



Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Διπλωματική Εργασία

Ασφάλεια σε Microservices: Μελέτη Περιπτώσης

Γεώργιος Λώλης
Α.Μ. 1054346

Επιβλέπων
Χρήστος Μακρής
Αναπληρωτής Καθηγητής

Συνεπιβλέποντας
Αριστείδης Ηλίας
ΕΔΙΠ

Πάτρα, 2023

Abstract

Στη σύγχρονη εποχή οι απαιτήσεις από τα ψηφιακά συστήματα που έχει επιτρέψει η τεχνολογική πρόοδος αυξάνονται διαρκώς. Μεγαλύτερα συστήματα, πολύπλοκες παροχές που πρέπει να είναι διαθέσιμες σε άμεσες ταχύτητες, πολλαπλοί χρήστες. Οι μονολιθικές αρχιτεκτονικές έχοντας υπηρετήσει τα μέγιστα των δυνατοτήτων τους, δίνουν την θέση τους σε αρχιτεκτονικές κατανεμημένης λογικής, όπως το Cloud και τα microservices.

Οι αρχιτεκτονικές αυτού του τύπου δίνουν την δυνατότητα διάσπασης του προβλήματος σε πολλά μικρότερα αυτόνομα τμήματα, παρέχοντας ευελιξία στους προγραμματιστές στην επιλογή τεχνολογιών και μεθόδων στην υλοποίηση, αλλά και τη δυνατότητα εύκολης κλιμάκωσης των εφαρμογών. Τα πολλαπλά πλεονεκτήματα των microservices τα έχουν καταστήσει μία από τις τεχνολογίες αιχμής του σήμερα, κερδίζοντας διαρκώς έδαφος στο πεδίο εφαρμογής. Ωστόσο, όπως κάθε νέα τεχνολογία, εμπεριέχει ανεξερεύνητα σημεία και μειονεκτήματα τα οποία δεν είναι ορατά από την αρχή, αλλά ανακύπτουν κατά την εφαρμογή.

Η ασφάλεια αυτών των αρχιτεκτονικών αποτελεί ένα από τα ανοιχτά προβλήματα που αντιμετωπίζουν διάφορες εφαρμογές, με διάφορες λύσεις να προτείνονται και διαφορετικές υλοποιήσεις να εφαρμόζονται ανά περίπτωση. Η παρακολούθηση ενός συστήματος αποτελεί καίριο κομμάτι της ασφάλειας για την ανίχνευση και αποτροπή επιθέσεων. Ωστόσο, η πολυπλοκότητα και η πολυμορφικότητα των microservices ανάγουν την εφαρμογή των συγκεκριμένων μεθόδων σε ένα πολύπλοκο, δύσκολο πρόβλημα.

Η παρούσα διπλωματική αποτελεί μία προσπάθεια καταγραφής του προβλήματος του security monitoring στις αρχιτεκτονικές microservices, των λύσεων και των μεθοδολογιών που προτείνονται στην ακαδημαϊκή βιβλιογραφία και στον κόσμο της βιομηχανίας. Σε ποιο σημείο βρίσκεται η έρευνα σήμερα, ποια είναι αυτά που πρέπει να γνωρίζει ένας προγραμματιστής και ποια κομμάτια παραμένουν ανεξερεύνητα; Σε αυτά και άλλα ερωτήματα προσπαθούν να απαντήσουν οι επόμενες σελίδες.

