότητας των Web Application Firewalls[1] στην αποτροπή επιθέσεων σε web εφαρμογές[2].

Με την πάροδο των χρόνων και την ραγδαία ανάπτυξη του διαδικτύου ακολούθησε και η ευκολία στην πρόσβαση αυτού. Παρά τα τεράστια θετικά που έφερε αυτό το γεγονός, δημιουργήθηκαν και δυσκολίες και προβλήματα. Δύο προβλήματα από αυτά θα μας απασχολήσουν στην εργασία αυτή.

Αρχικά η αυξημένη κίνηση στις εφαρμογές δημιουργεί την ανάγκη για υποδομές που θα ανταπεξέρχονται σε αυτήν χωρίς να επηρεάζεται η απόδοση και απόκριση αυτής αλλά και να επανέρχεται στις αρχική της κατάσταση όταν αυτές οι ανάγκες δεν υπάρχουν πλέον και ο σκοπός είναι να μειωθούν τα κόστη. Οι υποδομές πρέπει να δημιουργούνται με γρήγορο, ασφαλή και αυτοματοποιημένο τρόπο για την αντιμετώπιση της κίνησης αυτής με τεχνολογίες που κάθε μια επιτρέπει την δημιουργία ή παραμετροποίηση της υποδομής και τις εκάστοτε εφαρμογής που εξυπηρετεί.

Το δεύτερο πρόβλημα είναι το ότι οι υποδομές και εφαρμογές αυτές είναι παγκοσμίου σκέλους, προσβάσιμες από παντού και με δεδομένα χρηστών τα οποία τις καθιστούν στόχο για κακόβουλους χρήστες. Έτσι δημιουργείται η ανάγκη για συστήματα και κώδικα ο οποίος είναι ασφαλής. Όπου δεν είναι αυτό αρκετό όμως υπάρχουν οι εφαρμογές τύπου Web Application Firewall(WAF) που βάση κανόνων σταματούν την κίνηση προστατεύοντας με τον τρόπο αυτό από επιθέσεις.

1.2. Υποδομή ως κώδικας (Infrastructure as Code / IaC)

Η μέθοδος της Υποδομής ως Κώδικα[3], αναφέρεται στη χρήση εφαρμογων που μέσω κώδικα και scripts κάνουν την δημιουργία και διαχείριση μιας υποδομής πιο εύκολη και συνεπή σε αντίθεση με χειροκίνητες διαδικασίες. Αυτή η προσέγγιση προσφέρει πολλά θετικά όπως:

- **Αυτοματοποίηση, επαναληψιμότητα και συνέπεια**: Η δυνατότητα επανάληψης της ίδιας διαδικασίας χωρίς διαφοροποιήσεις εγγυάται τη συνέπεια στις υποδομές και μειώνει τα λάθη. Ο στόχος είναι η ύπαρξη αμεταβλητότητας για τα συστήματα, να υπάρχει δηλαδή η διασφάλιση ότι η υποδομή είναι πάντα σταθερή και αλλάζει μόνο μέσω εγκεκριμένου κώδικα. Αυτή η προσέγγιση μειώνει τον κίνδυνο απροβλέπτων προβλημάτων λόγω αλλαγών στο περιβάλλον, βοηθάει στην πρόβλεψη της συμπεριφοράς των συστημάτων και κάνει τη διαχείριση της υποδομής πιο απλή και ασφαλή.
- **Διαφάνεια και Καταγραφή**: Ο κώδικας που διαχειρίζεται τις υποδομές μπορεί να τεκμηριώσει ακριβώς τις ρυθμίσεις και τις αλλαγές, παρέχοντας διαφάνεια στη διαδικασία.
- **Ταχύτητα Δημιουργίας Υποδομών και Γρηγορότερη Διάθεση Πόρων**: Η ταχύτητα στη δημιουργία και διαχείριση των υποδομών είναι καθοριστική τόσο για την ερευνητική πρόοδο όσο και για την επιχειρηματική αποδοτικότητα. Επιταχύνει την ανάπτυξη νέων υποδομών, ερευνητικών και δοκιμαστικών σεναρίων και παράλληλα μειώνει τις εργατώρες και τα λειτουργικά κόστη. Επιπλέον επιτρέπει τη γρήγορη και αποδοτική διάθεση νέων πόρων, προσαρμόζοντας τις υποδομές στις ανάγκες της στιγμής με μεγάλη ευελιξία.

- **Λοιπές θετικά**: Κάποια επιπλέον θετικά με κυρίως επαγγελματικές εφαρμογές είναι για την υλοποίηση rolling updates σε εφαρμογές και συστηματα με αποτέλεσμα να μην υπάρχει παρεμβολή στην σωστή και διαρκή λειτουργία μιας υπηρεσίας, την δοκιμή νέων χαρακτηριστικών μέσω A/B testing ή να βοηθήσει στο scalling μιας εφαρμογής ανάλογα την επισκεψιμότητα με πολύ μικρή δυσκολία καθώς υποδομές, ρυθμίσεις κλπ βρίσκονται ήδη σε μορφή κώδικά. Τέλος, με την ύπαρξη modules είναι πιο εύκολο μια υποδομή να μεταφερθεί από έναν πάροχο σε κάποιον άλλο ή ακόμα και να γίνει χρήση πολλών παράλληλα.

Παράλληλα με τα θετικά όμως υπάρχουν και αρνητικά. Η ανάγκη για όλο και πιο περίπλοκα συστήματα που θα ανταποκρίνονται σε περισσότερο κόσμο επαρκώς αποδοτικά αλλά και η ίδια η πολυπλοκότητα των εργαλείων είναι κάποια από τα θέματα που αντιμετωπίζουν οι μηχανικοί καθημερινά. Συγκεκριμένα έχουμε:

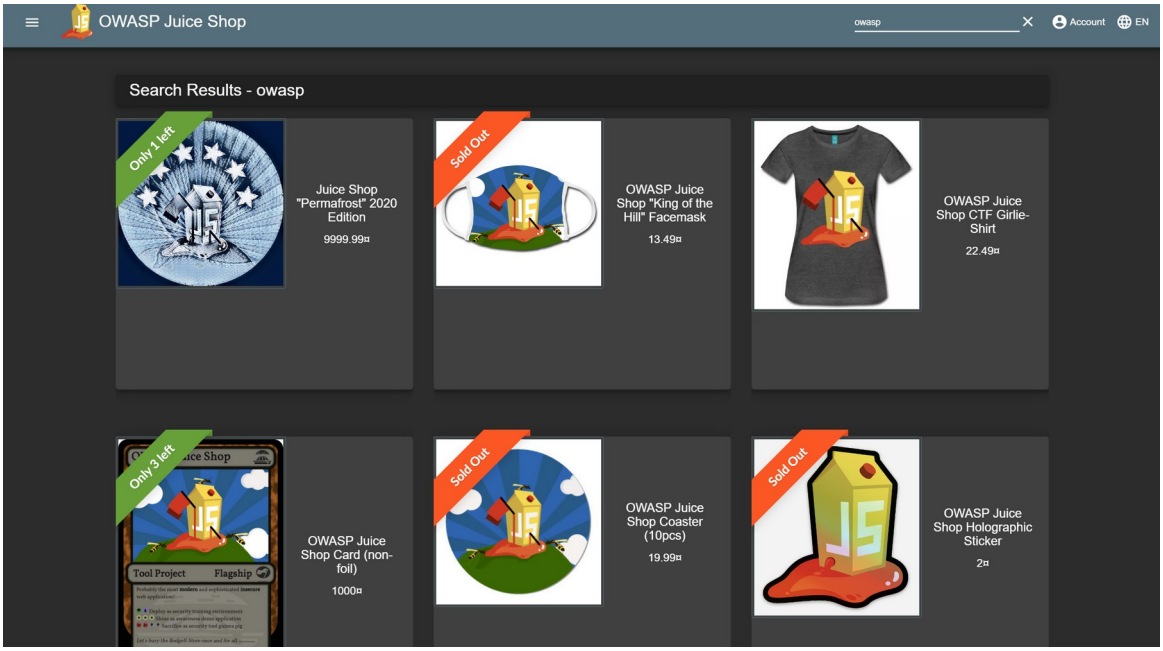
- **Πολυπλοκότητα**: Αν και η χρήση τεχνολογιών IaC γίνεται για την γρηγορότερη και ευκολότερη δημιουργία υποδομών, συνδυάζοντας πολλά από αυτά τα εργαλεία στα διαφορετικά στάδια της δόμησης της(υποδομή, σύστημα, σερβίρισμα εφαρμογών) αλλά και περαιτέρω χρήση εργαλείων όπως CI/CD λογισμικού (δεν θα γίνει μελέτη τους σε αυτήν την εργασία) κάνουν την ανάληψη αυτών εκθετικά πιο δύσκολη. Για τον λόγο αυτό θέσεις εργασίας σχετικές με το αντικείμενο όπως DevOps ή Platform engineers απαιτούν πολλά χρόνια εμπειρίας και οι εταιρίες που απασχολούν τέτοιους μηχανικούς ενδέχεται να χρειαστεί να επενδύσουν σε επιπλέον εκπαίδευση ή να προσλάβουν προσωπικό με τις απαιτούμενες γνώσεις.
- **Προκλήσεις Διαχείρισης Εκδόσεων**: Η μετάβαση σε τεχνολογίες IaC μπορεί να φέρει αντίσταση σε ομάδες που είναι συνηθισμένες σε χειροκίνητες διαδικασίες, κάτι που προκύπτει συνήθως από έλλειψη δεξιοτήτων. Αυτό συμβαίνει γιατί η διατήρηση της κατάστασης της υποδομής γίνεται περίπλοκη, ειδικά με εργαλεία που βασίζονται σε αρχεία κατάστασης (π.χ., Terraform). Επιπλέον όταν γίνονται χειροκίνητες αλλαγές εκτός του IaC, η εξασφάλιση ότι ο κώδικας αντικατοπτρίζει την πραγματική κατάσταση δεν μπορεί να εγγυηθεί και αυτό να οδηγήσει σε προβλήματα.
- **Κίνδυνοι Ασφάλειας λόγω μυστικών ή λανθασμένων ρυθμίσεων**: Εάν δεν διαχειριστούν σωστά τα μυστικά ή τα κλειδιά των εφαρμογών, ο κώδικας του IaC μπορεί να εκθέσει ευαίσθητες πληροφορίες, όπως API κλειδιά, κωδικούς πρόσβασης ή SSH κλειδιά. Επιπλέον εσφαλμένες ρυθμίσεις πολιτικών ή δικαιωμάτων στον κώδικα μπορεί να οδηγήσει σε υπερβολικά δικαιώματα πρόσβασης τα οποία μαζί με κάποια άλλη αδυναμία να δώσουν σε επιτιθέμενο πλήρη πρόσβαση στο σύστημα μας.

1.3. Web applications

Μια εφαρμογή ιστού (web application ή web app) είναι ένα λογισμικό εφαρμογής που δημιουργείται με τεχνολογίες ιστού και εκτελείται μέσω ενός περιηγητή ιστού. Οι εφαρμογές ιστού εμφανίστηκαν στα τέλη της δεκαετίας του 1990 και έδωσαν τη δυνατότητα στους διακομιστές να δημιουργούν δυναμικά απαντήσεις σε αιτήματα, σε αντίθεση με τις στατικές ιστοσελίδες.

Οι εφαρμογές ιστού διανέμονται συνήθως μέσω ενός διακομιστή ιστού. Υπάρχουν διάφορα συστήματα επιπέδων που χρησιμοποιούν οι εφαρμογές ιστού για την επικοινωνία μεταξύ περιηγητών ιστού, της διεπαφής χρήστη και των δεδομένων του διακομιστή. Κάθε σύστημα έχει διαφορετική χρήση καθώς λειτουργεί με διαφορετικό τρόπο. Ωστόσο, υπάρχουν πολλοί κίνδυνοι ασφάλειας που πρέπει να λαμβάνουν υπόψη οι προγραμματιστές κατά την ανάπτυξη.Η λήψη κατάλληλων μέτρων για την προστασία των δεδομένων των χρηστών είναι ζωτικής σημασίας.

Οι εφαρμογές ιστού συχνά κατασκευάζονται με τη χρήση ενός framework εφαρμογών ιστού. Οι προσεγγίσεις single-page και progressive επιτρέπουν σε έναν ιστότοπο να μοιάζει περισσότερο με μια εγγενή εφαρμογή.Στο μεγαλύτερο ποσοστό τους οι εφαρμογές ιστού για να λειτουργήσουν χρειάζονται και μια βάση δεδομένων για την αποθήκευση στοιχείων και της ίδιας της εφαρμογής αλλά και για να αποθηκεύσουν τα στοιχεία των χρηστών. Μέσω των αδυναμιών των εφαρμογών σε πολλές περιπτώσεις υπάρχουν διαρροές στοιχείων από τις βάσεις.



Εικόνα 1. Παράδειγμα web application

Πηγή: https://raw.githubusercontent.com/juice-shop/juice-shop/master/screenshots/screenshot01.png

1.4. Databases

Στην πληροφορική, μια βάση δεδομένων[4] (database) είναι μια οργανωμένη συλλογή δεδομένων ή ένας τύπος αποθήκης δεδομένων που βασίζεται στη χρήση ενός συστήματος διαχείρισης βάσεων δεδομένων (DBMS). Το DBMS είναι το λογισμικό που αλληλεπιδρά με τους τελικούς χρήστες, τις εφαρμογές και τη βάση δεδομένων για τη συλλογή και ανάλυση δεδομένων.

Επιπλέον, το DBMS παρέχει τις βασικές δυνατότητες για τη διαχείριση της βάσης δεδομένων. Το σύνολο της βάσης δεδομένων, του DBMS και των συνδεδεμένων εφαρμογών αναφέρεται συχνά ως σύστημα βάσεων δεδομένων. Ο όρος "βάση δεδομένων" χρησιμοποιείται συχνά και για να περιγράψει το DBMS, το σύστημα βάσεων δεδομένων ή μια εφαρμογή που σχετίζεται με τη βάση.

Μικρές βάσεις δεδομένων μπορούν να αποθηκευτούν σε ένα σύστημα αρχείων, ενώ οι μεγάλες βάσεις φιλοξενούνται σε συστοιχίες υπολογιστών (clusters) ή σε cloud storage. Ο σχεδιασμός των βάσεων δεδομένων περιλαμβάνει τυπικές τεχνικές και πρακτικές πτυχές, όπως η μοντελοποίηση δεδομένων, η αποδοτική αναπαράσταση και αποθήκευση δεδομένων, οι γλώσσες ερωτημάτων, η ασφάλεια και το απόρρητο ευαίσθητων δεδομένων, καθώς και ζητήματα κατανεμημένης πληροφορικής, όπως η υποστήριξη ταυτόχρονης πρόσβασης και η ανοχή σε σφάλματα.

Οι επιστήμονες υπολογιστών κατατάσσουν τα συστήματα διαχείρισης βάσεων δεδομένων (DBMS) με βάση τα μοντέλα βάσεων δεδομένων που υποστηρίζουν. Οι σχεσιακές βάσεις δεδομένων κυριάρχησαν τη δεκαετία του 1980. Αυτές αναπαριστούν τα δεδομένα ως σειρές και στήλες σε πίνακες, και η πλειονότητα τους χρησιμοποιεί SQL για την εγγραφή και ανάκτηση δεδομένων. Τη δεκαετία του 2000, έγιναν δημοφιλείς οι μη σχεσιακές βάσεις δεδομένων (NoSQL), οι οποίες χρησιμοποιούν διαφορετικές γλώσσες ερωτημάτων.

```
dvdrental=# select title, release_year, length, replacement_cost from film
dvdrental-#   where length > 120 and replacement_cost > 29.50
dvdrental-#   order by title desc;
          title          | release_year | length | replacement_cost
-------------------------+--------------+--------+------------------
 West Lion               |         2006 |    159 |            29.99
 Virgin Daisy            |         2006 |    179 |            29.99
 Uncut Suicides          |         2006 |    172 |            29.99
 Tracy Cider             |         2006 |    142 |            29.99
 Song Hedwig             |         2006 |    165 |            29.99
 Slacker Liaisons        |         2006 |    179 |            29.99
 Sassy Packer            |         2006 |    154 |            29.99
 River Outlaw            |         2006 |    149 |            29.99
 Right Cranes            |         2006 |    153 |            29.99
 Quest Mussolini         |         2006 |    177 |            29.99
 Poseidon Forever        |         2006 |    159 |            29.99
 Loathing Legally        |         2006 |    140 |            29.99
 Lawless Vision          |         2006 |    181 |            29.99
 Jingle Sagebrush        |         2006 |    124 |            29.99
 Jericho Mulan           |         2006 |    171 |            29.99
 Japanese Run            |         2006 |    135 |            29.99
 Gilmore Boiled          |         2006 |    163 |            29.99
 Floats Garden           |         2006 |    145 |            29.99
 Fantasia Park           |         2006 |    131 |            29.99
 Extraordinary Conquerer |         2006 |    122 |            29.99
 Everyone Craft          |         2006 |    163 |            29.99
 Dirty Ace               |         2006 |    147 |            29.99
 Clyde Theory            |         2006 |    139 |            29.99
 Clockwork Paradise      |         2006 |    143 |            29.99
 Ballroom Mockingbird    |         2006 |    173 |            29.99
(25 rows)
```

Εικόνα 2. Παράδειγμα query και αποτέλεσμα

Πηγή: https://en.wikipedia.org/wiki/File:DVD_Rental_Query.png

1.5. Vulnerabilities

Μια ευπάθεια[5] είναι ένα κενό ή μια αδυναμία σε μια εφαρμογή, η οποία μπορεί να προκύπτει από σχεδιαστικό σφάλμα ή σφάλμα υλοποίησης. Αυτή η αδυναμία επιτρέπει σε έναν επιτιθέμενο να προκαλέσει ζημιά στους ενδιαφερόμενους της εφαρμογής. Οι ενδιαφερόμενοι μπορεί να περιλαμβάνουν τον κάτοχο της εφαρμογής, τους χρήστες της εφαρμογής, καθώς και άλλες οντότητες που βασίζονται στην εφαρμογή.

Παραδείγματα ευπαθειών:

- Έλλειψη ελέγχου εγκυρότητας στην εισαγωγή δεδομένων από τον χρήστη.
- Έλλειψη επαρκούς μηχανισμού καταγραφής συμβάντων (logging).
- Χειρισμός σφαλμάτων με μη ασφαλή τρόπο (fail-open).
- Μη σωστό κλείσιμο της σύνδεσης με τη βάση δεδομένων.

1.6. Web Application Firewall

Ένα Web Application Firewall (WAF) είναι μια εξειδικευμένη μορφή τείχους προστασίας εφαρμογών που φιλτράρει, παρακολουθεί και αποκλείει την κίνηση HTTP προς και από μια υπηρεσία ιστού. Με την ανάλυση της HTTP κίνησης, το WAF μπορεί να αποτρέψει επιθέσεις που εκμεταλλεύονται γνωστές ευπάθειες εφαρμογών ιστού, όπως SQL Injection, Cross-Site Scripting (XSS), File Inclusion και ακατάλληλη διαμόρφωση συστημάτων.

Τα περισσότερα μεγάλα χρηματοπιστωτικά ιδρύματα χρησιμοποιούν WAFs για τη μείωση των ευπαθειών zero-day στις εφαρμογές ιστού, καθώς και για την αντιμετώπιση σφαλμάτων ή αδυναμιών που είναι δύσκολο να επιδιορθωθούν, μέσω προσαρμοσμένων υπογραφών επιθέσεων.

Τα WAFs συνήθως ακολουθούν ένα θετικό μοντέλο ασφάλειας, ένα αρνητικό μοντέλο ασφάλειας ή έναν συνδυασμό των δύο, όπως αναφέρει το SANS Institute. Τα WAFs χρησιμοποιούν έναν συνδυασμό λογικής που βασίζεται σε κανόνες, parsing και υπογραφές για την ανίχνευση και αποτροπή επιθέσεων, όπως Cross-Site Scripting (XSS) και SQL Injection.

Γενικά, τεχνικές όπως εξομοίωση περιηγητή, συγκαλυμμένη παρουσίαση δεδομένων (obfuscation), εικονικοποίηση και απόκρυψη IP χρησιμοποιούνται για να επιχειρήσουν την παράκαμψη ενός WAF. Ο OWASP[6] δημοσιεύει μια λίστα με τις δέκα κορυφαίες ευπάθειες ασφαλείας εφαρμογών ιστού (OWASP Top Ten), και όλες οι εμπορικές λύσεις WAF καλύπτουν τουλάχιστον αυτές τις ευπάθειες. Υπάρχουν επίσης και μη εμπορικές επιλογές, όπως η γνωστή ανοιχτού κώδικα μηχανή WAF, ModSecurity.

Ωστόσο, η μηχανή WAF από μόνη της δεν επαρκεί για να παρέχει επαρκή προστασία. Για αυτόν τον λόγο, ο OWASP σε συνεργασία με την Spiderlabs της Trustwave οργανώνουν και διατηρούν ένα Core-Rule Set μέσω του GitHub, το οποίο μπορεί να χρησιμοποιηθεί με τη μηχανή ModSecurity.