Περιεχόμενα

Abstract	i
Περιεχόμενα	i
1	1
<i>Εισαγωγή</i>	<i>1</i>
1.1 Σημασία του Προβλήματος	1
1.2 Συνεισφορά	2
1.3 Διάρθρωση	3
2	5
<i>Microservices: Ορισμός και Ιστορία</i>	<i>5</i>
2.1 Ιστορικά Στοιχεία	5
2.2 Ορισμός Αρχιτεκτονικής Microservices	8
2.3 Συγκριτικά Παραδείγματα Μονολιθικών Εφαρμογών και Microservices	12
3	17
<i>Security των Microservices</i>	<i>17</i>
3.1 Διαφοροποιήσεις αρχιτεκτονικών microservices	18
3.2 Επίπεδα ασφάλειας και ευάλωτα σημεία των Microservices	21
3.3 Αποτύπωση της ασφάλειας των microservices στην βιβλιογραφία	34
4	37
<i>Συστήματα Monitoring στην τεχνολογία των Microservices</i>	<i>37</i>
4.1 Εισαγωγή στην διαδικασία του monitoring	38
4.2 Monitoring και Microservices	44
4.2.1. Βασική Μεθοδολογία	46
4.2.2. Περιοχές Ενδιαφέροντος	52
4.2.3. Παραδείγματα Συστημάτων	53
4.3 Σημαντικότερες Τεχνολογίες Monitoring για την αρχιτεκτονική Microservices	61
5	67
<i>Προτάσεις για Security Monitoring στην αρχιτεκτονική Microservices</i>	<i>67</i>
5.1 Μεθοδολογία	69
5.2 Θεωρητικό πλαίσιο Security Monitoring	73
5.2.1. Μεθοδολογίες monitoring που εφαρμόζονται για σκοπούς ασφαλείας στις αρχιτεκτονικές microservices	73
5.3 Εφαρμογές συστημάτων monitoring σε microservices υπό το πρίσμα της ασφάλειας	79

6	87
<i>Συμπεράσματα και Συζήτηση</i>	87
<i>Βιβλιογραφία - Αναφορές</i>	91
<i>Εικόνες</i>	94
<i>Πίνακες.....</i>	97

1

Εισαγωγή

1.1 Σημασία του Προβλήματος

Η τεχνολογία των microservices αποτελεί μία από τις πιο ταχύτατα διαδεδομένες αρχιτεκτονικές στον σχεδιασμό λογισμικού σήμερα. Η ευελιξία στην χρήση πόρων και διαφορετικών τεχνολογιών, η προσαρμοστικότητα σε πρότζεκτ μεγάλης κλίμακας και η ευκολία κατακερματισμού των προβλημάτων είναι κάποια από τα χαρακτηριστικά που κατατάσσουν την λύση των microservices μία από τις πιο δημοφιλείς επιλογές σήμερα. Οι σύγχρονες απαιτήσεις σε ταχύτητα, ακρίβεια και κλίμακα οδήγησαν λίγα χρόνια πριν στην επαναεμφάνιση της λύσης των κατακερματισμένων συστημάτων, τα οποία σταδιακά εξελίχθηκαν σε πιο συγκεκριμένες λύσεις.

Ως μια από τις πιο καινούριες λύσεις, η αρχιτεκτονική των microservices, παρόλο που βρίσκεται πολύ γρήγορα εφαρμογή σε διάφορες περιπτώσεις, εξακολουθεί να εμπεριέχει ανεξερεύνητες περιοχές και αναπάντητα ερωτήματα. Ένα από τα μεγαλύτερα ζητήματα γύρω από αυτήν την τεχνολογία είναι ο επαναορισμός της ασφάλειας και των αντίστοιχων μεθόδων και τεχνικών, που πλέον έχουν μια πολύ πιο περίπλοκη πρόκληση σε σύγκριση με τις κλασικές μονολιθικές υλοποιήσεις, ιδιαίτερα μέσα στο δυναμικό πλαίσιο του διαδικτύου και του cloud. Ο τομέας της ασφάλειας των microservices

αποτελεί έναν ακόμα υπό διαμόρφωση τομέα, καθώς ακαδημαϊκά δεν έχει προκύψει μία σαφής μεθοδολογία, αλλά υπάρχουν παραδείγματα διάφορων λύσεων όπου ακόμα εξελίσσονται.