1.7. Σκοπός και στόχοι

Στην παρούσα εργασία θα αναλυθεί το κομμάτι της αυτοματοποίησης της υποδομής ως κώδικα. Η αυτοματοποίηση στην εργασία μπορεί να χωριστεί σε κάποια επιμέρους κομμάτια. Αρχικά την υλοποίηση της υποδομής, δηλαδή τα εικονικά μηχανήματα στα οποία θα τρέξουν οι εφαρμογές καθώς και οι δικτυακές ανάγκες που υπάρχουν για αυτά, firewall αλλά και επικοινωνία μεταξύ των εφαρμογών. Στην συνέχεια θα χρειαστεί να αυτοματοποιηθούν οι αλλαγές και ρυθμίσεις του λειτουργικού συστήματος που θα χρησιμοποιηθεί και τέλος θα γίνει η εγκατάσταση και αυτοματοποίηση των ευπαθών εφαρμογών καθώς και των WAF τα οποία θα βοηθήσουν στις επιθέσεις που θα διεξαχθούν.

Το υπό μελέτη πρόβλημα επικεντρώνεται στην αυτοματοποίηση της δημιουργίας και διαχείρισης υποδομών πληροφορικής, με ιδιαίτερη έμφαση στην υλοποίηση των συστημάτων, τη δικτυακή σύνδεση και την εγκατάσταση ευπαθών εφαρμογών, καθώς και των Web Application Firewalls (WAF) για τη δοκιμή επιθέσεων στο πλαίσιο της ασφάλειας ιστοσελίδων. Η διαδικασία αυτή απαιτεί την ακριβή ανάλυση των βημάτων, σχεδιασμού της εργασίας και υλοποίησης αυτής. Στην συνέχεια η αποδοτική αυτοματοποίηση των βημάτων θα βοηθήσει στην σύγκριση των δυο διαδικασιών ώστε να υπάρξει συμπέρασμα για την αποδοτικότητα ή μη της μεθοδολογίας. Το κύριο ενδιαφέρον έγκειται στην ανάπτυξη μιας ολοκληρωμένης και ευέλικτης προσέγγισης που θα επιτρέπει την γρήγορη και ασφαλή δοκιμή διαφορετικών σεναρίων επιθέσεων σε web εφαρμογές, μέσω της αυτοματοποιημένης υλοποίησης και διαχείρισης της σχετικής υποδομής.

1.8. Ανάλυση κεφαλαίων

Κεφάλαιο 2: Ανάλυση τεχνολογιών για την αυτοματοποίηση υποδομών

Στο κεφάλαιο αυτό θα μελετηθούν οι τεχνολογίες που είναι σχετικές με το IaC. Θα γίνει διαχωρισμός βάση του σταδίου στο οποίο θα γίνει χρήση τους και τον σκοπό που εξυπηρετούν καθώς και σύγκριση των χαρακτηριστικών ώστε να γίνει και η ανάλογη επιλογή για την περίπτωση που χρειάζεται.

Κεφάλαιο 3: Σενάρια υποδομών, σχεδιασμός και αυτοματοποίηση

Αρχικά θα γίνει ανάλυση μιας σειράς σεναρίων τα οποία χρησιμοποιούνται στην καθημερινότητα από οργανισμούς για την εξυπηρέτηση των επαγγελματικών τους αναγκών μαζί με σύγκριση αυτών.

Στην συνέχεια θα γίνει η δημιουργία της υποδομής στον πάροχο VPS που θα επιλεχθεί και η συλλογή των στοιχείων των εικονικών μηχανών(δίσκος, CPU, RAM, γεωγραφική τοποθεσία εάν υπάρχει κλπ.) και θα γίνει συγγραφή κώδικα με την τεχνολογία που θα επιλέξουμε για την αυτόματη δημιουργία τους..

Τα ίδια βήματα θα ακολουθηθούν ακριβώς και για την παραμετροποίηση των εικονικών μηχανών. Αρχικά χειροκίνητα και στην συνέχεια μετάφραση των ρυθμίσεων σε κώδικα και τέλος για το σερβίρισμα της εφαρμογής με και χωρίς WAF.

Για να κλείσει το μέρος της αυτοματοποίησης θα ενώσουμε τα εργαλεία μεταξύ τους με modules ώστε να καλεί το κάθε ένα το επόμενο και η διαδικασία να γίνει τελείως αυτόματα.

Κεφάλαιο 4: Ενίσχυση ασφάλειας υποδομών με την εφαρμογή τεχνολογιών υποδομής ως κώδικα

Στο κεφάλαιο αυτό θα μελετηθούν οι τεχνολογίες που είναι σχετικές με την ασφάλεια των web εφαρμογών. Θα δωθούν παραδείγματα για εφαρμογές με ευάλωτο περιεχόμενο και θα γίνει σύγκριση των χαρακτηριστικών τους και σχολιασμός καταλληλότητας τους αναλόγως του σεναρίου.

Επιπλέον θα γίνει σύγκριση εργαλείων WAF τα οποία θα βοηθήσουν στο επόμενο κεφάλαιο κατά την επίδειξη των επιθέσεων.

Όπως και στο κεφάλαιο 3 θα γίνει ανάλυση κάποιον παραδειγμάτων από επιθέσεις με διαφορετικούς στόχους όπως βάσεις δεδομένων ή ldap servers. Αφού γίνει μελέτη της μεθοδολογίας από τα παραπάνω παραδείγματα θα γίνει επιλογή μιας από τις διαθέσιμες επιθέσεις που επιτρέπει η εφαρμογή και θα γίνει εκτέλεση των βημάτων για την εκμετάλλευση αυτής.

Στην συνέχεια θα γίνει ενεργοποίηση του WAF και δοκιμή για την εκτέλεση των παραπάνω βημάτων με σκοπό την ανάλυση της λειτουργίας προστασίας.

Κεφάλαιο 5: Συμπεράσματα

Στο τελευταίο κεφάλαιο θα γίνει τελικός σχολιασμός των παραπάνω κεφαλαίων, χρησιμότητα των εργαλείων που μελετήθηκαν στην εργασία καθώς και πρακτικές εφαρμογές αυτών σε ερευνητικά και επαγγελματικά σενάρια.

2. Ανάλυση τεχνολογιών για την αυτοματοποίηση υποδομών

Στο κεφάλαιο αυτό θα γίνει μελέτη των τεχνολογιών που θα χρησιμοποιηθούν. Θα γίνει διαχωρισμός σε δύο μέρη, το πρώτο θα είναι για το λογισμικό γύρω από την αυτοματοποίηση των συστημάτων με εναλλακτικές και το δεύτερο γύρω από τις τεχνολογίες για τις δοκιμές ασφαλείας που θα δούμε στο δεύτερο μέρος της εργασίας.

2.1 Διαχείριση διαμόρφωσης (Configuration Management)

Η διαχείριση διαμόρφωσης[7] είναι μια διαδικασία για τη διατήρηση μιας επιθυμητής κατάστασης σε συστήματα και στοιχεία πληροφορικής. Βοηθά στη διασφάλιση ότι ένα σύστημα αποδίδει σταθερά όπως αναμένεται καθ' όλη τη διάρκεια του κύκλου ζωής του.

Οι διαχειριστές συστημάτων μπορούν να χρησιμοποιούν εργαλεία διαχείρισης διαμόρφωσης για τη ρύθμιση ενός συστήματος πληροφορικής, όπως ένας διακομιστής ή ένας σταθμός εργασίας, και στη συνέχεια να δημιουργούν και να διατηρούν άλλους διακομιστές και σταθμούς εργασίας με τις ίδιες ρυθμίσεις. Επιπλέον, μπορούν να πραγματοποιούν αξιολογήσεις διαμόρφωσης και αναλύσεις αποκλίσεων για να εντοπίζουν συστήματα που έχουν αποκλίνει από την επιθυμητή κατάσταση και χρειάζονται ενημέρωση, αναδιαμόρφωση ή επιδιόρθωση.

Στα εταιρικά περιβάλλοντα, οι ομάδες πληροφορικής διαχειρίζονται ένα ευρύ φάσμα εφαρμογών και συστημάτων, όπως cloud, δίκτυα, αποθηκευτικά μέσα, διακομιστές και edge συσκευές. Εκτός από τη σωστή αρχική ρύθμιση των συστημάτων, η τακτική και λεπτομερής συντήρηση είναι εξίσου σημαντική, καθώς βοηθά στην πρόληψη πιο δαπανηρών προβλημάτων στο μέλλον.

Η εισαγωγή μικρών ασυνεπειών ή σφαλμάτων διαμόρφωσης στα συστήματα πληροφορικής μπορεί να οδηγήσει σε απόκλιση διαμόρφωσης (configuration drift), γεγονός που μπορεί να προκαλέσει επιβράδυνση συστημάτων, κινδύνους ασφάλειας και συμμόρφωσης, ακόμη και διακοπές λειτουργίας.

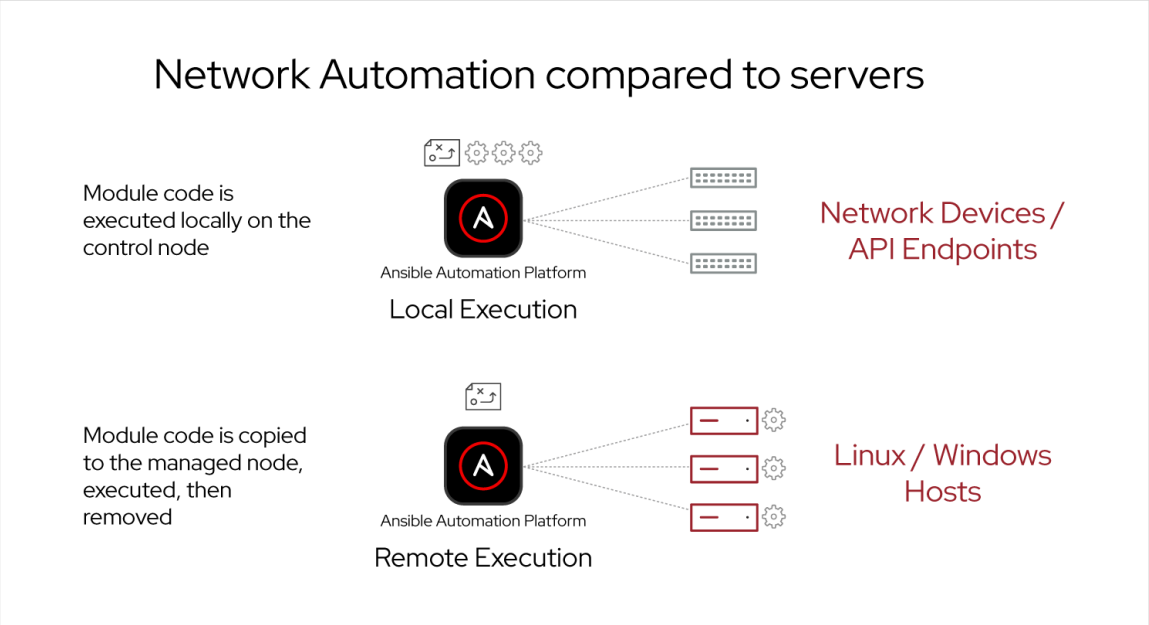
Για την πρόληψη αυτών των προβλημάτων, η διαχείριση διαμόρφωσης περιλαμβάνει τη θέσπιση μιας σαφούς προσέγγισης για την τεκμηρίωση, τη συντήρηση και τον έλεγχο αλλαγών, ώστε τα συστήματα να διαμορφώνονται με συνέπεια και ακρίβεια σε σύνθετα περιβάλλοντα.

- Ansible

Το Ansible[8] είναι μια εφαρμογή αυτοματισμού ανοιχτού κώδικα, που λειτουργεί μέσω γραμμής εντολών και έχει γραφτεί σε Python. Μπορεί να διαμορφώνει συστήματα, να αναπτύσσει λογισμικό και να οργανώνει προχωρημένες ροές εργασίας για να υποστηρίξει την ανάπτυξη εφαρμογών, ενημερώσεις συστημάτων κλπ.

Οι κύριες δυνάμεις του Ansible είναι η απλότητα και η ευκολία χρήσης. Έχει επίσης έναν ισχυρό προσανατολισμό προς την ασφάλεια και την αξιοπιστία, διαθέτοντας ελάχιστα κινούμενα μέρη. Χρησιμοποιεί το OpenSSH για τη σύνδεση (με επιπλέον μέσα σύνδεσης και εναλλακτικούς τρόπους pull ως εναλλακτικές λύσεις) και χρησιμοποιεί μια γλώσσα κατανοητή από τους ανθρώπους(yaml – yet another markup language), η οποία είναι σχεδιασμένη για γρήγορη εκκίνηση προγραμματισμού και χωρίς την ανάγκη για πολλή εκπαίδευση.

Η χρήση του θα μας βοηθήσει στην παραμετροποίηση των συστημάτων καθώς και στην εγκατάσταση, ρύθμιση και διασύνδεση των τεχνολογιών που θα χρησιμοποιήσουμε στην συνέχεια.



Εικόνα 3. Τρόπος λειτουργίας ansible

Πηγή: https://www.redhat.com/rhdc/managed-files/network-automation-comparison.png

- Progress Chef

Το Progress Chef[9] είναι ένα εργαλείο διαχείρισης διαμόρφωσης, γραμμένο σε Ruby και Erlang. Χρησιμοποιεί μια γλώσσα ειδικού τομέα (DSL) βασισμένη εξ ολοκλήρου στη Ruby για τη συγγραφή "συνταγών" διαμόρφωσης συστημάτων. Το Chef χρησιμοποιείται για την απλοποίηση της διαδικασίας διαμόρφωσης και συντήρησης των διακομιστών μιας εταιρείας, ενώ μπορεί να ενσωματωθεί με πλατφόρμες που βασίζονται στο cloud, όπως οι Amazon EC2, Google Cloud Platform, Oracle Cloud, OpenStack, IBM Cloud, Microsoft Azure και Rackspace, για την αυτόματη προμήθεια και διαμόρφωση νέων μηχανών. Το Chef προσφέρει λύσεις για συστήματα τόσο μικρής όσο και μεγάλης κλίμακας.

- Puppet

Το Puppet[10] είναι ένα εργαλείο διαχείρισης διαμόρφωσης λογισμικού που αναπτύχθηκε από την Puppet Inc., η οποία ανήκει στην Perforce, που με τη σειρά της ελέγχεται από ιδιωτικά επενδυτικά κεφάλαια. Το Puppet χρησιμοποιείται για τη διαχείριση των σταδίων του κύκλου ζωής της υποδομής πληροφορικής.

Το Puppet υιοθετεί ένα μοντέλο open-core: η δωρεάν έκδοσή του κυκλοφόρησε υπό την έκδοση 2 της άδειας GNU General Public License (GPL) έως την έκδοση 2.7.0· οι μεταγενέστερες εκδόσεις χρησιμοποιούν την άδεια Apache License, ενώ το Puppet Enterprise διαθέτει ιδιόκτητη άδεια.

Το Puppet και το Puppet Enterprise λειτουργούν σε διάφορα Unix-like συστήματα (όπως Linux, Solaris, BSD, Mac OS X, AIX, HP-UX) και υποστηρίζουν επίσης Microsoft Windows. Το ίδιο το Puppet είναι γραμμένο σε Ruby. Η βιβλιοθήκη Facter, που προσφέρει διαλειτουργική ανάλυση συστήματος, είναι γραμμένη σε C++, ενώ τα Puppet Server και Puppet DB είναι γραμμένα σε Clojure.

Οι κύριες διαφορές μεταξύ Ansible, Puppet και Chef εντοπίζονται στην αρχιτεκτονική, τη ρύθμιση και τη χρήση τους:

1. **Αρχιτεκτονική και Μοντέλο Agent**:

- **Ansible**:

    - **Χωρίς agent**: Δεν απαιτεί την εγκατάσταση agents στους διαχειριζόμενους κόμβους, χρησιμοποιεί **SSH** για επικοινωνία.
    - Εύκολο στη ρύθμιση και συντήρηση, καθώς απαιτεί μόνο έναν κόμβο ελέγχου με εγκατεστημένο το Ansible.
- **Puppet**:
    - **Με agent**: Απαιτεί την εγκατάσταση του Puppet agent σε κάθε διαχειριζόμενο κόμβο.
    - Χρησιμοποιεί αρχιτεκτονική master-agent, όπου ο master server επικοινωνεί με τους agents.
- **Chef**:
    - **Με agent**: Απαιτεί τον Chef client σε κάθε διαχειριζόμενο κόμβο.
    - Ακολουθεί μοντέλο client-server, όπου ο Chef server αλληλεπιδρά με τους client nodes.

2. **Ευκολία Χρήσης και Καμπύλη Μάθησης**:

- **Ansible**:
    - Εύκολο στη μάθηση λόγω των YAML-based playbooks και της απλής ρύθμισης.
    - Ιδανικό για χρήστες με περιορισμένη εμπειρία στον προγραμματισμό.
- **Puppet**:
    - Χρησιμοποιεί τη δική του δηλωτική γλώσσα (Puppet DSL), η οποία έχει μεγαλύτερη καμπύλη μάθησης.
- **Chef**:
    - Οι συνταγές (recipes) και τα cookbooks γράφονται σε Ruby, που είναι πιο πολύπλοκο για όσους δεν γνωρίζουν προγραμματισμό. \

3. **Μοντέλο Εκτέλεσης**:

- **Ansible**:
    - **Push-based**: Ο κόμβος ελέγχου στέλνει τις ρυθμίσεις στους διαχειριζόμενους κόμβους.
- **Puppet**:
    - **Pull-based**: Οι agents τραβούν περιοδικά τις ρυθμίσεις από τον Puppet master.
- **Chef**:
    - **Pull-based**: Ο Chef client τραβά ρυθμίσεις από τον Chef server.

Πίνακας Συνοψίσεως:

Χαρακτηριστικό	**Ansible**	**Puppet**	**Chef**
Αρχιτεκτονική	Χωρίς agent	Με agent	Με agent
Καμπύλη Μάθησης	Εύκολη (YAML)	Μέτρια (DSL)	Δύσκολη (Ruby)
Μοντέλο Εκτέλεσης	Push-based	Pull-based	Pull-based
Ρύθμιση	Απλή	Μέτρια	Πολύπλοκη
Χρήση	Μικρές, δυναμικές υποδομές	Μεγάλες, συμμορφούμενες	Προσαρμόσιμες λύσεις

2.2 Διαχείριση υποδομής (Infrastructure Management)

Το Infrastructure as Code (IaC) είναι η διαχείριση και η παροχή υποδομών μέσω κώδικα αντί για χειροκίνητες διαδικασίες.

Με το IaC δημιουργούνται αρχεία διαμόρφωσης που περιέχουν τις προδιαγραφές της υποδομής, διευκολύνοντας την επεξεργασία και τη διανομή των διαμορφώσεων. Παράλληλα, διασφαλίζεται ότι παρέχεται το ίδιο περιβάλλον κάθε φορά. Η κωδικοποίηση και τεκμηρίωση των προδιαγραφών διαμόρφωσης μέσω IaC υποστηρίζει τη διαχείριση διαμόρφωσης και βοηθά στην αποφυγή αδόμητων και μη τεκμηριωμένων αλλαγών.

Ο έλεγχος εκδόσεων (version control) αποτελεί σημαντικό μέρος του IaC και τα αρχεία διαμόρφωσης θα πρέπει να βρίσκονται υπό έλεγχο πηγαίου κώδικα, όπως κάθε άλλο αρχείο πηγαίου κώδικα λογισμικού. Η ανάπτυξη υποδομών ως κώδικα επιτρέπει επίσης τη διαίρεση της υποδομής σε αρθρωτά στοιχεία, τα οποία μπορούν να συνδυαστούν με διαφορετικούς τρόπους μέσω αυτοματισμού.

Η αυτοματοποίηση της παροχής υποδομών με IaC σημαίνει ότι οι προγραμματιστές δεν χρειάζεται να παρέχουν και να διαχειρίζονται χειροκίνητα διακομιστές, λειτουργικά συστήματα, αποθηκευτικά μέσα και άλλα στοιχεία υποδομής κάθε φορά που αναπτύσσουν ή αναπτύσσουν μια εφαρμογή.

Υπάρχουν δύο προσεγγίσεις για το Infrastructure as Code (IaC): η δηλωτική (declarative) και η επιτακτική (imperative).

Στην δηλωτική προσέγγιση, ορίζεται η επιθυμητή κατάσταση του συστήματος, συμπεριλαμβανομένων των πόρων που χρειάζεστε και των ιδιοτήτων που πρέπει να έχουν. Ένα εργαλείο IaC αναλαμβάνει να διαμορφώσει το σύστημα σύμφωνα με αυτές τις προδιαγραφές. Η δηλωτική προσέγγιση διατηρεί επίσης μια λίστα με την τρέχουσα κατάσταση των αντικειμένων του συστήματος, διευκολύνοντας τη διαχείριση της απεγκατάστασης της υποδομής.

Στην επιτακτική προσέγγιση, ορίζονται οι συγκεκριμένες εντολές που χρειάζονται για την επίτευξη της επιθυμητής διαμόρφωσης, οι οποίες πρέπει να εκτελούνται με τη σωστή σειρά.

- Terraform

Το HashiCorp Terraform [11] είναι ένα εργαλείο υποδομής ως κώδικα που σου επιτρέπει να ορίζεις τόσο τους πόρους στο cloud όσο και σε τοπικές υποδομές (on-prem) μέσω αρχείων διαμόρφωσης κατανοητών από τον άνθρωπο, τα οποία μπορείς να ελέγχεις σε εκδόσεις, να επαναχρησιμοποιείς και να μοιράζεσαι. Στη συνέχεια, μπορείς να χρησιμοποιήσεις ένα συνεπές workflow για να προμηθεύσεις και να διαχειρίζεσαι όλες τις υποδομές σου καθ' όλη τη διάρκεια του κύκλου ζωής τους. Το Terraform μπορεί να διαχειριστεί στοιχεία χαμηλού επιπέδου όπως υπολογιστικοί πόροι, μέσα αποθήκευσης και δικτυακοί πόροι, καθώς και στοιχεία υψηλού επιπέδου όπως εγγραφές DNS και χαρακτηριστικά SaaS.

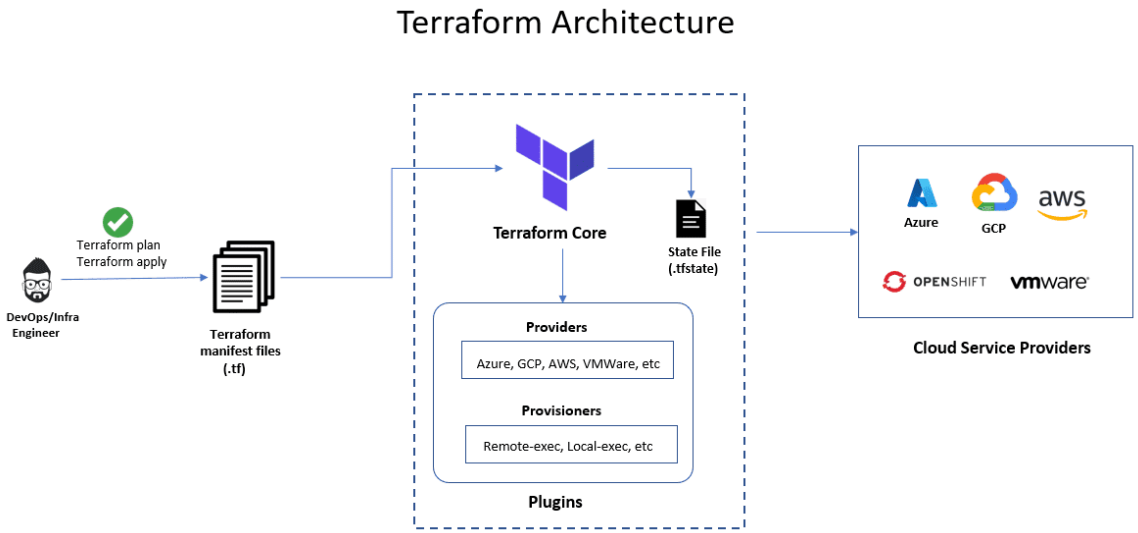
Το Terraform δημιουργεί και διαχειρίζεται πόρους σε πλατφόρμες cloud και άλλες υπηρεσίες με modules[12] και σε συνεργασία με τις διεπαφές προγραμματισμού εφαρμογών (APIs) τους. Οι πάροχοι επιτρέπουν στο Terraform να λειτουργεί με σχεδόν οποιαδήποτε πλατφόρμα ή υπηρεσία με προσβάσιμο API.

Το βασικό workflow του Terraform αποτελείται από τρία στάδια:

- Σχεδιασμός - Γράψιμο: Ορίζεις πόρους, οι οποίοι μπορεί να βρίσκονται σε πολλαπλούς παρόχους cloud και υπηρεσίες. Για παράδειγμα, μπορείς να δημιουργήσεις μια διαμόρφωση για την ανάπτυξη μιας εφαρμογής σε εικονικές μηχανές μέσα σε ένα δίκτυο Virtual Private Cloud (VPC) με ομάδες ασφαλείας και έναν Load Balancer.
- Πλάνο: Το Terraform δημιουργεί ένα σχέδιο εκτέλεσης που περιγράφει την υποδομή που θα δημιουργήσει, θα ενημερώσει, ή θα καταστρέψει βάσει της υπάρχουσας υποδομής και της διαμόρφωσής που έχει γίνει.
- Εφαρμογή: Με την έγκριση του χρήστη, το Terraform εκτελεί τις προτεινόμενες λειτουργίες με τη σωστή σειρά, σεβόμενο τυχόν εξαρτήσεις πόρων. Για παράδειγμα, αν ενημερώσεις τις ιδιότητες ενός VPC και αλλάξεις τον αριθμό των εικονικών μηχανών σε

αυτό το VPC, το Terraform θα ξαναδημιουργήσει το VPC πριν αλλάξει το μέγεθος των εικονικών μηχανών.

Η χρήση του Terraform θα μας βοηθήσει στο να κάνουμε πιο γρήγορη την δημιουργία της υποδομής μας, δηλαδή να φτιάξουμε εικονικά μηχανήματα, δίκτυα μεταξύ τους αλλά και να περιορίσουμε όπου είναι απαραίτητο την πρόσβαση στα συστήματα για λόγους ασφαλείας.



Εικόνα 4. Τρόπος λειτουργίας terraform

Πηγή: https://spacelift.io/_next/image?url=https%3A%2F%2Fspaceliftio.wpcomstaging.com%2Fwp-content%2Fuploads%2F2023%2F03%2Fterraform-architecture-diagram.png&w=3840&q=75

- Pulumi

Το Pulumi[13] είναι μια σύγχρονη πλατφόρμα Infrastructure as Code (IaC) και διαχείρισης μυστικών (secrets management), που επιτρέπει να γίνετε χρήση από γνωστές γλώσσες προγραμματισμού και εργαλεία για την αυτοματοποίηση, ασφάλεια και διαχείρηση ό,τι υποδομής εκτελείτε στο cloud.

Οι κύριες διαφορές μεταξύ Terraform και Pulumi έγκεινται στην προσέγγισή τους στο Infrastructure as Code (IaC), στις υποστηριζόμενες γλώσσες και στις ροές εργασίας τους.

1. **Υποστήριξη Γλωσσών**:

- **Terraform**:
  - Χρησιμοποιεί τη HashiCorp Configuration Language (HCL), μια δηλωτική γλώσσα ειδικά σχεδιασμένη για το IaC.
  - Υποστηρίζει επίσης JSON ως εναλλακτική.
- **Pulumi**:
  - Υποστηρίζει γλώσσες γενικού σκοπού όπως Python, JavaScript, TypeScript, Go, C#, κ.ά.
  - Επιτρέπει στους προγραμματιστές να χρησιμοποιούν γνωστές προγραμματιστικές δομές όπως βρόχους, συνθήκες και συναρτήσεις.

2. **Μοντέλο Εκτέλεσης**:

- **Terraform**:

    - **Δηλωτικό**: Περιγράφει την επιθυμητή τελική κατάσταση, και το Terraform αποφασίζει πώς να την επιτύχει.
    - Εστιάζει στο "τι" πρέπει να γίνει, παρά στο "πώς".
- **Pulumi**:
    - **Υβριδικό Δηλωτικό και Επιτακτικό**: Επιτρέπει τόσο τη δήλωση πόρων όσο και τη χρήση λογικής προγραμματισμού για τον έλεγχο της παροχής υποδομής.
    - Οι προγραμματιστές μπορούν να καθορίσουν άμεσα το "πώς" δημιουργούνται ή διαχειρίζονται οι πόροι.

3. **Διαχείριση Κατάστασης (State Management)**:

- **Terraform**:
    - Διαχειρίζεται αρχεία κατάστασης (π.χ., `terraform.tfstate`) για την παρακολούθηση της υποδομής.
    - Υποστηρίζει τοπική ή απομακρυσμένη αποθήκευση (π.χ., S3, GCS, Terraform Cloud).
- **Pulumi**:
    - Αποθηκεύει την κατάσταση στην backend υπηρεσία του Pulumi (προεπιλογή) ή σε εναλλακτικές όπως S3, Azure Blob Storage, ή αυτοδιαχειριζόμενα backends.
    - Η διαχείριση κατάστασης του Pulumi ενσωματώνεται άψογα σε ροές εργασίας CI/CD.

4. **Κοινότητα και Οικοσύστημα**:

- **Terraform**:
    - Μεγάλη και ώριμη κοινότητα με εκτεταμένη βιβλιοθήκη προκατασκευασμένων modules στο Terraform Registry.
    - Ευρέως αποδεκτό και υποστηριζόμενο από πολλούς παρόχους cloud.
- **Pulumi**:
    - Αναπτυσσόμενη κοινότητα με υποστήριξη για μεγάλο εύρος cloud παρόχων και εργαλείων.
    - Εστιάζει στις σύγχρονες πρακτικές ανάπτυξης και στις cloud-native τεχνολογίες.

Πίνακας Σύνοψης:

Χαρακτηριστικό	**Terraform**	**Pulumi**
Γλώσσα	HCL (δηλωτική)	Γενικού σκοπού (Python, Go, κ.ά.)
Μοντέλο Εκτέλεσης	Δηλωτικό	Επιτακτικό + Δηλωτικό
Διαχείριση Κατάστασης	Τοπικά ή απομακρυσμένα αρχεία	Διαχειριζόμενο ή αυτο-φιλοξενούμενο backend
Καμπύλη Μάθησης	Μέτρια (HCL-specific)	Εύκολη για προγραμματιστές
Κοινότητα	Μεγάλη, ώριμη	Αναπτυσσόμενη
Περιπτώσεις Χρήσης	Ops-focused	Dev-focused

2.3 Τεχνολογίες Virtualization και Containarization

- KVM

Το Kernel-based Virtual Machine (KVM)[14] είναι ένα ανοιχτού κώδικα μοντέλο εικονικοποίησης στον πυρήνα του Linux, που επιτρέπει στον πυρήνα να λειτουργεί ως υπερβατής (hypervisor). Ενσωματώθηκε στον κύριο πυρήνα του Linux στην έκδοση 2.6.20, η οποία κυκλοφόρησε στις 5 Φεβρουαρίου 2007. Το KVM απαιτεί έναν επεξεργαστή με επεκτάσεις υλικού εικονικοποίησης, όπως το Intel VT ή το AMD-V. Το KVM έχει επίσης μεταφερθεί σε άλλα λειτουργικά συστήματα όπως το FreeBSD και το illumos με τη μορφή φορτώσιμων μονάδων πυρήνα.

Το KVM αρχικά σχεδιάστηκε για επεξεργαστές x86 αλλά από τότε έχει μεταφερθεί σε ESA/390, PowerPC, IA-64 και ARM. Η μεταφορά για IA-64 αφαιρέθηκε το 2014.

Το KVM υποστηρίζει εικονικοποίηση με υποστήριξη υλικού για μια ευρεία ποικιλία λειτουργικών συστημάτων επισκέπτη, συμπεριλαμβανομένων των BSD, Solaris, Windows, Haiku, ReactOS, Plan 9, AROS, macOS και ακόμη και άλλων συστημάτων Linux. Επιπλέον, το Android 2.2, GNU/Hurd (Debian K16), Minix 3.1.2a, Solaris 10 U3 και Darwin 8.0.1, μαζί με άλλα λειτουργικά συστήματα και μερικές νεότερες εκδόσεις αυτών που αναφέρονται, είναι γνωστό ότι λειτουργούν με ορισμένους περιορισμούς.

Επιπρόσθετα, το KVM παρέχει υποστήριξη paravirtualization για Linux, OpenBSD, FreeBSD, NetBSD, Plan 9 και Windows επισκέπτες χρησιμοποιώντας το API VirtIO. Αυτό περιλαμβάνει μια paravirtual Ethernet κάρτα, ελεγκτή I/O δίσκου, οδηγό balloon και ένα γραφικό περιβάλλον VGA χρησιμοποιώντας οδηγούς SPICE ή Vmware.

Η χρήση του KVM θα είναι για την δημιουργία εικονικών μηχανημάτων σε τοπικό επίπεδο αρχικά και για τα πλαίσια δοκιμών των υπόλοιπων τεχνολογιών. Έτσι θα μπορέσουμε και να κάνουμε πιο γρήγορα τις δοκιμές μας αλλά και να κρατήσουμε το επίπεδο δυσκολίας σε αρχικό στάδιο χαμηλό.

- VPC & VPS

Ένα εικονικό ιδιωτικό δίκτυο (VPC)[15] είναι ένας διαμορφώσιμος κατά παραγγελία δεξαμενή κοινόχρηστων πόρων εντός ενός δημόσιου περιβάλλοντος cloud, προσφέροντας ένα ορισμένο επίπεδο απομόνωσης μεταξύ των διαφορετικών οργανισμών (που στο εξής θα αναφέρονται ως χρήστες) που χρησιμοποιούν τους πόρους. Η απομόνωση μεταξύ ενός χρήστη VPC και όλων των άλλων χρηστών του ίδιου cloud (άλλων χρηστών VPC καθώς και άλλων χρηστών δημόσιου cloud) επιτυγχάνεται συνήθως μέσω της ανάθεσης ενός ιδιωτικού υποδικτύου IP και ενός εικονικού κατασκευασμάτος επικοινωνίας (όπως ένα VLAN ή ένα σύνολο κρυπτογραφημένων καναλιών επικοινωνίας) ανά χρήστη. Σε ένα VPC, ο προηγουμένως περιγραφείς μηχανισμός, που παρέχει απομόνωση εντός του cloud, συνοδεύεται από μια λειτουργία εικονικού ιδιωτικού δικτύου (VPN) (και πάλι, ανατεθειμένη ανά χρήστη VPC) που εξασφαλίζει, μέσω της ταυτοποίησης και κρυπτογράφησης, την απομακρυσμένη πρόσβαση του οργανισμού στους πόρους του VPC. Με την εισαγωγή των περιγραφόμενων επιπέδων απομόνωσης, ένας οργανισμός που χρησιμοποιεί αυτή την υπηρεσία λειτουργεί στην πράξη σε ένα 'εικονικά ιδιωτικό' cloud (δηλαδή, σαν να μην μοιράζεται την υποδομή cloud με άλλους χρήστες), και εξού γι' αυτό ονομάζεται VPC.

Το VPC χρησιμοποιείται πιο συχνά στο πλαίσιο της υποδομής ως υπηρεσία cloud. Σε αυτό το πλαίσιο, ο πάροχος της υποδομής, που παρέχει την υποκείμενη δημόσια υποδομή cloud, και ο πάροχος που υλοποιεί την υπηρεσία VPC πάνω σε αυτή την υποδομή, μπορεί να είναι διαφορετικοί προμηθευτές.

Ένας εικονικός ιδιωτικός διακομιστής (VPS)[16] είναι μια εικονική μηχανή που πωλείται ως υπηρεσία από μια υπηρεσία φιλοξενίας στο Διαδίκτυο. Ο όρος "εικονικός αφιερωμένος διακομιστής" (VDS) έχει επίσης παρόμοιο νόημα.

Ένας εικονικός ιδιωτικός διακομιστής τρέχει το δικό του αντίγραφο ενός λειτουργικού συστήματος (OS), και οι πελάτες μπορούν να έχουν πρόσβαση επιπέδου superuser σε αυτήν την περίπτωση του λειτουργικού συστήματος, έτσι ώστε να μπορούν να εγκαταστήσουν σχεδόν οποιοδήποτε λογισμικό λειτουργεί σε αυτό το OS. Για πολλούς σκοπούς, είναι λειτουργικά ισοδύναμο με έναν αφιερωμένο φυσικό διακομιστή και, καθώς είναι ορισμένο από λογισμικό, μπορεί να δημιουργηθεί και να διαμορφωθεί πιο εύκολα. Ένας εικονικός διακομιστής κοστίζει λιγότερο από έναν αντίστοιχο φυσικό διακομιστή. Ωστόσο, καθώς οι εικονικοί διακομιστές μοιράζονται το υποκείμενο φυσικό υλικό με άλλα VPS, η απόδοση μπορεί να είναι χαμηλότερη ανάλογα με το φόρτο εργασίας οποιωνδήποτε άλλων εκτελούμενων εικονικών μηχανών.

Η χρήση ενός VPS ή VPC θα γίνει στα πλαίσια του πρώτου θέματος και θα επεκταθεί στο δεύτερο. Πιο συγκεκριμένα θα δούμε πως μπορούμε από την αρχιτεκτονική ενός μηχανήματος στο οποίο τρέχουν όλες οι υπηρεσίες μας να διαχωρίσουμε την εφαρμογή από το WAF ή πιθανώς και άλλα στοιχεία τα οποία χρησιμοποιεί, όπως για παράδειγμα την βάση.

- Containers – Docker / Podman & Docker / Podman compose

Τα containers[17] είναι τεχνολογίες που επιτρέπουν τη συσκευασία και την απομόνωση εφαρμογών μαζί με όλο το περιβάλλον εκτέλεσής τους – όλα τα αρχεία που είναι απαραίτητα για τη λειτουργία τους. Αυτό καθιστά εύκολη τη μεταφορά της εφαρμογής που περιέχεται σε διαφορετικά περιβάλλοντα (όπως ανάπτυξης, δοκιμών, παραγωγής κ.λ.π.) διατηρώντας πλήρως τη λειτουργικότητά της.

Τα containers αποτελούν επίσης σημαντικό μέρος της ασφάλειας πληροφορικής. Ενσωματώνοντας την ασφάλεια στον κύκλο ζωής των containers και προστατεύοντας την υποδομή, τα containers παραμένουν αξιόπιστα, επεκτάσιμα και ασφαλή. Επιπλέον, μπορείτε εύκολα να μετακινήσετε τις εφαρμογές που είναι σε containers μεταξύ δημόσιων, ιδιωτικών και υβριδικών περιβαλλόντων cloud, καθώς και μεταξύ data centers ή τοπικών εγκαταστάσεων (on-premises), με συνεπή συμπεριφορά και λειτουργικότητα.

Για την ευκολότερη διαχείρηση των container θα κάνουμε χρήση των τεχνολογιών docker και podman. Το docker ήταν ο πρώτος διαχειριστής container με αποτέλεσμα να είναι συνδεδεμένο το όνομα docker με την λέξη container. Στην συνέχεια όμως έχουν δημιουργηθεί πολλές άλλες τεχνολογίες διαχείρισης container η κάθε μια με τα δικά της χαρακτηριστικά.

Docker:

Το Docker[18] είναι μια ανοιχτή πλατφόρμα για την ανάπτυξη, αποστολή και εκτέλεση εφαρμογών. Το Docker μας επιτρέπει να διαχωρίζουμε τις εφαρμογές από την υποδομή μας, διευκολύνοντας την ταχύτερη παράδοση λογισμικού.

Με το Docker, μπορούμε να διαχειριστούμε την υποδομή μας με τον ίδιο τρόπο που διαχειριζόμαστε τις εφαρμογές μας. Αξιοποιώντας τις μεθοδολογίες του Docker για την αποστολή, δοκιμή και ανάπτυξη κώδικα, μπορούμε να μειώσουμε σημαντικά την καθυστέρηση μεταξύ της συγγραφής κώδικα και της εκτέλεσής του σε περιβάλλον παραγωγής.

Podman:

Το Podman[19] είναι ένα εργαλείο χωρίς daemon, ανοιχτού κώδικα, που είναι εγγενές στο Linux και έχει σχεδιαστεί για να διευκολύνει την εύρεση, εκτέλεση, δημιουργία, κοινή χρήση και ανάπτυξη εφαρμογών χρησιμοποιώντας Containers και Container Images που συμμορφώνονται με την πρωτοβουλία Open Containers Initiative (OCI).

Το Podman παρέχει ένα περιβάλλον γραμμής εντολών (CLI) που είναι οικείο σε όσους έχουν χρησιμοποιήσει το Docker Container Engine, κάνοντας τη μετάβαση εύκολη και διαισθητική.

Compose:

Για τις δυο παραπάνω τεχνολογίες υπάρχει η υποεντολή compose. Ξεκίνησε ως ξεχωριστό εργαλείο το οποίο στην συνέχεια όμως ενοποιήθηκε με την κύρια εφαρμογή λόγω της χρησιμότητας της αλλά και για να εξασφαλιστεί η καλύτερη δυνατή συμβατότητα με την κύρια εφαρμογή.

Το Docker Compose[20] είναι ένα εργαλείο για τον ορισμό και την εκτέλεση εφαρμογών πολλαπλών κοντέινερ. Είναι το κλειδί για την απλοποιημένη και αποδοτική εμπειρία ανάπτυξης και ανάπτυξης εφαρμογών.

Το Compose[21] απλοποιεί τον έλεγχο ολόκληρης της στοίβας εφαρμογών σας, καθιστώντας εύκολη τη διαχείριση υπηρεσιών, δικτύων και χώρων αποθήκευσης σε ένα ενιαίο, κατανοητό αρχείο διαμόρφωσης YAML. Στη συνέχεια, με μία μόνο εντολή, μπορείτε να δημιουργήσετε και να εκκινήσετε όλες τις υπηρεσίες από το αρχείο διαμόρφωσης.

Το Compose λειτουργεί σε όλα τα περιβάλλοντα: παραγωγή, staging, ανάπτυξη, δοκιμές, καθώς και σε ροές εργασιών CI. Επίσης, διαθέτει εντολές για τη διαχείριση ολόκληρου του κύκλου ζωής της εφαρμογής σας:

- Εκκίνηση, διακοπή και ανακατασκευή υπηρεσιών
- Προβολή της κατάστασης των τρεχουσών υπηρεσιών
- Ροή εξόδου των αρχείων καταγραφής των τρεχουσών υπηρεσιών
- Εκτέλεση μιας εφάπαξ εντολής σε μια υπηρεσία

Ακριβώς με τον ίδιο τρόπο προσπαθεί να λειτουργήσει και το podman compose το οποίο σκοπεύει να είναι drop-in αντικαταστάτης (δηλαδή να αντιστοιχεί μια προς μια τις λειτουργίες τις εφαρμογής) του docker-compose.

Το podman compose είναι ένα ελαφρύ περιτύλιγμα γύρω από έναν εξωτερικό πάροχο compose, όπως το docker-compose ή το podman-compose. Αυτό σημαίνει ότι το podman compose εκτελεί ένα άλλο εργαλείο που υλοποιεί τη λειτουργικότητα compose, αλλά διαμορφώνει το περιβάλλον με τέτοιο τρόπο ώστε ο πάροχος compose να μπορεί να επικοινωνεί διαφανώς με την τοπική σύνδεση (socket) του Podman.