Ο συνδυασμός της λογικής του monitoring πάνω σε θέματα ασφάλειας μπορεί να δείχνει έναν δρόμο στον οποίο δίπλα στην παρακολούθηση της απόδοσης και της σωστής λειτουργίας, έρχεται να προστεθεί και η διαρκής παρακολούθηση της υγείας των εφαρμογών. Ως δύο σύγχρονες έννοιες και τομείς, το monitoring και το security αποτελούν μια διαρκής πρόκληση σήμερα στις περισσότερες εφαρμογές. Η σημασία του προβλήματος γίνεται λοιπόν σύγχρονη, μεγάλη και επιτακτική, καθώς η ταχύτητα προτίμησης της συγκεκριμένης τεχνολογίας είναι γεγονός, και τα ζητήματα ασφάλειας και παρακολούθησης είναι βαρύνουσας σημασίας και λόγω νομικού πλαισίου (GDPR), και για την ασφάλεια και εξυπηρέτηση του χρήστη στην εκάστοτε περίπτωση.

1.2 Συνεισφορά

Σκοπός της συγκεκριμένης διπλωματικής είναι η ανάδειξη μέσω της βιβλιογραφίας του προβλήματος της ασφάλειας στην αρχιτεκτονική των microservices, και η συλλογή και παράθεση λύσεων μέσω της μεθοδολογίας του monitoring για την διαρκή παρακολούθηση της υγείας των microservices. Στο πρώτο σκέλος, απαραίτητος είναι ο ορισμός της ασφάλειας, η ανάδειξη των ζητημάτων ασφάλειας και των σημείων που χρήζουν προσοχής και τέλος των διάφορων τεχνικών που χρησιμοποιούνται ή προτείνονται για τον σχεδιασμό ασφάλειας στην συγκεκριμένη αρχιτεκτονική. Στο δεύτερο σκέλος, αναδεικνύεται το πώς οι διαδικασίες monitoring μπορούν επιπρόσθετα να οργανώσουν την ασφάλεια και να δίνουν μια διαρκή εικόνα της κατάστασης υγείας και ασφάλειας της εφαρμογής, καθώς, σε κάθε περίπτωση, η διασφάλιση ορισμένων ευάλωτων σημείων σε μία εφαρμογή δεν συνεπάγεται την διαρκή της προστασία. Νέοι κίνδυνοι και κενά ανακύπτουν και νέες επιθέσεις καταγράφονται καθημερινά. Η παρακολούθηση λοιπόν της λειτουργίας, μπορεί να οδηγήσει στην παρακολούθηση της κατάστασης της ασφάλειας.

Η συγκριτική μελέτη και παράθεση διάφορων προσπαθειών της παρακολούθησης της κατάστασης ασφάλειας εφαρμογών, μπορεί να αποτελέσει εφόδιο για τους προγραμματιστές και τους σχεδιαστές λογισμικού του σήμερα. Μία τέτοια μελέτη καθιστά δυνατό το να γίνουν ευδιάκριτα τα κενά, οι τάσεις, οι επικρατούσες μέθοδοι και οι απαιτήσεις για τις μελλοντικές υλοποιήσεις και έρευνα. Υποδεικνύει επίσης τα σημεία στα οποία θα έπρεπε βάσει σχεδιασμού να διατίθενται τα δεδομένα που είναι απαραίτητα για τον διαρκή έλεγχο της υγείας της εφαρμογής.

1.3 Διάρθρωση

Η παρούσα εργασία δομείται ως εξής. Αρχικά, στο δεύτερο κεφάλαιο, γίνεται μια εισαγωγή στην τεχνολογία των microservices, της ιστορικής αναγκαιότητας που ανέδειξε την χρήση τους, των βασικών χαρακτηριστικών τους και των διαφοροποιήσεών τους σε σύγκριση με τις κλασικές μονολιθικές υλοποιήσεις και σχεδιασμούς. Σκοπός του κεφαλαίου είναι να δώσει τις απαραίτητες πληροφορίες ώστε στη συνέχεια να γίνουν κατανοητά τα ζητήματα ασφάλειας και το πώς αποτελούν μια νέα πρόκληση/πρόβλημα το οποίο δεν υπήρχε με αυτή τη μορφή στις προηγούμενες αρχιτεκτονικές που σιγά σιγά σε πολλές περιπτώσεις αντικαθίστανται.