Οι καθορισμένες επιλογές, καθώς και η εντολή και τα ορίσματα, μεταβιβάζονται απευθείας στον πάροχο compose.

- Kubernetes

Το Kubernetes[22] (γνωστό και ως K8s) είναι ένα σύστημα ανοιχτού κώδικα για την αυτοματοποίηση της ανάπτυξης, της κλιμάκωσης και της διαχείρισης εφαρμογών που βασίζονται σε κοντέινερ. Ομαδοποιεί τα κοντέινερ που αποτελούν μια εφαρμογή σε λογικές μονάδες για εύκολη διαχείριση και ανακάλυψη.

Χαρακτηριστικά του Kubernetes:

1. **Αυτοματοποιημένες Αναπτύξεις και Επαναφορές (Rollouts and Rollbacks)**
   Το Kubernetes προοδευτικά εφαρμόζει αλλαγές στην εφαρμογή ή στη διαμόρφωσή της, παρακολουθώντας την υγεία της εφαρμογής για να διασφαλίσει ότι δεν θα σταματήσουν όλες οι περιπτώσεις (instances) ταυτόχρονα. Αν κάτι πάει στραβά, επαναφέρει την αλλαγή αυτόματα. Επωφεληθείτε από ένα συνεχώς αναπτυσσόμενο οικοσύστημα λύσεων ανάπτυξης.

2. **Ανακάλυψη Υπηρεσιών και Εξισορρόπηση Φορτίου (Service Discovery and Load Balancing)**
   Δεν χρειάζεται να τροποποιήσετε την εφαρμογή σας για να χρησιμοποιήσετε μηχανισμούς ανακάλυψης υπηρεσιών. Το Kubernetes παρέχει στα Pods μοναδικές IP διευθύνσεις και ένα DNS όνομα για ένα σύνολο Pods, επιτρέποντας την εξισορρόπηση φορτίου μεταξύ τους.

3. **Ορχήστρωση Αποθήκευσης (Storage Orchestration)**
   Το Kubernetes μπορεί να προσαρτήσει αυτόματα το σύστημα αποθήκευσης της επιλογής σας, είτε πρόκειται για τοπική αποθήκευση, είτε για δημόσιο cloud provider, είτε για δίκτυο αποθήκευσης όπως iSCSI ή NFS.

4. **Αυτο-ίαση (Self-Healing)**
   Επανεκκινεί κοντέινερ που αποτυγχάνουν, αντικαθιστά και επαναπρογραμματίζει κοντέινερ όταν οι κόμβοι αποτυγχάνουν, σταματά κοντέινερ που δεν ανταποκρίνονται στις

προκαθορισμένες δοκιμές υγείας και δεν τα διαφημίζει στους πελάτες μέχρι να είναι έτοιμα να εξυπηρετήσουν.

5. **Διαχείριση Μυστικών και Διαμόρφωσης (Secret and Configuration Management)**
   Αναπτύξτε και ενημερώστε μυστικά και τη διαμόρφωση της εφαρμογής σας χωρίς να χρειάζεται να αναδημιουργήσετε το image σας και χωρίς να εκθέτετε τα μυστικά στη διαμόρφωση του stack.
6. **Αυτόματη Δέσμευση Πόρων (Automatic Bin Packing)**
   Το Kubernetes τοποθετεί αυτόματα κοντέινερ με βάση τις απαιτήσεις πόρων και άλλους περιορισμούς, χωρίς να θυσιάζει τη διαθεσιμότητα. Μπορείτε να συνδυάσετε κρίσιμα και μη κρίσιμα workloads για καλύτερη αξιοποίηση των πόρων.
7. **Εκτέλεση Παρτίδων (Batch Execution)**
   Εκτός από τις υπηρεσίες, το Kubernetes μπορεί να διαχειριστεί παρτίδες και εργασίες συνεχιζόμενης ενσωμάτωσης (CI), αντικαθιστώντας κοντέινερ που αποτυγχάνουν, εφόσον είναι επιθυμητό.
8. **Οριζόντια Κλιμάκωση (Horizontal Scaling)**
   Αυξήστε ή μειώστε την κλίμακα της εφαρμογής σας με μία απλή εντολή, μέσω ενός UI, ή αυτόματα, βάσει της χρήσης της CPU.
9. **Διπλή Στοίβα IPv4/IPv6 (IPv4/IPv6 Dual-Stack)**
   Δυνατότητα κατανομής διευθύνσεων IPv4 και IPv6 σε Pods και Υπηρεσίες.
10. **Σχεδιασμένο για Επεκτασιμότητα (Designed for Extensibility)**
    Προσθέστε χαρακτηριστικά στο Kubernetes cluster σας χωρίς να χρειάζεται να τροποποιήσετε τον αρχικό κώδικα.

Οι βασικές διαφορές μεταξύ Virtual Machines (VMs) και Containers είναι οι εξής:

1. **Αρχιτεκτονική**

- **VMs**: Κάθε VM εκτελεί ένα πλήρες λειτουργικό σύστημα (OS) και βασίζεται σε έναν hypervisor για τη διαχείριση των πόρων.
  - Περιλαμβάνουν kernel, drivers και libraries.
  - Είναι βαρύτερα λόγω της πλήρους προσομοίωσης OS.
- **Containers**: Μοιράζονται τον πυρήνα του host OS και εκτελούν εφαρμογές μεμονωμένα σε απομονωμένα περιβάλλοντα.
  - Ελαφρύτερα, χωρίς πλήρες OS, μόνο τις απαραίτητες βιβλιοθήκες και αρχεία.

2. **Ταχύτητα εκκίνησης**

- **VMs**: Αργή εκκίνηση, αφού χρειάζεται να φορτώσει ολόκληρο το OS.
- **Containers**: Πολύ γρήγορη εκκίνηση, καθώς χρησιμοποιούν τον ήδη φορτωμένο πυρήνα του host OS.

3. Αποδοτικότητα πόρων

- **VMs**: Καταναλώνουν περισσότερους πόρους (CPU, RAM) λόγω του πλήρους OS και των virtualized hardware.
- **Containers**: Ελαφρύτερα, χρησιμοποιούν λιγότερους πόρους αφού μοιράζονται το OS.

4. **Φορητότητα**

- **VMs**: Δυσκολότερη μεταφορά, καθώς είναι δεμένες με τον hypervisor και μεγαλύτερα σε μέγεθος.
- **Containers**: Πολύ πιο φορητά και ανεξάρτητα από το υποκείμενο hardware, αφού περιλαμβάνουν μόνο το runtime και τις εφαρμογές.

5. **Χρήση και παραδείγματα**

- **VMs**: Κατάλληλα για πολυπλοκότερα περιβάλλοντα που απαιτούν πλήρη απομόνωση ή διαφορετικά OS (π.χ. Windows πάνω σε Linux).
- **Containers**: Ιδανικά για microservices, DevOps, και συνεχόμενη ανάπτυξη και παράδοση (CI/CD).

Συνοπτικά, οι VMs προσφέρουν ισχυρότερη απομόνωση αλλά είναι πιο βαριές, ενώ οι Containers είναι ελαφρύτεροι, ταχύτεροι και πιο κατάλληλοι για μοντέρνες αρχιτεκτονικές cloud.

Με τις παραπάνω τεχνολογίες να εχουν αναλυθεί θα γίνει επιλογή του docker / podman και του compose καθώς επιτρέπει την δημιουργία της υποδομής πιο γρήγορα. Στην περίπτωση που οι ανάγκες της υποδομής ήταν μεγαλύτερες, όπως για παράδειγμα ένα σενάριο με πολλές ομάδες ή μια πραγματική εφαρμογή μαζί με ένα WAF όπου η επισκεψιμότητα είναι πολύ μεγαλύτερη τότε θα γινόταν επιλογή K8s για να γίνει χρήση της ιδιότητας του Scaling που προσφέρει.

3. Σενάρια υποδομών, σχεδιασμός και αυτοματοποίηση

Στο κεφάλαιο αυτό θα εξετασθεί το κομμάτι της υποδομής και της αυτοματοποίησης της. Αρχικά θα γίνει μια έρευνα γύρω από πιθανά σενάρια υποδομών τα οποία συνήθως συναντώνται σε επαγγελματικά σενάρια όπου εταιρίες έχουν εφαρμογές τις οποίες θέλουν να προστατέψουν από επιθέσεις. Ένας από τους σκοπούς της εργασίας είναι να μελετηθούν αυτές οι υποδομές και ο τρόπος που μπορεί κάποιος να φτάσει σε αυτές πιο εύκολα.

Η διαδικασία της αυτοματοποίησης στην εργασία μπορεί να χωριστεί σε κάποια επιμέρους κομμάτια. Αρχικά την υλοποίηση της υποδομής, δηλαδή τα εικονικά μηχανήματα και τα firewall στον πάροχο VPS αλλά και λοιπές δικτυακές ανάγκες που μπορεί να υπάρχουν για αυτά. Στην συνέχεια θα γίνουν ρυθμίσεις πάνω στο λειτουργικό/α συστήματα, όπως δημιουργία χρηστών, εγκατάσταση εφαρμογών και ρύθμιση τους και τέλος θα πρέπει να αυτοματοποιηθεί και η εγκατάσταση των ευπαθών εφαρμογών καθώς και των WAF τα οποία θα μελετηθούν στις επιθέσεις που θα εκτελεστούν.

3.1 Σενάρια υποδομών για Web applications

Στην ενότητα αυτή θα μελετηθούν πιθανές υποδομές οι οποίες συναντώνται σε επαγγελματικά πλαίσια και θα μπορούσαν να ακολουθηθούν σε σενάρια για web εφαρμογές.

Η εγκατάσταση μιας εφαρμογής μαζί με ένα Web Application Firewall (WAF) μπορεί να γίνει με διάφορους τρόπους, ανάλογα με τις ανάγκες ασφαλείας, την αρχιτεκτονική της εφαρμογής και τις υποδομές που είναι διαθέσιμες. Παρακάτω θα δούμε μερικά παραδείγματα:

- Όλα σε έναν server

Σε αυτό το σενάριο, η εφαρμογή, ο web server, και το WAF είναι όλα εγκατεστημένα σε έναν μόνο server.

Πλεονεκτήματα

- Απλότητα στη διαχείριση και την ανάπτυξη.

- Μειωμένο κόστος σε σχέση με τη χρήση πολλών servers.

Μειονεκτήματα

- Περιορισμένη επεκτασιμότητα και απόδοση.

- Αυξημένος κίνδυνος για ένα σημείο αποτυχίας.

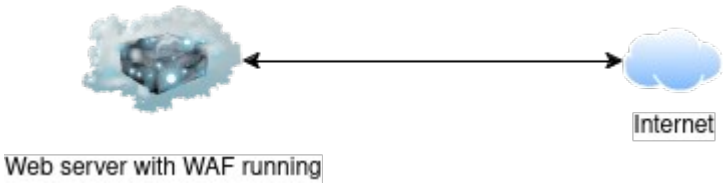
Παράδειγμα

- Server: Ένας φυσικός ή εικονικός server.

- Λογισμικό: Apache ή Nginx για web server, ModSecurity ως WAF (για Apache), ή NAXSI (για Nginx).

- Διάταξη:

  - Αίτηση χρήστη → WAF (ModSecurity/NAXSI) → Web Server (Apache/Nginx) → Εφαρμογή.



Εικόνα 5. Web server με WAF στον ίδιο server

- WAF σε ξεχωριστό Server ως Reverse Proxy

Σε αυτό το σενάριο, το WAF εγκαθίσταται σε έναν ξεχωριστό server και λειτουργεί ως reverse proxy για την προστασία της εφαρμογής.

Πλεονεκτήματα

- Καλύτερη απομόνωση ασφαλείας.

- Ευκολότερη διαχείριση του WAF χωρίς να επηρεάζεται η εφαρμογή.

Μειονεκτήματα

- Απαιτείται περισσότερος εξοπλισμός ή περισσότερα VM.

- Αυξημένη πολυπλοκότητα στην αρχιτεκτονική.

Παράδειγμα

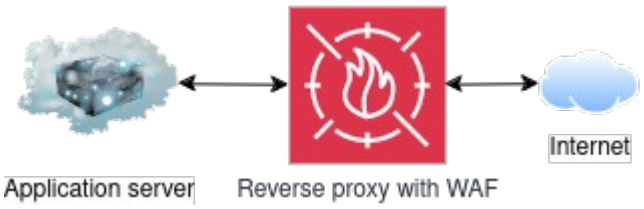
- Server 1 (WAF/Reverse Proxy): Ένας φυσικός ή εικονικός server.

  - Λογισμικό: Nginx ή HAProxy με WAF (όπως ModSecurity ή NAXSI).

  - Διάταξη: Αίτηση χρήστη → WAF/Reverse Proxy → Web Server της εφαρμογής.

- Server 2 (Web Server): Ένας φυσικός ή εικονικός server που φιλοξενεί την εφαρμογή.

  - Λογισμικό: Apache, Nginx ή άλλο web server.



Εικόνα 6. WAF με λειτουργία reverse proxy

- WAF ως Υπηρεσία (WAFaaS)

Σε αυτό το σενάριο, χρησιμοποιείται μια υπηρεσία WAF που προσφέρεται από κάποιον τρίτο πάροχο (π.χ. Cloudflare, AWS WAF).

Πλεονεκτήματα

- Μείωση της ανάγκης για διαχείριση υποδομών.

- Συνεχής ενημέρωση και προστασία από νέες απειλές από τον πάροχο.

Μειονεκτήματα

- Εξάρτηση από τον τρίτο πάροχο.

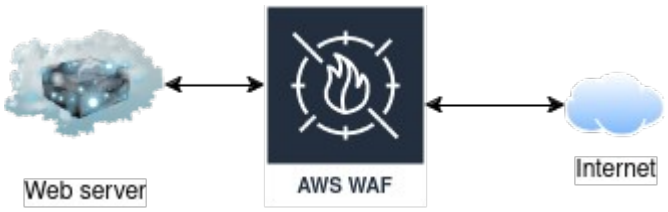
- Μπορεί να υπάρξει καθυστέρηση λόγω του επιπρόσθετου βήματος στον χειρισμό των αιτήσεων.

Παράδειγμα

- WAF ως υπηρεσία: Cloudflare WAF ή AWS WAF.

- Διάταξη: Αίτηση χρήστη → WAFaaS (Cloudflare/AWS) → Web Server της εφαρμογής.

- Server (Web Server): Ένας φυσικός ή εικονικός server που φιλοξενεί την εφαρμογή.



Εικόνα 7. WAF as a service από το aws

- WAF με Load Balancer

Σε αυτό το σενάριο, το WAF εγκαθίσταται μαζί με έναν load balancer που διανέμει την κυκλοφορία σε πολλούς web servers.

Πλεονεκτήματα

- Υψηλή επεκτασιμότητα και απόδοση.

- Καλύτερη διαχείριση της κυκλοφορίας και ανοχή σε αποτυχίες.

Μειονεκτήματα

- Αυξημένο κόστος και πολυπλοκότητα.

- Απαιτεί καλή γνώση διαχείρισης δικτύων και υποδομών.

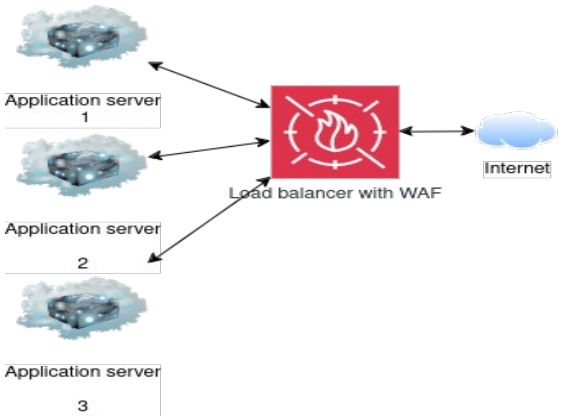
Παράδειγμα

- Server 1 (WAF/Load Balancer): Ένας φυσικός ή εικονικός server.

- Λογισμικό: Nginx ή HAProxy με WAF (όπως ModSecurity ή NAXSI).

- Διάταξη: Αίτηση χρήστη → WAF/Load Balancer → Web Servers της εφαρμογής.

- Servers 2+ (Web Servers): Πολλοί φυσικοί ή εικονικοί servers που φιλοξενούν την εφαρμογή



Εικόνα 8. WAF μαζί με load balancer

Κάθε σενάριο έχει τα δικά του πλεονεκτήματα και μειονεκτήματα και η επιλογή του κατάλληλου εξαρτάται από τις συγκεκριμένες ανάγκες της εφαρμογής και των χρηστών της

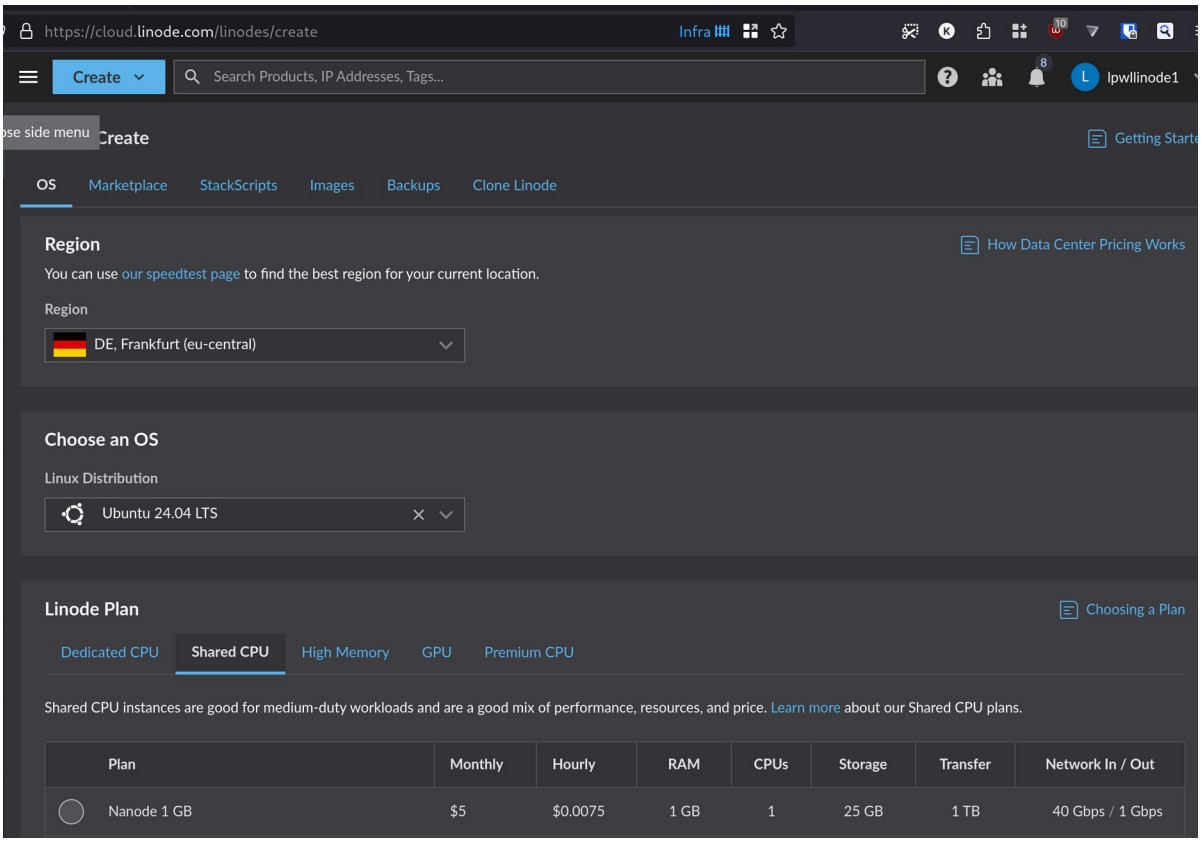
3.2 Υλοποίηση και αυτοματοποίηση αρχιτεκτονικής

Στο κεφάλαιο αυτό θα μελετηθεί η αρχική υποδομή και ο σχεδιασμός της, δηλαδή η δημιουργία των μηχανημάτων και των firewall, τα βήματα τα οποία θα ακολουθηθούν για την αυτοματοποίηση του σεναρίου και την εγκατάσταση της εφαρμογής και του WaF με την βοήθεια των οποίων θα γίνουν οι επιθέσεις.

- Δημιουργία Linux VM στον πάροχο Linode

Το πρώτο βήμα είναι να δημιουργηθεί ένα εικονικό μηχάνημα (VM) στην σελίδα του παρόχου. Υπάρχουν επιλογές όπως το είδος του μηχανήματος με τα χαρακτηριστικά με τα οποία μπορεί να παραμετροποιηθεί (επεξεργαστή, μνήμη, δίσκο κλπ.). Επιπλέον υπάρχει η δυνατότητα για επιλογή λειτουργικού συστήματος από διάφορες προεπιλογές (π.χ Ubuntu, Fedora) αλλά και σε μερικές περιπτώσεις η δυνατότητα να ανέβει μια εικόνα ISO από κάποιο άλλο λειτουργικό σύστημα και να γίνει χειροκίνητα η εγκατάσταση.

Η διαφορά μεταξύ των δύο επιλογών είναι ότι στις προεπιλογές ο πάροχος δίνει την δυνατότητα για αυτοματοποιημένες ρυθμίσεις αλλιώς πρέπει να γίνουν οι ρυθμίσεις από τον χρήστη, είτε χειροκίνητα είτε με εργαλεία αυτοματισμού. Επιπλέον δεν δίνεται η δυνατότητα να γίνει χρήση υπηρεσιών του παρόχου όπως Scripts που εγκαθιστούν εφαρμογές χωρίς την παρέμβαση των χρηστών, όπως για παράδειγμα πλατφόρμες ιστοσελίδων όπως wordpress και drupal.

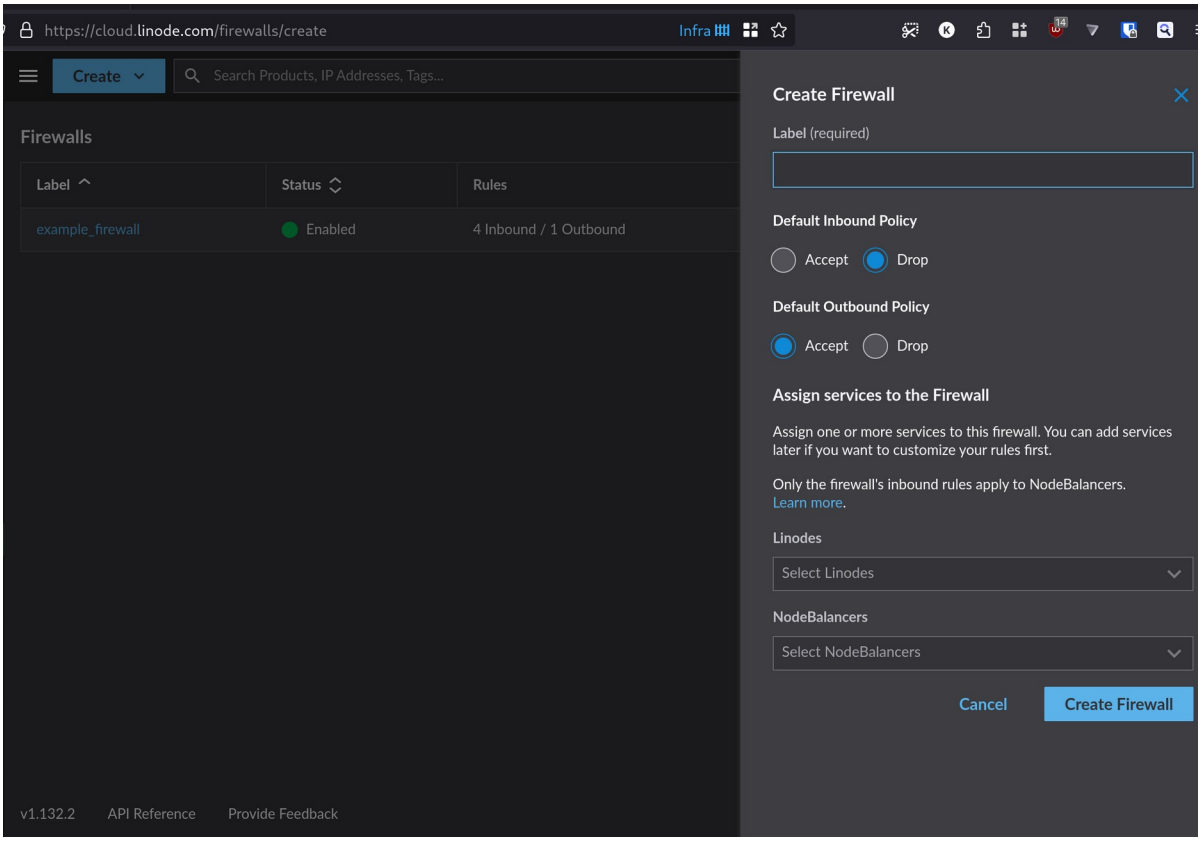


Εικόνα 9. Χαρακτηριστικά μηχανήματος στον πάροχο Linode

Πηγή: https://cloud.linode.com/linodes/create

Στην συνέχεια δίνεται η δυνατότητα για προσθήκη κλειδιών SSH για την ευκολότερη και ασφαλέστερη σύνδεση στο μηχάνημα, δημιουργία firewall καθώς και δικτυακά την τοποθεσία του γεωγραφικά, αν υπάρχει η ανάγκη ένταξης σε ένα VPC ή να είναι μέρος ενός VLAN.

Στην συνέχεια για την εργασία μας γίνει χρήση firewall από τον πάροχο. Όπως φαίνεται και στην εικόνα, μπορεί να γίνει ανάθεση σε ένα ή περισσότερα VM, να βάλουμε default policies όπως και ένα firewall σε σύστημα linux (π.χ iptables). Επιπλέον, δίνετε η επιλογή να μπουν κανόνες για πρωτόκολλα, πόρτες, πηγή ή προορισμό σε διεύθυνση IP και τι δράση θα ακολουθήσει για τα παραπάνω.



Εικόνα 10. Χαρακτηριστικά firewall στον πάροχο Linode

Πηγή: https://cloud.linode.com/firewalls/create

Τα παραπάνω θα αυτοματοποιηθούν με terraform / opentofu. Από την δημιουργία VM με τα χαρακτηριστικά που εξυπηρετούν τις ανάγκες του σεναρίου, τα firewall με τους κανόνες τους και την ανάθεση αυτών στα VM κατά την δημιουργία τους.

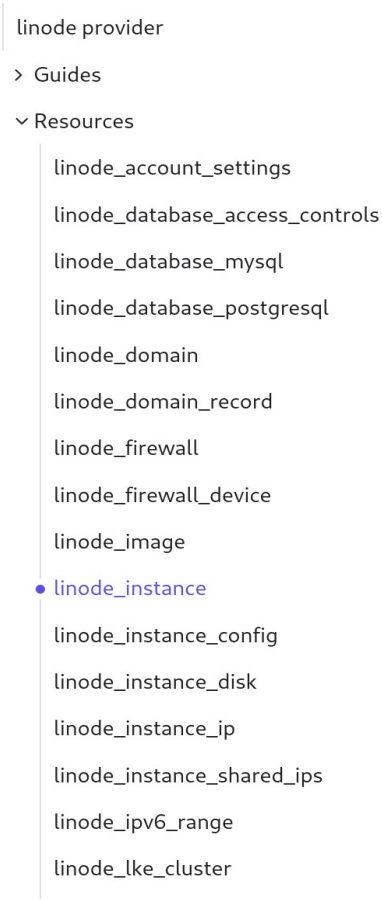
- Αυτοματοποίηση με terraform / opentofu

Το terraform λειτουργεί με παρόχους[23] (providers) για τις υπηρεσίες με τις οποίες χρειάζεται να επικοινωνήσει, όπως SaaS, APIs κλπ.

Κάθε provider προσθέτει έναν σύνολο τύπων πόρων (resource types) και/ή πηγών δεδομένων (data sources) που το Terraform μπορεί να διαχειριστεί. Κάθε τύπος πόρου υλοποιείται από έναν provider· χωρίς providers, το Terraform δεν μπορεί να διαχειριστεί καμία μορφή υποδομής.

Οι περισσότεροι providers ρυθμίζουν μια συγκεκριμένη πλατφόρμα υποδομής (είτε cloud είτε αυτο-φιλοξενούμενη). Επιπλέον, οι providers μπορούν να προσφέρουν τοπικές λειτουργίες, όπως η δημιουργία τυχαίων αριθμών για μοναδικά ονόματα πόρων.

Το πρώτο βήμα λοιπόν που πρέπει να γίνει στον κώδικα είναι να δηλωθεί ένας provider. Στην παρακάτω εικόνα βλέπουμε και τα resources που προσφέρει ο linode provider. Δίνεται η δυνατότητα για δημιουργία από VM και firewall μέχρι βάσεις δεδομένων και managed kubernetes cluster.



Εικόνα 11. Χαρακτηριστικά firewall στον πάροχο Linode

Πηγή: https://registry.terraform.io/providers/linode/linode/latest/docs

Έχοντας αυτές τις πληροφορίες ξεκινάει η υλοποίηση της εργασίας. Στην πρώτη εικόνα παρατηρείται η δήλωση του παρόχου καθώς και μερικές μεταβλητές που θα χρησιμοποιηθούν στην συνέχεια για την διευκόλυνση της διασύνδεσης των πόρων. Κάποιες από αυτές είναι δηλωμένες μέσα στον κώδικα καθώς είναι γνωστά στοιχεία όπως για παράδειγμα οι IP διευθύνσεις για πρόσβαση στο firewall ενώ άλλες είναι δηλωμένες ως μυστικά όπως το token του παρόχου που δεν πρέπει να διαμοιράζεται καθώς δίνει πρόσβαση σε όλη την υποδομή.

Help

main.tf

main.tf

```
# Declare the provider that is used to connect to the cloud
# Webpage https://registry.terraform.io/providers/linode/linode/latest/docs
terraform {
  required_providers {
    linode = {
      source  = "linode/linode"
      version = "~> 1.0"
    }
  }
}

# Declaring variables used when creating instances and / or other resources
variable "token" {
    type = string
}

variable "ipv4" {
  default = ["85.75.103.43/32"]
}

provider "linode" {
  token = var.token
}
```

Εικόνα 12. Δήλωση παρόχου και μεταβλητών

Στην δεύτερη εικόνα παρατηρείται η δήλωση των δύο εικονικών μηχανημάτων που θα δημιουργηθούν για το σενάριο. Η μόνη διαφοροποίηση τους θα είναι το λειτουργικό(Ubuntu και Fedora) και η τεχνολογία με την οποία θα τρέξουν σε αυτά τα containers με την εφαρμογή. Αυτό θα βοηθήσει στην μελέτη επιπλέον χαρακτηριστικών σε επίπεδο λειτουργικού και τεχνολογιών container αλλά και αυτοματοποιήσεων όπως conditionals στο ansible.

```
# Declare instance with Ubuntu operating system
# Webpage https://registry.terraform.io/providers/linode/linode/latest/docs/
resources/instance
resource "linode_instance" "example" {
  label  = "example"
  image  = "linode/ubuntu22.04"
  region = "us-east"
  type   = "g6-nanode-1"
  root_pass = "toorroottoor"
}

# Declare instance with fedora operating system
resource "linode_instance" "example-fedora" {
  label  = "example-fedora"
  image  = "linode/fedora41"
  region = "us-east"
  type   = "g6-nanode-1"
  root_pass = "toorroottoor"
}
```

Εικόνα 13. Δήλωση εικονικών μηχανημάτων

Στην τρίτη φωτογραφία παρατηρείται ένα κομμάτι από τον κώδικα για την δημιουργία του firewall. Δηλώνονται τα default policies για εισερχόμενη και εξερχόμενη κίνηση καθώς και κανόνες για συγκεκριμένες πόρτες (π.χ 22) και πρωτόκολλα (π.χ ICMP).

```
1
2   # Declare a firewall on our provider. Allows access from specific IPs only to
    prevent unwanted external access to our scenario.
3   # Webpage https://registry.terraform.io/providers/linode/linode/latest/docs/
    resources/firewall
4   resource "linode_firewall" "example_firewall" {
5     label = "example_firewall"
6
7     linodes = [
8       linode_instance.example.id
9     ]
0
1     inbound_policy = "DROP"
2     outbound_policy = "ACCEPT"
3
4     inbound {
5       label     = "SSH"
6       action    = "ACCEPT"
7       protocol  = "TCP"
8       ports     = "22"
9       ipv4      = var.ipv4
0     }
1
2     inbound {
3       label     = "PING"
4       action    = "ACCEPT"
5       protocol  = "ICMP"
6       ipv4      = var.ipv4
7     }
8
9     inbound {
```

Εικόνα 14. Δήλωση firewall

Για την δημιουργία της υποδομής θα γίνει χρήση των εντολών “tofu plan” η οποία κάνει ένα σχεδιάγραμμα όλων των στοιχείων που θα φτιαχτούν και αφού γίνει η επαλήθευση τους γίνεται χρήση της εντολής “tofu apply”. Στην παρακάτω φωτογραφία φαίνεται ένα από τα στοιχεία της υποδομής όταν γίνεται το πλάνο αυτής.

```
linode_instance.example: Refreshing state... [id=69253145]                                                    11:53:22 [35/63]
linode_firewall.example_firewall: Refreshing state... [id=1460629]

OpenTofu used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

OpenTofu will perform the following actions:

  # ansible_playbook.playbook will be created
  + resource "ansible_playbook" "playbook" {
      + ansible_playbook_binary = "ansible-playbook"
      + ansible_playbook_stderr = (known after apply)
      + ansible_playbook_stdout = (known after apply)
      + args                    = (known after apply)
      + check_mode              = false
      + diff_mode               = true
      + extra_vars              = {
          + "become" = "true"
          + "user"   = "root"
        }
      + force_handlers          = false
      + groups                  = [
          + "webservers",
        ]
      + id                      = (known after apply)
      + ignore_playbook_failure = false
      + name                    = "96.126.107.46"
      + playbook                = "/home/knalpa/code/thesis/ansible/playbook.yml"
      + replayable              = true
      + temp_inventory_file     = (known after apply)
      + vault_password_file     = "/home/knalpa/code/thesis/ansible/become.yml"
      + verbosity               = 0
    }

  # ansible_playbook.playbook2 will be created
  + resource "ansible_playbook" "playbook2" {
```

Εικόνα 15. Πλάνο υποδομής

- Εγκατάσταση και αυτοματοποίηση Docker / Podman μέσω ansible

Στο κομμάτι αυτό θα γίνει η εγκατάσταση του απαραίτητου λογισμικού στον server μέσω ansible. Θα γίνει χρήση των ρόλων που είναι ένα χαρακτηριστικό του ansible. Ένας ρόλος[24] στο ansible είναι μια αυτοτελής, φορητή μονάδα αυτοματοποίησης που αποτελεί την προτιμώμενη μέθοδο ομαδοποίησης σχετικών εργασιών (tasks) και των συνδεδεμένων μεταβλητών, αρχείων, χειριστών (handlers) και άλλων πόρων σε μια γνωστή δομή αρχείων.

Παρόλο που οι εργασίες αυτοματοποίησης μπορούν να γραφούν αποκλειστικά σε ένα Ansible Playbook(μια μορφή script), οι Ansible Roles επιτρέπουν τη δημιουργία πακέτων αυτοματοποιημένου περιεχομένου που μπορούν να εκτελεστούν σε ένα ή περισσότερα plays, να επαναχρησιμοποιηθούν μεταξύ διαφορετικών playbooks και να μοιραστούν με άλλους χρήστες μέσω συλλογών (collections). Κάποια επιπλέον θετικά είναι η ευκολία στην ανάγνωση και το modularity. Είναι ευκολότερο με τον τρόπο αυτό να μπουν conditionals σε κομμάτια του κώδικα, μεταβλητές που αρμόζουν σε κάθε διαφορετική έκδοση Linux και η δυνατότητα δημιουργίας αυτοματοποιημένου testing στην συνέχεια.

Ο ρόλος που έχει φτιαχτεί έχει να κάνει με την εγκατάσταση των απαραίτητων πακέτων στο σύστημα. Θα γίνει χρήση των conditionals ώστε να διαχωριστεί η εγκατάσταση docker σε σύστημα Ubuntu και podman σε σύστημα Fedora. Οι τεχνολογίες είναι σχετικά παρόμοιες σε λειτουργίες αλλά το podman είναι φτιαγμένο από την RedHat και έχει καλύτερη υποστήριξη σε συστήματα της εταιρείας.

ansible > roles > ansible-role-docker > tasks > main.yml

```yaml
---
# tasks file for ansible-role-docker

- name: Ensure keyring directory exists
  ansible.builtin.file:
    path: /etc/apt/keyrings
    mode: '0755'
    state: directory
    owner: root
    group: root
  when: ansible_facts['distribution'] == "Ubuntu"

- name: Add docker keyring
  ansible.builtin.get_url:
    url: https://download.docker.com/linux/ubuntu/gpg
    dest: /etc/apt/keyrings/docker.asc
    mode: '0644'
  when: ansible_facts['distribution'] == "Ubuntu"

- name: Add docker repo
  ansible.builtin.lineinfile:
    path: /etc/apt/sources.list.d/tailscale.list
    mode: '0644'
    create: true
    line: 'deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc] https://
    download.docker.com/linux/ubuntu   jammy stable'
  when: ansible_facts['distribution'] == "Ubuntu"

```

Εικόνα 16. Ansible εγκατάσταση repository

ansible > roles > ansible-role-docker > tasks > main.yml

```yaml
- name: Install docker related packages
  ansible.builtin.apt:
    pkg:
      - docker-buildx-plugin
      - docker-compose-plugin
      - helix
    state: present
    update_cache: true
  when: ansible_facts['distribution'] == "Ubuntu"

- name: Run docker hello world
  ansible.builtin.shell: docker run hello-world
  register: result
  when: ansible_facts['distribution'] == "Ubuntu"

- name: Get output from variable
  ansible.builtin.debug:
    var: result
  when: ansible_facts['distribution'] == "Ubuntu"

# Redhat related
- name: Install podman related packages
  ansible.builtin.yum:
    pkg:
      - podman
      - podman-compose
      - neovim
    state: present
    update_cache: true
  when: ansible_facts['distribution'] == "Fedora"

```

Do you mind taking a quick feedback survey?

Take Survey | Remind Me Later | Don't Show Again

Εικόνα 17. Ansible εγκατάσταση πακέτων μέσω conditionals

- Ansible through terraform

Το κομμάτι αυτό είναι κάπως μικρότερο καθώς ουσιαστικά θα γίνει χρήση ενός του ansible provider[25] για terraform ώστε να τρέξουν τα playbook, και ως αποτέλεσμα τους ρόλους, κατευθείαν μετά την δημιουργία των μηχανημάτων. Έτσι θα αυξηθεί το επίπεδο της αυτοματοποίησης της εργασίας.

Στην εικόνα αυτή παρατηρείται το μπλοκ κώδικα με το οποίο καλεί το ansible να τρέξει ένα συγκεκριμένο playbook σε έναν server ο οποίος ορίζεται στον κώδικα.

Κατά την συγγραφή δυστυχώς υπάρχουν ακόμα περιορισμοί που δεν επιτρέπουν την εκτέλεση ενός playbook σε πολλαπλούς server χωρίς κάποιο workaround(π.χ με ένα loop στον κώδικα του terraform).

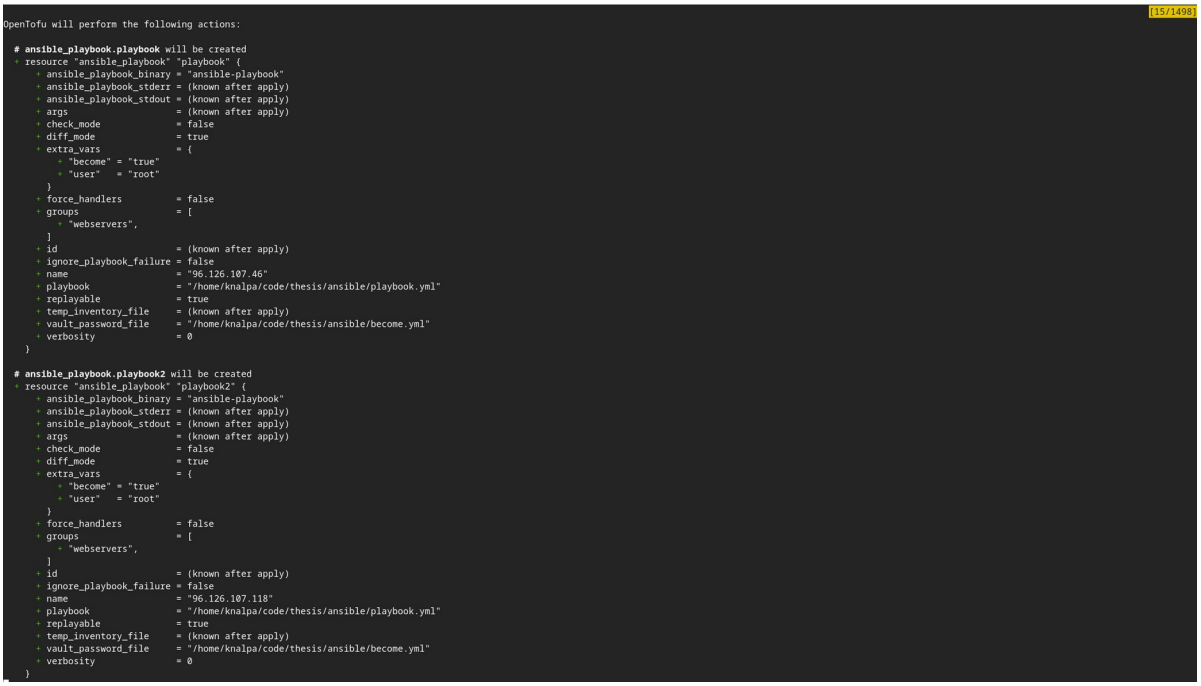
```
resource "ansible_playbook" "playbook" {
  playbook   = "/home/knalpa/code/thesis/ansible/playbook.yml"
  vault_password_file   = "/home/knalpa/code/thesis/ansible/become.yml"
  name       = "96.126.107.46"
  groups     = ["webservers"]
  #check_mode = "true"
  diff_mode  = "true"

extra_vars = {
  become = true
  user   = "root"
  }
}
```

Εικόνα 18. Ansible provider σε κώδικα terraform

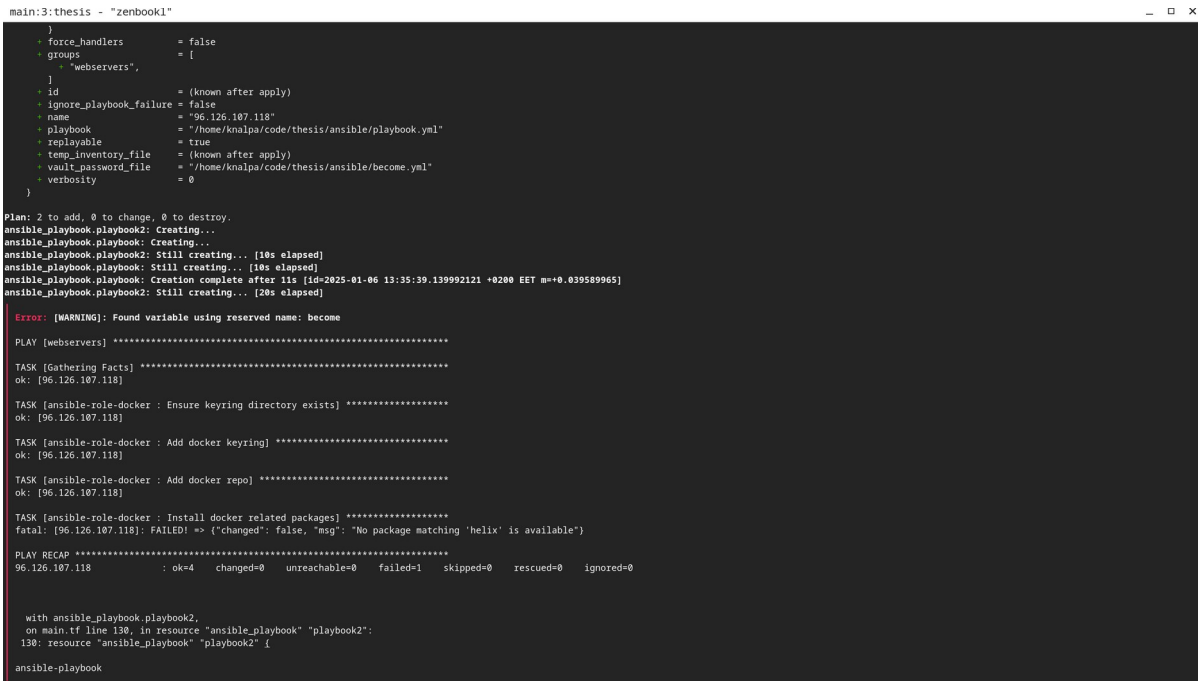
Παρακάτω παρατηρούνται τα αποτελέσματα του κώδικα ansible όταν εκτελείται το playbook μέσω terraform. Όπως φαίνεται δεν δίνει αρκετές πληροφορίες σε αντίθεση με το να τρέξει το playbook μόνο του μέσω της εντολής ansible-playbook(εικόνες 15 και 16) . Παρόλα αυτά, έχοντας τρέξει τον κώδικα νωρίτερα αλλά και έχοντας την δυνατότητα για αυτοματοποιημένα τεστ μέσω ansible, υπάρχει η βεβαιότητα ότι ο κώδικας τρέχει σωστά.

Εικόνα 15. Terraform apply



Εικόνα 19. Terraform plan output

Η επόμενη φωτογραφία μας δείχνει τι θα γινόταν αν κάποιο σημείο του κώδικα παρουσίαζε κάποιο πρόβλημα.



Εικόνα 20. Ansible output

- Docker – Podman

Η εφαρμογή θα σερβιριστεί με την βοήθεια του Docker / Podman. Συγκεκριμένα θα πρέπει να τρέξουν οι εντολές μεμονωμένα ώστε να επιβεβαιωθεί η σωστή λειτουργία τους και στην συνέχεια θα αυτοματοποιηθεί η διαδικασία με την βοήθεια του compose που προσφέρει η κάθε τεχνολογία. Δίνετε έτσι η δυνατότητα για περιγραφή με συγκεκριμένο τρόπο τι container χρειάζεται με τις παραμέτρους τους καθώς και το πως συνδέονται μεταξύ τους, δικτυακά και ακόμα και με συγκεκριμένη σειρά εξάρτησης(π.χ δημιουργία βάσης δεδομένων πρώτα και μετά δημιουργία web εφαρμογής).

Η εντολή για να τρέξει το juice-shop μόνο είναι η παρακάτω:

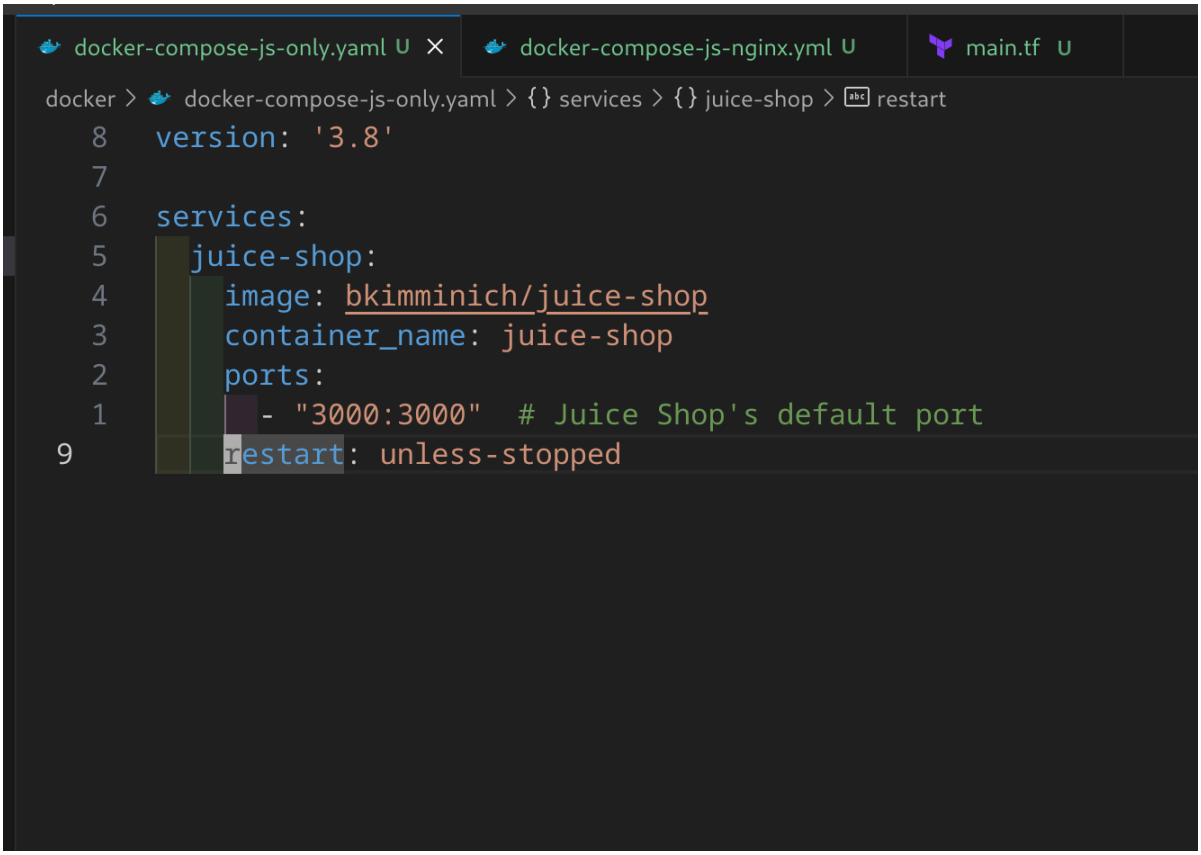
“docker run -d -p 3000:3000 -v juice-shop-data:/data bkimminich/juice-shop”

Με αυτήν την εντολή γίνεται ανάθεση πορτών εντός και εκτός container με το -p και μονάδων αποθήκευσης με το -v. Πριν την άνω και κάτω τελεία γίνεται αναφορά στο host σύστημα και μετά στο docker container.

Αφού γίνει η δοκιμή και είναι επιτυχής, για να τρέξει το αρχείο compose το οποίο περιέχει όλη την υποδομή της εφαρμογής θα χρειαστεί η εντολή

“docker-compose -f docker-compose.yml up -d ”

Παρακάτω παραθέτονται τρεις φωτογραφίες με την υποδομή σε μορφή κώδικα σε τρία διαφορετικά στάδια δοκιμών.



Εικόνα 21. Docker installation του Juice shop

docker-compose-js-only.yaml U | docker-compose-js-nginx.yml U | main.tf U

docker > js-nginx > docker-compose-js-nginx.yml > { } services > { } nginx

```yaml
version: '3.8'

services:
  juice-shop:
    image: bkimminich/juice-shop
    container_name: juice-shop
    ports:
      - "3000:3000"  # Juice Shop's default port
    restart: unless-stopped

  nginx:
    image: nginx
    container_name: nginx
    depends_on:
      - juice-shop
    ports:
      - "80:80"  # Expose Nginx on port 80
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro  # Mount Nginx config
    restart: unless-stopped

```

Εικόνα 22. Εγκατάσταση juice shop μαζί με nginx reverse proxy

docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications (compose-spec.json)

```yaml
version: '3.8'

services:
  juice-shop:
    image: bkimminich/juice-shop
    container_name: juice-shop
    networks:
      - net
    ports:
      - "3000:3000" # Juice-shop default port
    restart: unless-stopped

  apache:
    image: owasp/modsecurity-crs:apache
    container_name: apache-modsec
    user: root
    depends_on:
      - juice-shop
    networks:
      - net
    ports:
      - "8080:8080"
    environment:
      SERVERNAME: modsec2-apache
      BACKEND: http://juice-shop:3000
      PORT: "8080"
      #MODSEC_RULE_ENGINE: DetectionOnly
      BLOCKING_PARANOIA: 2
      TZ: "${TZ}"
      #ERRORLOG: "/var/log/error.log"
      #ACCESSLOG: "/var/log/access.log"
      MODSEC_AUDIT_LOG_FORMAT: Native
      MODSEC_AUDIT_LOG_TYPE: Serial
      MODSEC_AUDIT_LOG: "/var/log/modsec_audit.log"
      MODSEC_TMP_DIR: "/tmp"
      MODSEC_RESP_BODY_ACCESS: "On"
      MODSEC_RESP_BODY_MIMETYPE: "text/plain text/html text/xml application/
      json"
      COMBINED_FILE_SIZES: "65535"
    restart: unless-stopped

networks:
  net:
```

Εικόνα 23. Εγκατάσταση juice shop μαζί με apache reverse proxy και χρήση modsecurity

4. Ενίσχυση ασφάλειας υποδομών με την εφαρμογή τεχνολογιών υποδομής ως κώδικα

Στο κεφάλαιο αυτό θα γίνει μελέτη τεχνολογιών ασφάλειας σε web εφαρμογές και η χρήση αυτών για τον σχεδιασμό σεναρίων και υλοποίηση τους, δηλαδή αναζήτηση ευπαθειών και εκμετάλλευση αυτών.

4.1 Θεωρητικό υπόβαθρο

Οι ευπάθειες[26] είναι αδυναμίες σε ένα σύστημα υπολογιστή ή σε εφαρμογές που αποδυναμώνουν τη συνολική ασφάλεια αυτών.

Παρά τις προθέσεις για επίτευξη πλήρους ορθότητας, σχεδόν όλο το υλικό και το λογισμικό περιέχει σφάλματα όπου το σύστημα δεν συμπεριφέρεται όπως αναμένεται. Εάν το σφάλμα θα μπορούσε να επιτρέψει σε έναν επιτιθέμενο να παραβιάσει την εμπιστευτικότητα, την ακεραιότητα ή τη διαθεσιμότητα των πόρων του συστήματος, τότε αυτό ονομάζεται ευπάθεια. Οι μη ασφαλείς πρακτικές ανάπτυξης λογισμικού, καθώς και παράγοντες σχεδιασμού, όπως η πολυπλοκότητα, μπορούν να αυξήσουν την επιβάρυνση από τις ευπάθειες. Υπάρχουν διάφοροι τύποι, πιο συχνοί σε διαφορετικά σημεία όπως το υλικό, τα λειτουργικά συστήματα και οι εφαρμογές.

Οι ευπάθειες μπορούν να αξιολογηθούν για τον κίνδυνο σύμφωνα με το Κοινό Σύστημα Αξιολόγησης Ευπαθειών (Common Vulnerability Scoring System) ή άλλα συστήματα, και να προστεθούν σε βάσεις δεδομένων ευπαθειών. Από το 2023, υπάρχουν περισσότερες από 20 εκατομμύρια ευπάθειες καταγεγραμμένες στη βάση δεδομένων Κοινών Ευπαθειών και Εκθέσεων (Common Vulnerabilities and Exposures – CVE).

Μια ευπάθεια ξεκινά όταν εισάγεται στο υλικό ή το λογισμικό. Γίνεται ενεργή και εκμεταλλεύσιμη όταν το λογισμικό ή το υλικό που περιέχει την ευπάθεια είναι σε λειτουργία. Η ευπάθεια μπορεί να ανακαλυφθεί από τον προμηθευτή ή από τρίτο μέρος. Η γνωστοποίηση της ευπάθειας (ως ενημέρωση κώδικα ή με άλλο τρόπο) σχετίζεται με αυξημένο κίνδυνο παραβίασης, καθώς οι επιτιθέμενοι συχνά κινούνται ταχύτερα από την ανάπτυξη των ενημερώσεων κώδικα. Ανεξάρτητα από το αν θα κυκλοφορήσει ποτέ μια ενημέρωση κώδικα για την αντιμετώπιση της ευπάθειας, ο κύκλος ζωής της θα λήξει τελικά όταν το σύστημα ή οι παλαιότερες εκδόσεις του τεθούν εκτός χρήσης.

Παρά τον στόχο των προγραμματιστών να παραδώσουν ένα προϊόν που λειτουργεί εξ ολοκλήρου όπως προορίζεται, σχεδόν όλο το λογισμικό και το υλικό περιέχουν σφάλματα. Εάν ένα σφάλμα δημιουργεί κίνδυνο ασφάλειας, ονομάζεται ευπάθεια. Παρόλο που ορισμένες ευπάθειες μπορούν να χρησιμοποιηθούν μόνο για επιθέσεις άρνησης υπηρεσίας, πιο επικίνδυνες είναι αυτές που επιτρέπουν στον επιτιθέμενο να εισάγει και να εκτελέσει τον δικό του κώδικα (ονομαζόμενο κακόβουλο λογισμικό), χωρίς ο χρήστης να το αντιληφθεί. Μόνο μια μικρή μερίδα ευπαθειών επιτρέπει κλιμάκωση προνομίων, που είναι απαραίτητη για πιο σοβαρές επιθέσεις. Χωρίς μια ευπάθεια, η εκμετάλλευση δεν μπορεί να αποκτήσει πρόσβαση. Είναι επίσης δυνατό να εγκατασταθεί κακόβουλο λογισμικό απευθείας, χωρίς εκμετάλλευση, αν ο επιτιθέμενος χρησιμοποιήσει κοινωνική μηχανική ή εμφυτεύσει το κακόβουλο λογισμικό σε νόμιμο λογισμικό που κατεβαίνει σκόπιμα.

Βασικοί παράγοντες σχεδιασμού που μπορούν να αυξήσουν την επιβάρυνση από τις ευπάθειες περιλαμβάνουν:

- **Πολυπλοκότητα**: Τα μεγάλα, πολύπλοκα συστήματα αυξάνουν την πιθανότητα σφαλμάτων και μη προγραμματισμένων σημείων πρόσβασης.

- **Οικειότητα**: Η χρήση κοινού, γνωστού κώδικα, λογισμικού, λειτουργικών συστημάτων και/ή υλικού αυξάνει την πιθανότητα ο επιτιθέμενος να έχει ή να μπορεί να βρει τη γνώση και τα εργαλεία για να εκμεταλλευτεί το σφάλμα.

- **Συνδεσιμότητα**: Οποιοδήποτε σύστημα συνδεδεμένο στο διαδίκτυο μπορεί να παραβιαστεί και να υποστεί επίθεση. Η αποσύνδεση των συστημάτων από το διαδίκτυο είναι ένα πραγματικά αποτελεσματικό μέτρο κατά των επιθέσεων, αλλά σπάνια είναι εφικτό.

- **Παλαιό λογισμικό και υλικό**: Το παλαιό λογισμικό και υλικό διατρέχει αυξημένο κίνδυνο, αλλά η συχνή αναβάθμιση είναι συχνά απαγορευτική όσον αφορά το κόστος και τον χρόνο εκτός λειτουργίας.

Οι διαδικτυακές εφαρμογές λειτουργούν σε πολλούς ιστότοπους. Επειδή είναι εκ φύσεως λιγότερο ασφαλείς από άλλες εφαρμογές, αποτελούν μία από τις κύριες πηγές παραβιάσεων δεδομένων και άλλων περιστατικών ασφάλειας.

Κοινά είδη ευπαθειών που εντοπίζονται σε αυτές τις εφαρμογές περιλαμβάνουν:

- XSS

Οι επιθέσεις Cross-Site Scripting (XSS)[[27]είναι ένας τύπος injection, όπου κακόβουλα scripts εισάγονται σε κατά τα άλλα αξιόπιστες και ασφαλείς ιστοσελίδες. Οι επιθέσεις XSS συμβαίνουν όταν ένας επιτιθέμενος χρησιμοποιεί μια εφαρμογή ιστού για να στείλει κακόβουλο κώδικα, συνήθως με τη μορφή script που εκτελείται στον περιηγητή, σε έναν άλλο τελικό χρήστη. Τα κενά που επιτρέπουν την επιτυχία αυτών των επιθέσεων είναι ευρέως διαδεδομένα και εμφανίζονται όπου μια εφαρμογή ιστού χρησιμοποιεί εισαγωγή από έναν χρήστη στο περιεχόμενο που παράγει, χωρίς να την ελέγχει ή να την κωδικοποιεί.

Ένας επιτιθέμενος μπορεί να χρησιμοποιήσει την XSS για να στείλει ένα κακόβουλο script σε έναν ανυποψίαστο χρήστη. Ο περιηγητής του τελικού χρήστη δεν έχει τρόπο να γνωρίζει ότι το script δεν πρέπει να είναι αξιόπιστο και θα το εκτελέσει. Επειδή θεωρεί ότι το script προέρχεται από αξιόπιστη πηγή, το κακόβουλο script μπορεί να αποκτήσει πρόσβαση σε cookies, tokens συνεδρίας ή άλλες ευαίσθητες πληροφορίες που διατηρεί ο περιηγητής και χρησιμοποιούνται με τον συγκεκριμένο ιστότοπο. Αυτά τα scripts μπορούν ακόμη και να τροποποιήσουν το περιεχόμενο της HTML σελίδας.

- Injection

Η επίθεση injection [28] είναι μια προσπάθεια ενός επιτιθέμενου να στείλει δεδομένα σε μια εφαρμογή με τρόπο που αλλάζει τη σημασία των εντολών που αποστέλλονται σε έναν διερμηνέα. Ένα από τα πιο συνηθισμένα παραδείγματα είναι η SQL Injection, όπου ένας επιτιθέμενος στέλνει δεδομένα όπως το "101 OR 1=1" αντί για απλά "101". Όταν αυτό το δεδομένο ενσωματωθεί σε ένα SQL query, αλλάζει τη σημασία του ώστε να επιστρέφει όλες τις εγγραφές αντί για μία συγκεκριμένη.

Υπάρχουν πολλοί διερμηνείς σε ένα τυπικό περιβάλλον ιστού, όπως SQL, LDAP, Λειτουργικά Συστήματα, XPath, XQuery, Expression Language και άλλοι. Οποιοδήποτε σύστημα διαθέτει "διεπαφή εντολών" που συνδυάζει δεδομένα με εντολές είναι ευάλωτο. Ακόμη και οι επιθέσεις XSS μπορούν να θεωρηθούν ως μια μορφή HTML Injection.

Συχνά, αυτοί οι διερμηνείς εκτελούνται με ευρεία δικαιώματα πρόσβασης, επομένως μια επιτυχημένη επίθεση μπορεί εύκολα να οδηγήσει σε σημαντικές παραβιάσεις δεδομένων ή ακόμη και σε απώλεια ελέγχου ενός περιηγητή, μιας εφαρμογής ή ενός διακομιστή. Οι επιθέσεις injection αποτελούν σημαντικό ποσοστό των σοβαρών κινδύνων ασφάλειας εφαρμογών.

Πολλοί οργανισμοί εφαρμόζουν ανεπαρκή μέτρα ασφαλείας για την πρόληψη αυτών των επιθέσεων. Γενικές συστάσεις, όπως ο έλεγχος εισαγωγών και η κωδικοποίηση εξόδου, δεν

επαρκούν για την αποτροπή αυτών των ευπαθειών. Αντίθετα, συνιστάται η χρήση ισχυρών ελέγχων που ενσωματώνονται στα frameworks των εφαρμογών. Ο στόχος είναι να γίνει η injection αδύνατη για τους προγραμματιστές.

- CSRF

Η επίθεση Cross-Site Request Forgery (CSRF)[29] είναι μια επίθεση που αναγκάζει έναν τελικό χρήστη να εκτελέσει ανεπιθύμητες ενέργειες σε μια διαδικτυακή εφαρμογή στην οποία είναι ήδη αυθεντικοποιημένος. Με τη βοήθεια της κοινωνικής μηχανικής (social engineering) (όπως η αποστολή ενός συνδέσμου μέσω email ή συνομιλίας), ένας επιτιθέμενος μπορεί να παραπλανήσει τους χρήστες μιας διαδικτυακής εφαρμογής ώστε να εκτελέσουν ενέργειες της επιλογής του επιτιθέμενου.

Αν το θύμα είναι ένας απλός χρήστης, μια επιτυχημένη επίθεση CSRF μπορεί να τον αναγκάσει να εκτελέσει αιτήματα που αλλάζουν την κατάσταση, όπως η μεταφορά χρημάτων, η αλλαγή της διεύθυνσης ηλεκτρονικού ταχυδρομείου και άλλα. Αν το θύμα είναι ένας λογαριασμός διαχειριστή, η επίθεση CSRF μπορεί να θέσει σε κίνδυνο ολόκληρη την διαδικτυακή εφαρμογή.

4.2 Τεχνολογίες πειραματικών δοκιμών ασφάλειας web εφαρμογών

- Juice-shop

Το OWASP Juice Shop[30] είναι πιθανότατα η πιο σύγχρονη και εξελιγμένη μη ασφαλής εφαρμογή ιστού. Μπορεί να χρησιμοποιηθεί σε εκπαιδεύσεις ασφάλειας, επιδείξεις ευαισθητοποίησης, CTFs (Capture The Flag) και ως πειραματική εφαρμογή για εργαλεία ασφάλειας. Το Juice Shop περιλαμβάνει ευπάθειες από ολόκληρο το OWASP Top Ten, καθώς και πολλές άλλες αδυναμίες ασφαλείας που συναντώνται σε εφαρμογές του πραγματικού κόσμου.

Η εφαρμογή είναι γραμμένη σε Node.js, Express και Angular. Ήταν η πρώτη εφαρμογή γραμμένη εξ ολοκλήρου σε JavaScript που συμπεριλήφθηκε στον OWASP VWA Directory.

Η εφαρμογή περιέχει έναν μεγάλο αριθμό προκλήσεων hacking με διαφορετικό επίπεδο δυσκολίας, όπου ο χρήστης καλείται να εκμεταλλευτεί τις υποκείμενες ευπάθειες. Η πρόοδος του hacking παρακολουθείται μέσω ενός πίνακα βαθμολογίας (score board), του οποίου η εύρεση αποτελεί μία από τις εύκολες προκλήσεις.

Εκτός από την εκπαίδευση σε hacking και την ευαισθητοποίηση, το Juice Shop μπορεί να χρησιμοποιηθεί από pentesting proxies ή εργαλεία σάρωσης ασφαλείας ως πειραματική εφαρμογή για να αξιολογηθεί η αποτελεσματικότητά τους σε εφαρμογές με JavaScript-heavy frontends και REST APIs.

Εναλλακτικές για το juice-shop είναι τα DVWA(Damn Vulnerable Web Application), bWAPP, WebGoat από τον OWASP, hackazon από την rapid7 και pixi μεταξύ άλλων. Τα περισσίτερα προσφέρουν παρόμοιες δυνατότητες αλλά το juice-shop μας δίνει και την δυνατότητα με ευκολία να σερβιριστεί μέσω docker ή και kubrnetes μειώνοντας έτσι τον χρόνο που χρειάζεται για την διεξαγωγή των επιθέσεων.

- Juiceshop kubernetes - Multijuicer

Η διεξαγωγή CTFs και εκπαιδεύσεων ασφάλειας με το OWASP Juice Shop είναι συνήθως αρκετά δύσκολη, καθώς το Juice Shop δεν έχει σχεδιαστεί για να χρησιμοποιείται από πολλούς χρήστες ταυτόχρονα. Το να δωθεί οδηγία σε κάθε συμμετέχοντα στο να ξεκινήσουν το Juice Shop στη δική του εικονική μηχανή λειτουργεί, αλλά καταναλώνει πολύτιμο χρόνο.

Το MultiJuicer[31] δίνει τη δυνατότητα να εκτελούνται ξεχωριστές περιπτώσεις (instances) του Juice Shop για κάθε συμμετέχοντα σε έναν κεντρικό Kubernetes cluster, διευκολύνοντας τη διεξαγωγή εκδηλώσεων χωρίς την ανάγκη για τοπικές εγκαταστάσεις του Juice Shop.

Τι κάνει:

- Δημιουργεί δυναμικά νέα instances του Juice Shop όταν χρειάζεται.
- Λειτουργεί σε ένα ενιαίο domain και περιλαμβάνει ένα LoadBalancer που διανέμει την κίνηση στο instance του Juice Shop κάθε συμμετέχοντα.
- Κάνει backup και εφαρμόζει αυτόματα την πρόοδο των προκλήσεων σε περίπτωση επανεκκίνησης του Juice Shop container.
- Καθαρίζει αυτόματα παλιά και αχρησιμοποίητα instances.

Για την εργασία θα γίνει χρήση του απλού juice-shop λογισμικού καθώς η δημιουργία του είναι πιο απλή και για δοκιμές σε μικρό σκέλος και ομάδες είναι ιδανικό.

- Web application firewall

Ένα web application firewall (WAF) είναι μια συγκεκριμένη μορφή firewall εφαρμογών που φιλτράρει, παρακολουθεί και μπλοκάρει την κίνηση HTTP προς και από μια web υπηρεσία. Μέσω της επιθεώρησης της κίνησης HTTP, μπορεί να αποτρέψει επιθέσεις που εκμεταλλεύονται γνωστές ευπάθειες μιας web εφαρμογής, όπως είναι η SQL injection, η διασταυρωμένη εκτέλεση σεναρίων (cross-site scripting, XSS), η περίληψη αρχείων και η ακατάλληλη διαμόρφωση του συστήματος. Μπορεί να εισαγάγουν μείωση της απόδοσης χωρίς τη σωστή διαμόρφωση και ρύθμιση από ειδικούς Κυβερνοασφάλειας. Ωστόσο, τα περισσότερα από τα μεγαλύτερα χρηματοοικονομικά ιδρύματα χρησιμοποιούν τα WAF για να βοηθήσουν στη μετρίαση των ευπαθειών 'μηδενικής ημέρας' των web εφαρμογών, καθώς και δύσκολων στην επιδιόρθωση σφαλμάτων ή αδυναμιών μέσω προσαρμοσμένων συμβολοσειρών υπογραφής επιθέσεων.

Το WAF θα χρησιμοποιηθεί στο δεύτερο μέρος του κεφαλαίου αυτού. Πρώτα θα γίνει επίδειξη επιθέσεις σε ευάλωτη εφαρμογή, εφαρμογή που θα μπορούσε να είναι και παραγωγική σε επαγγελματικά πλαίσια, και στην συνέχεια θα ρυθμιστεί αυτή ώστε η κίνηση να περνάει πρώτα από ένα WAF. Έτσι θα αναλυθεί αν είναι ένας επαρκής τρόπος για να αντιμετωπιστούν οι επιθέσεις αυτές.

- Modsecurity

Το ModSecurity[32] είναι μια μηχανή Web Application Firewall (WAF) ανοιχτού κώδικα και πολλαπλών πλατφορμών για Apache, IIS και Nginx. Διαθέτει μια ισχυρή γλώσσα προγραμματισμού βασισμένη σε γεγονότα, η οποία προσφέρει προστασία από μια σειρά επιθέσεων κατά των εφαρμογών ιστού, ενώ επιτρέπει την παρακολούθηση, καταγραφή και ανάλυση της HTTP κίνησης σε πραγματικό χρόνο.

Χαρακτηριστικά του Modsecurity είναι τα εξής:

- Παρακολούθηση ασφάλειας εφαρμογών και έλεγχος πρόσβασης σε πραγματικό χρόνο
- Πλήρης καταγραφή της HTTP κίνησης
- Συνεχής παθητική αξιολόγηση ασφάλειας
- Ενίσχυση της ασφάλειας εφαρμογών ιστού

- App protect

Το F5 NGINX App Protect[33] προσφέρει λύση ασφάλειας λογισμικού που προστατεύει εφαρμογές και APIs, ενσωματώνεται άψογα σε περιβάλλοντα DevOps και λειτουργεί ως ελαφρύ Web Application Firewall (WAF). Παρέχει προστασία από επιθέσεις Denial-of-Service (DoS) στο

επίπεδο 7, προστασία από bots, ασφάλεια APIs και υπηρεσίες ανάλυσης απειλών, διασφαλίζοντας συνεπή προστασία σε κατανεμημένες αρχιτεκτονικές και υβριδικά περιβάλλοντα.

Το προϊόν επιταχύνει την ανάπτυξη ασφαλών εφαρμογών μέσω αυτοματοποιημένων διαδικασιών ασφάλειας που ενσωματώνονται στη ροή εργασιών ανάπτυξης. Χρησιμοποιεί μηχανική μάθηση για την αυτόματη ρύθμιση πολιτικών ασφαλείας, μειώνοντας το λειτουργικό κόστος. Παρέχει κεντρική διαχείριση, διευκολύνοντας την ορατότητα και τον έλεγχο πολιτικών ασφαλείας για ολόκληρο τον στόλο WAF. Επιπλέον, υποστηρίζει ασφαλή APIs, όπως gRPC, GraphQL, REST, και καλύπτει τις προδιαγραφές του OWASP Top 10 APIs.

- Openappsec

Το open-appsec[34] είναι μια μηχανή ασφάλειας που βασίζεται στη μηχανική μάθηση και προληπτικά, καθώς και αυτόματα, αποτρέπει απειλές κατά των Web Applications και APIs. Τα χαρακτηριστικά του είναι:

Προληπτική Προστασία: Αποτρέπει απειλές OWASP Top 10 και zero-day επιθέσεις κατά εφαρμογών ιστού και APIs, χρησιμοποιώντας ασφάλεια βασισμένη στη μηχανική μάθηση, χωρίς ανάγκη για ενημερώσεις υπογραφών (π.χ., μπλοκάρισμα των Log4Shell και Spring4Shell χωρίς ενημερώσεις).

Ακρίβεια: Η συνεχής μάθηση εξασφαλίζει ακριβή ανίχνευση, εντοπίζοντας περισσότερες επιθέσεις ενώ εξαλείφει την ανάγκη για συνεχή ρύθμιση και δημιουργία εξαιρέσεων που είναι απαραίτητες σε παραδοσιακά WAFs.

Αυτοματοποίηση: Υποστηρίζει cloud native ανάπτυξη φιλική σε περιβάλλοντα CI/CD, με αυτοματοποίηση από την εγκατάσταση έως τις αναβαθμίσεις και τη διαμόρφωση, χρησιμοποιώντας declarative infrastructure-as-code ή APIs.

Open Source WAF: Προσφέρει αποτελεσματική ασφάλεια για εφαρμογές ιστού και APIs με μια μηχανή βασισμένη στη μηχανική μάθηση, εύκολη στη ρύθμιση και διαχείριση, διαθέσιμη για όλους να τη χρησιμοποιήσουν και να την επεκτείνουν.

Από τα παραπάνω θα γίνει επιλογή του Modsecurity καθώς παρέχει container το οποίο περιλαμβάνει μέσα σε αυτό και το coreruleset και είναι χρήσιμο για την ευκολότερη σύνδεση με το container του juice-shop.

- Coreruleset (CRS)

Το OWASP Core Rule Set (CRS)[35] είναι ένα σύνολο γενικών κανόνων ανίχνευσης επιθέσεων που χρησιμοποιούνται με το ModSecurity ή άλλα συμβατά Web Application Firewalls. Στόχος του είναι να προστατεύει εφαρμογές ιστού από ένα ευρύ φάσμα επιθέσεων, συμπεριλαμβανομένων αυτών που καταγράφονται στο OWASP Top Ten, με ελάχιστες ψευδείς ειδοποιήσεις.

Το CRS παρέχει προστασία έναντι πολλών κοινών κατηγοριών επιθέσεων, όπως SQL Injection, Cross-Site Scripting (XSS), Local File Inclusion και άλλες.

Το CRS 4 η οποία είναι και η πιο πρόσφατη έκδοση περιλαμβάνει πολλές βελτιώσεις κάλυψης, καθώς και τα ακόλουθα νέα χαρακτηριστικά:

- Αρχιτεκτονική plug-in, που επιτρέπει την ενσωμάτωση επίσημων και τρίτων plugins στο CRS.
- Δυνατότητα Early-Blocking για άμεση διακοπή επιθέσεων.
- Πάνω από 500 παρακάμψεις κανόνων διορθώθηκαν μέσω ενός μεγάλου προγράμματος Bug Bounty.
- Νέα δυνατότητα ανίχνευσης web shells.
- Πλήρης συμβατότητα με RE2/Hyperscan για βελτιωμένη απόδοση.
- Υποστήριξη για HTTP/3.

- Πιο λεπτομερείς επιλογές αναφοράς.

Επιθέσεις από τις οποίες προστατεύει

- **Cross Site Scripting (XSS)**
- **Local File Inclusion (LFI)**
- **Remote File Inclusion (RFI)**
- **PHP Code Injection**
- **Java Code Injection**
- **HTTPoxy**
- **Shellshock**
- **Unix/Windows Shell Injection**
- **Session Fixation**
- **Scanner/Bot Detection**
- **Metadata/Error Leakages**
- **SQL Injection (SQLi)**

4.3 Σενάρια επιθέσεων

Στην ενότητα αυτή θα γίνει ανάλυση κάποιων θεωρητικών σεναρίων και παράλληλα η μεθοδολογία που ακολουθείται για την προσέγγιση της εύρεσης και εκμετάλλευσης ευπαθειών. Επιπλέον παρατίθενται και στοιχεία της εφαρμογής και υποδομής αλλά και πως συνδέεται το IaC από το προηγούμενο κεφάλαιο για την ανάπτυξη αυτών.

Σε τέτοιου είδους σενάρια η αυτοματοποίηση συστημάτων και το IAC (Infrastructure as Code) είναι πλέον άρικτα συνδεδεμένα. Μας επιτρέπει αρχικά να σχεδιάσουμε την υποδομή με μεγαλύτερη ευκολία και βοηθάει επιπλέον στην ανάγνωση αυτής.

Στα CTF, τα σενάρια συνήθως απαιτούν σύνθετες αρχιτεκτονικές, που περιλαμβάνουν δίκτυα, ευάλωτους servers, εφαρμογές και διαφορετικά περιβάλλοντα (on-premises ή cloud). Με το IaC, ως διοργανωτές μπορούμε να αυτοματοποιήσουμε τη δημιουργία των σύνθετων αυτών υποδομών και να την αναπτύξουμε με ακρίβεια. Επιπλέον αποφεύγονται σφάλματα χειρωνακτικής ρύθμισης και μπορούμε να προσαρμώσουμε γρήγορα τις απαιτήσεις του σεναρίου, όπως η κλιμάκωση ή η προσθήκη νέων στοιχείων.

- Σενάριο επίθεσης sql injection σε web application

Το σενάριο επικεντρώνεται σε μια ευπάθεια SQL Injection σε μια εφαρμογή web. Η ευπάθεια προκύπτει από την ανεπαρκή επεξεργασία εισόδων χρήστη σε ένα πεδίο αναζήτησης, το οποίο συνδέεται απευθείας με τη βάση δεδομένων. Οι συμμετέχοντες καλούνται να εκμεταλλευτούν αυτή την ευπάθεια για να εξάγουν δεδομένα από τη βάση.

Αναγνώριση Ευπάθειας:

- Οι συμμετέχοντες εντοπίζουν ότι το πεδίο αναζήτησης της εφαρμογής δεν φιλτράρει σωστά τις εισόδους.
- Εκτέλεση SQL Injection:
- Χρησιμοποιούν injection payloads για την ανάκτηση δεδομένων χρηστών από τη βάση.
- Ανάκτηση Ευαίσθητων Πληροφοριών:

- Εξαγωγή στοιχείων χρηστών, όπως usernames και passwords.
- Διδασκαλία Ασφαλών Πρακτικών:
- Οι συμμετέχοντες κατανοούν τη σημασία της χρήσης prepared statements και του validation εισόδων.

Infrastructure as Code (IaC):

- Χρησιμοποιείται για την αυτοματοποίηση της ανάπτυξης της εφαρμογής.
- Επιτρέπει τη γρήγορη επαναφορά του περιβάλλοντος για επόμενα CTF.
- Web Application:
- Μια εφαρμογή web βασισμένη σε PHP και MySQL ή Python (Flask/Django) που περιλαμβάνει την ευπάθεια.
- Database Server:
- Μια βάση δεδομένων MySQL/PostgreSQL που περιέχει ευαίσθητες πληροφορίες χρηστών.

Κατανόηση Ευπάθειας SQL Injection:

- Οι συμμετέχοντες βλέπουν πώς οι μη ασφαλείς είσοδος μπορεί να αλλοιώσει ένα SQL query.

2. Εκπαίδευση σε Προστατευτικά Μέτρα:
    - Χρήση prepared statements (π.χ., `PDO` στην PHP ή `Parameterized Queries` στην Python).
    - Validation και sanitization των εισόδων χρηστών.
    - Περιορισμός δικαιωμάτων στη βάση δεδομένων (π.χ., αποτροπή εκτέλεσης `DROP TABLE` ή `UNION` από χρήστες).

Πολλαπλές Ευπάθειες:

- Συνδυασμός SQL Injection με άλλες ευπάθειες, όπως XSS ή Broken Authentication.
- Ευπάθεια στοχευμένη σε Admin Portal:
- Δημιουργία admin panel με ευπάθειες SQL Injection για πιο προχωρημένες δοκιμές.
- Ενσωμάτωση Εργαλείων Εντοπισμού Ευπαθειών:
- Διδασκαλία χρήσης εργαλείων, όπως το `sqlmap`, για την αυτοματοποίηση της επίθεσης.

4.4 Σενάριο επίθεσης σε εφαρμογή με LDAP authentication

Το Lightweight Directory Access Protocol (LDAP)[36] είναι ένα πρωτόκολλο που βοηθά τους χρήστες να βρουν δεδομένα σχετικά με οργανισμούς, άτομα και άλλα. Το LDAP έχει δύο βασικούς στόχους: την αποθήκευση δεδομένων στον κατάλογο LDAP και τον έλεγχο ταυτότητας των χρηστών για την πρόσβαση στον κατάλογο. Επιπλέον, παρέχει τη γλώσσα επικοινωνίας που απαιτούν οι εφαρμογές για να στέλνουν και να λαμβάνουν πληροφορίες από υπηρεσίες καταλόγου. Μια υπηρεσία καταλόγου παρέχει πρόσβαση σε πληροφορίες που αφορούν οργανισμούς, άτομα και άλλα δεδομένα, όπως είναι τοποθετημένα μέσα σε ένα δίκτυο.

Η πιο κοινή περίπτωση χρήσης του LDAP είναι η παροχή ενός κεντρικού σημείου πρόσβασης και διαχείρισης υπηρεσιών καταλόγου. Το LDAP επιτρέπει στις οργανώσεις να αποθηκεύουν, να διαχειρίζονται και να διασφαλίζουν πληροφορίες σχετικά με τον οργανισμό, τους χρήστες του και τα περιουσιακά του στοιχεία – όπως ονόματα χρηστών και κωδικούς πρόσβασης. Αυτό βοηθά στην απλοποίηση της πρόσβασης στα δεδομένα μέσω της παροχής μιας ιεραρχικής δομής

πληροφοριών, και μπορεί να είναι κρίσιμο για τις εταιρείες καθώς αναπτύσσονται και αποκτούν περισσότερα δεδομένα χρηστών και περιουσιακά στοιχεία.

Το LDAP λειτουργεί επίσης ως λύση διαχείρισης ταυτότητας και πρόσβασης (IAM), η οποία επικεντρώνεται στον έλεγχο ταυτότητας χρηστών, περιλαμβάνοντας υποστήριξη για Kerberos και Single Sign-On (SSO), το Simple Authentication Security Layer (SASL) και το Secure Sockets Layer (SSL).

Η επίθεση LDAP Injection χρησιμοποιείται για την εκμετάλλευση διαδικτυακών εφαρμογών που δημιουργούν δηλώσεις LDAP βάσει εισόδου του χρήστη. Όταν μια εφαρμογή δεν φιλτράρει σωστά τα δεδομένα που εισάγει ο χρήστης, είναι δυνατόν να τροποποιηθούν οι δηλώσεις LDAP μέσω ενός τοπικού διαμεσολαβητή (proxy). Αυτό μπορεί να οδηγήσει στην εκτέλεση αυθαίρετων εντολών, όπως η παραχώρηση δικαιωμάτων σε μη εξουσιοδοτημένα ερωτήματα ή η τροποποίηση περιεχομένου μέσα στο δέντρο LDAP.

Οι ίδιες προηγμένες τεχνικές εκμετάλλευσης που χρησιμοποιούνται σε επιθέσεις SQL Injection μπορούν να εφαρμοστούν παρόμοια και σε επιθέσεις LDAP Injection[37].

Περιγραφή σεναρίου:

Το συγκεκριμένο σενάριο επικεντρώνεται σε μια ευπάθεια LDAP που προκύπτει από κακή διαμόρφωση της υπηρεσίας. Ο LDAP server έχει παραμετροποιηθεί έτσι ώστε να επιτρέπει ανώνυμη πρόσβαση χωρίς έλεγχο ταυτότητας, να εκθέτει δεδομένα χρηστών μέσω ελλιπών κανόνων πρόσβασης (ACL) και να είναι ευάλωτος σε LDAP Injection, δίνοντας τη δυνατότητα σε έναν επιτιθέμενο να εξάγει περισσότερες πληροφορίες από αυτές που θα έπρεπε να είναι προσβάσιμες.

Στόχοι του Σενάριου:

Οι παίκτες πρέπει να ανιχνεύσουν την ύπαρξη μιας ευπαθούς υπηρεσίας LDAP μέσω εργαλείων δικτύου (π.χ. `nmap`). Στην συνέχεια πρέπει να κάνουν χρήση ανώνυμης πρόσβασης για την ανάκτηση δεδομένων χρηστών μέσω `ldapsearch` `ώστε να εκμεταλλευτεί την κακή διαμόρφωση που έχει γίνει και να ανακαλύψουν` δεδομένα όπως usernames και passwords, που δεν θα έπρεπε να είναι διαθέσιμα.

Αυτό θα βοηθήσει στην κατανόηση της ευπάθειας με χρήση ειδικών queries για πρόσβαση σε δεδομένα που προστατεύονται ανεπαρκώς.

LDAP Server:

Ένας OpenLDAP server παραμετροποιείται για να περιλαμβάνει ευάλωτα δεδομένα χρηστών. Οι κανόνες πρόσβασης (ACLs) διαμορφώνονται σκόπιμα με αδύναμες πολιτικές.

Infrastructure as Code (IaC):

- Χρησιμοποιείται για την αυτοματοποίηση της ανάπτυξης της εφαρμογής και του openldap server.
- Επιτρέπει την διαμόρφωση του server χωρίς admin password με αποτέλεσμα την δημιουργια ευπάθειας.
- Επιτρέπει την είσοδο δεδομένων στην βάση LDAP μέσω αρχείου LDIF.
- Ενεργοποιεί την ανώνυμη αναζήτηση.
- Database Server:
- Επανεκκίνηση της υπηρεσίας για την εκμετάλλευση μέσω της συνδεδεμένης εφαρμογής

Διεξαγωγή επίθεσης:

Ανακάλυψη Υπηρεσίας: Οι συμμετέχοντες εντοπίζουν την ανοιχτή θύρα 389 (LDAP) και καταλαβαίνουν ότι πρόκειται για μια υπηρεσία LDAP.

Ανώνυμη Σύνδεση: Η υπηρεσία LDAP επιτρέπει ανώνυμη πρόσβαση, παρέχοντας στον επιτιθέμενο τη δυνατότητα να εκτελέσει queries χωρίς κωδικούς πρόσβασης.

Εκτέλεση Queries: Οι συμμετέχοντες χρησιμοποιούν εργαλεία όπως `ldapsearch` για να εξάγουν δεδομένα. Για παράδειγμα:

Ανακάλυψη χρηστών (π.χ., `uid=john`, `uid=admin`).

Ανάκτηση ευαίσθητων δεδομένων, όπως ονόματα, emails και passwords που είναι αποθηκευμένα στον LDAP.

LDAP Injection: Οι συμμετέχοντες δοκιμάζουν queries με εισαγωγή φίλτρων, όπως:

`(uid=*)(objectClass=*)` για να εξάγουν όλους τους χρήστες.

Παράκαμψη φίλτρων αναζήτησης με την εισαγωγή ειδικών χαρακτήρων.

Εξαγωγή Ευαίσθητων Δεδομένων: Με σωστή εκμετάλλευση, οι συμμετέχοντες μπορούν να εξαγάγουν στοιχεία διαχειριστών ή άλλες χρήσιμες πληροφορίες που θα χρησιμοποιήσουν σε επόμενες φάσεις (π.χ., brute force).

Αποτέλεσμα:

Κατανόηση Κακής Διαμόρφωσης: Οι συμμετέχοντες μαθαίνουν πώς μικρά λάθη στις ρυθμίσεις (π.χ., ενεργοποίηση ανώνυμης πρόσβασης ή αδύναμοι ACLs) μπορούν να έχουν σοβαρές συνέπειες.

Χρήση LDAP Queries: Εκπαιδεύονται στη χρήση εργαλείων και μεθοδολογιών για την εκμετάλλευση ευπαθειών LDAP.

Ασφάλεια Υπηρεσιών: Διδάσκονται τρόπους προστασίας, όπως:

Απενεργοποίηση ανώνυμης πρόσβασης.

Ενίσχυση κανόνων ACL.

Χρήση ισχυρών πιστοποιήσεων.

Συνδυασμός της ευπάθειας LDAP με άλλες υπηρεσίες, όπως SSH, για να δημιουργηθεί μια πολυεπίπεδη επίθεση.

- Ενσωμάτωση μεθόδων brute force για αποκρυπτογράφηση credentials.
- Πρόσθεση layer monitoring, ώστε οι συμμετέχοντες να κατανοούν πώς μπορούν να ανιχνευθούν τέτοιες επιθέσεις από διαχειριστές συστημάτων.

4.5 Διεξαγωγή επίθεσης σε Juice-shop

- Injection – Admin Login

Η πρόκληση αφορά την εκμετάλλευση ενός ευάλωτου λογαριασμού διαχειριστή στην διαδικτυακή εφαρμογή juice-shop. Παρουσιάζονται διάφορες στρατηγικές επίθεσης, όπως η χρήση γνωστής διεύθυνσης email του διαχειριστή, η επίθεση με ειδικά μοτίβα ή η ανάλυση hash του κωδικού πρόσβασης. Υπάρχει επίσης η δυνατότητα επίλυσης της πρόκλησης μέσω συνδυαστικών

επιθέσεων χωρίς την ανάγκη για SQL Injection. Παρακάτω θα γίνει ανάλυση μια απλής μεθόδου injection καθώς είναι γρήγορη και θα βοηθήσει στην ευκολότερη εξαγωγή συμπερασμάτων.

Όπως αναφέρθηκε και νωρίτερα για το πως λειτουργούν τυπικά οι επιθέσεις SQL injection θα γίνει δοκιμή του παρακάτω snippet για τον έλεγχο παρουσίας ευπάθειας.

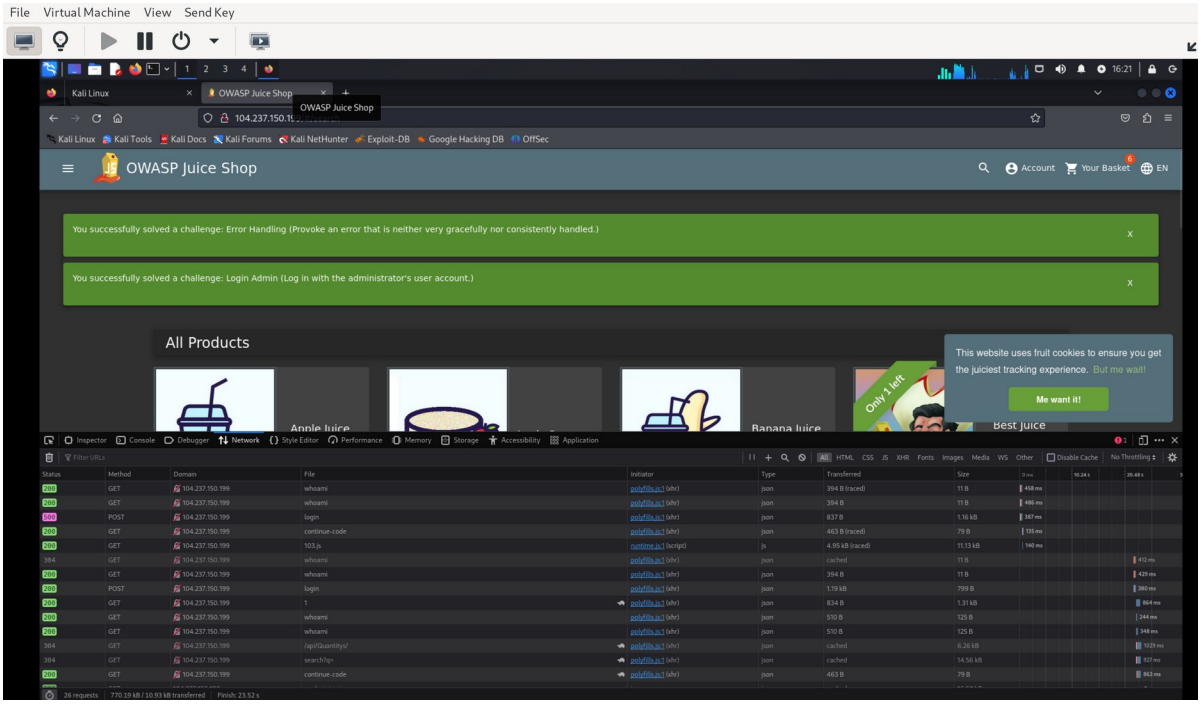
“ ‘ OR 1=1 -- “

Αν η σελίδα εισόδου είναι ευάλωτη σε SQL Injection, μπορεί να γίνει παράκαμψη στον έλεγχο ταυτότητας εκτελώντας κακόβουλες SQL εντολές.

Στο πεδίο του ονόματος χρήστη, ο επιτιθέμενος εισάγει το παραπάνω και έτσι το SQL ερώτημα γίνεται κάτι σαν:

```
“ SELECT * FROM users WHERE email = ' OR 1=1 -- ' AND password = '...';
“
```

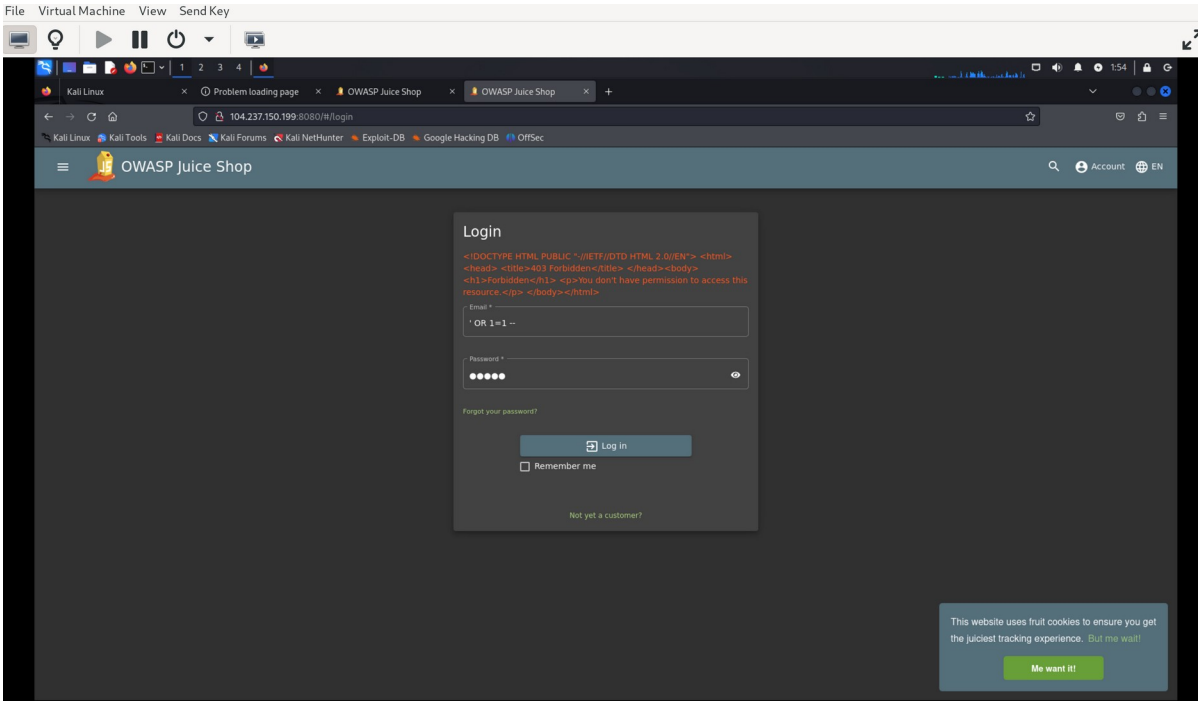
Το `'1'='1'` είναι πάντα αληθές, με αποτέλεσμα να επιτραπεί η πρόσβαση όπως φαίνεται και στην εικόνα παρακάτω.



Εικόνα 24. Επιτυχημένη σύνδεση μέσω sql injection

- Injection -Admin Login με ενεργοποιημένο το WAF

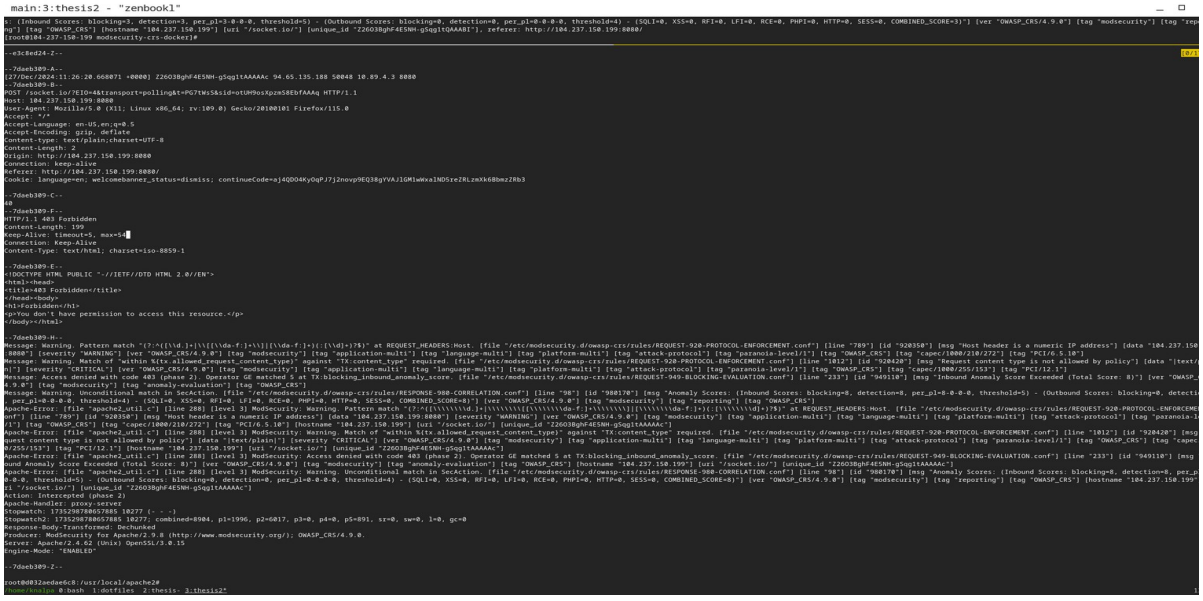
Εάν πραγματοποιηθεί η ίδια επίθεση με το modsecurity container ενεργοποιημένο, δηλαδή με το image του modsecurity apache να λειτουργεί ως reverse proxy και με το CRS φορτωμένο στο WAF θα παρατηρηθεί ότι η επίθεση δεν προχωράει. Στην εικονα 21 παρατηρείται το error που εμφανίζεται στην σελίδα του web application.



Εικόνα 25. Αποτυχημένη επίθεση στο login page

Στην συνέχεια μπορεί να γίνει σύνδεση στο container που τρέχει το modesecurity και να γίνει φιλτράρισμα και ανάλυση των logs που βρίσκοντα στο /var/log/modsec_audit.log αλλά και κάποιες επιπλέον πληροφορίες για το request στα error logs του apache2 στο /var/log/apache2/error.log .

Στην εικόνα 22 παρατίθεται το error στα logfiles του modsecurity.



Εικόνα 26. Modsecurity log files.

Από την ανάλυση παρατηρείται πως λειτουργεί το modesecurty βάση τα διάφορα βάρη που συνοδεύουν τα request τα οποία δέχεται το modsecurity και πως με αυτά αποφασίζει εάν θα κάνει block ή όχι την κίνηση.

Συγκεκριμένα στην εικόνα δείχνει ότι ένα αίτημα HTTP απορρίφθηκε από τον διακομιστή που χρησιμοποιεί ModSecurity με ενεργοποιημένο το OWASP Core Rule Set (CRS). Το αίτημα προήλθε από τη διεύθυνση IP `104.237.150.199` και ενεργοποίησε πολλούς κανόνες ασφαλείας λόγω παραβιάσεων πρωτοκόλλου, όπως η χρήση αριθμητικής IP στη κεφαλίδα `Host` και η έλλειψη της κεφαλίδας `Content-Type`. Το συνολικό σκορ ανωμαλιών έφτασε το 8, υπερβαίνοντας το αποδεκτό όριο του 4, με αποτέλεσμα την απόρριψη του αιτήματος με κωδικό 403 (Forbidden). Το αίτημα θεωρήθηκε πιθανόν κακόβουλο ή κακώς διαμορφωμένο. Συνιστάται παρακολούθηση της διεύθυνσης IP για επαναλαμβανόμενες επιθέσεις ή προσαρμογή των κανόνων για αξιόπιστα αιτήματα.

5. Συμπεράσματα

Απ' τα παραπάνω κεφάλαια και δοκιμές μπορούν να εξαχθούν κάποια συμπεράσματα. Αρχικά, όπως συζητήθηκε, η αυτοματοποίηση μπορεί ακόμα και να περιπλέξει την δημιουργία υποδομών εάν δεν γίνει η σωστή επιλογή τεχνολογίας. Για τον λόγω αυτό έγινε επιλογή terraform και Ansible για την σχεδίαση και υλοποίησης των υποδομών.

Με το terraform και τον linode provider γίνεται πολύ εύκολη η διαχείριση της υποδομής. Σε πολύ λίγες γραμμές κώδικα δηλώνονται τα μηχανήματα και το firewall που τα προστατεύει και με μια εντολή δημιουργείται η υποδομή μέσα σε λίγα λεπτά. Αυτό βοήθησε όταν κατά τις δοκιμές γινόταν εγκατάσταση πακέτων και χρειαζόταν να γίνει επαναφορά της υποδομής. Η διαδικασία ήταν η καταστροφή και η δημιουργία της απ' την αρχή με σκοπό την επίτευξη της αμεταβλητότητας.

Με παρόμοιο τρόπο το Ansible επιτρέπει ταχύτατη και συνεπή ρύθμιση των μηχανημάτων σε εξίσου γρήγορο χρόνο μετά την δημιουργία του κάθε μηχανήματος. Ως αποτέλεσμα, μια μικρή ομάδα ή ακόμα και ένας ερευνητής μόνος μπορεί να κάνει δοκιμές σε υποδομές και εφαρμογές πριν την χρήση τους. Στην περίπτωση που οι ανάγκες ήταν διαφορετικές, όπως για παράδειγμα περισσότερα μηχανήματα λόγω διαφορετικών δικτυακών αλλαγών αυτό θα ήταν επίσης εύκολο καθώς μέσω κώδικα δίνεται η δυνατότητα να γίνει αυτό με ελάχιστες αλλαγές.

Επιπλέον με την χρήση των container τεχνολογιών docker και podman οι οποίες υποστηρίζονται απ' το juice-shop και το modsecurity παρατηρείται ότι εφαρμογές που έχουν σχεδιαστεί με τις τεχνολογίες αυτές κατά νου επιτρέπουν την εύκολη υιοθέτηση τους με λίγες ή και καθόλου περαιτέρω ρυθμίσεις. Έτσι καταλήγουμε στο συμπέρασμα ότι αυτές οι τεχνολογίες με το modularity τους μπορούν να χρησιμοποιηθούν για την δημιουργία και μιας βασικής υποδομής αλλά και για την προσθήκη επιπλέον λειτουργιών σε μια υπάρχουσα. Από ένα απλό σύστημα σε ένα με επιπλέον δικλείδες ασφαλείας όπως προστασία από waf αλλά και μια απλή υποδομή να πάμε σε μια εφαρμογή η οποία να μοιράζει την κίνηση σε πολλούς server για την εξυπηρέτηση πολλαπλών χρηστών, συνδυάζοντας λοιπόν ασφάλεια και αποδοτικότητα.

Βιβλιογραφία

1. What is a web application firewall. Web application firewall – Wikipedia

2. What is a web application Web application – Wikipedia

3. What is Infrastructure as Code https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac - Redhat

4. What is a database Database – Wikipedia

5. What is a vulnerability Vulnerabilities | OWASP Foundation

6. OWASP https://owasp.org/ - OWASP

7. What is configuration management What is configuration management – Redhat

8. Ansible https://www.redhat.com/en/ansible-collaborative - Redhat

9. Chef Progress Chef – Wikipedia

10. Puppet Puppet (software) - Wikipedia

11. Terraform https://www.terraform.io/ - Terraform

12. Terraform modules https://developer.hashicorp.com/terraform/language/modules - Terraform

13. Pulumi Get Started with Pulumi | Pulumi Docs

14. KVM https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine - Wikipedia

15. VPC https://en.wikipedia.org/wiki/Virtual_private_cloud - Wikipedia

16. VPS https://en.wikipedia.org/wiki/Virtual_private_server - Wikipedia

17. Containers https://www.redhat.com/en/topics/containers – Redhat

18. Docker https://docs.docker.com/get-started/docker-overview/

19. Podman https://docs.podman.io/en/latest/

20 Docker compose https://docs.docker.com/compose/

21 Podman compose https://docs.podman.io/en/latest/markdown/podman-compose.1.html

22 Kubernetes https://kubernetes.io/

23 Terraform provider https://developer.hashicorp.com/terraform/language/providers - Hashicorp

24 Ansible role https://www.redhat.com/en/topics/automation/what-is-an-ansible-role - Redhat

25 Ansible provider for terraform https://registry.terraform.io/providers/ansible/ansible/latest/docs - Terraform

26 Vulnerabilities https://en.wikipedia.org/wiki/Vulnerability_(computer_security) - Wikipedia

27 XSS https://owasp.org/www-community/attacks/xss/ - OWASP

28 Injection https://owasp.org/www-community/Injection_Theory - OWASP

29 CSRF https://owasp.org/www-community/attacks/csrf - OWASP

30 Juice-shop https://owasp.org/www-project-juice-shop/ - OWASP

31 Multijuice https://github.com/juice-shop/multi-juicer - OWASP

32 Modsecurity https://modsecurity.org/ - Modsecurity

33 App protect https://www.f5.com/products/nginx/nginx-app-protect - F5

34 Openappsec https://www.openappsec.io/ - Openappsec

35 Coreruleset https://owasp.org/www-project-modsecurity-core-rule-set/ - OWASP

36 LDAP https://www.redhat.com/en/topics/security/what-is-ldap-authentication - Redhat

37 LDAP Injection https://owasp.org/www-community/attacks/LDAP_Injection - OWASP

Παραρτήματα

Github repository: https://github.com/konstantinos-null/thesis