Στη συνέχεια, στο τρίτο κεφάλαιο ορίζεται το πρόβλημα της ασφάλειας στην τεχνολογία των microservices. Αναζητούνται οι διαφοροποιήσεις στην ασφάλεια που φέρει η νέα αυτή αρχιτεκτονική: ποια ζητήματα παραμένουν ίδια, ποια αλλάζουν, και ποια είναι τα καινούρια που προκύπτουν λόγω της διαφορετικής λογικής σχεδιασμού. Πώς τα ευάλωτα σημεία αλλάζουν και προκύπτουν, πώς οι παράμετροι ασφαλείας μετατίθενται και πώς νέες παράμετροι εμφανίζονται.

Στο τέταρτο κεφάλαιο γίνεται μια προσπάθεια ανάδειξης της σημασίας του monitoring στα πλαίσια της ασφάλειας. Η τοποθέτηση alerts, τα δυναμικά online dashboards ασφάλειας, η παρακολούθηση δεικτών, η διαρκής διενέργηση ελέγχων και

παρακολούθηση των αποτελεσμάτων τους είναι από τις τεχνικές που θα παρουσιαστούν και θα συνδεθούν με την ασφάλεια των microservices.

Τελειώνοντας την εισαγωγή στο κομμάτι του monitoring για λόγους ασφαλείας, το πέμπτο κεφάλαιο στοχεύει στην παράθεση συνδυαστικών λύσεων και προτάσεων από την βιβλιογραφία. Μέσω αυτής της έρευνας, σκοπός είναι να ανιχνευτούν τεχνικές οι οποίες έχουν την τάση να χρησιμοποιούνται, να συγκριθούν διαφορετικές λύσεις, και τέλος, συνδυαστικά με την βιβλιογραφική έρευνα στα ζητήματα ασφαλείας των microservices, να εντοπιστούν τα προβλήματα τα οποία κυριαρχούν και αυτά τα οποία δεν εξετάζονται στον ίδιο βαθμό. Τέλος, στο έκτο κεφάλαιο, παρατίθεται μια σειρά από συμπεράσματα, παρατηρήσεις και προτάσεις για μελλοντική έρευνα και σκέψεις που προέκυψαν κατά την εκπόνηση της έρευνας.

2

Microservices: Ορισμός και Ιστορία

Σε αυτό το κεφάλαιο θα γίνει μια προσπάθεια να παρουσιαστεί η τεχνολογία των microservices. Ξεκινώντας με την αναγκαιότητα που οδήγησε στην σύλληψη της ιδέας της συγκεκριμένης τεχνολογίας και την μικρή σύγχρονη ιστορία τους, μέχρι τις τεχνικές λεπτομέρειες και τις σύγχρονες εφαρμογές τους, σκοπός του κεφαλαίου είναι να θέσει τα θεμέλια ώστε στη συνέχεια να μπορέσει να αναδειχθεί το πρόβλημα της ασφάλειας και η αναγκαιότητα της παρακολούθησης (monitoring) των microservices. Τέλος, θα παρουσιαστούν δύο παραδείγματα αρχιτεκτονικών microservices, ένα από τον χώρο των επιχειρήσεων και ένα από τον ακαδημαϊκό χώρο, ώστε να γίνει πιο κατανοητή η ιδέα της συγκεκριμένης τεχνολογίας.

2.1 Ιστορικά Στοιχεία

Ξεκινώντας με το πώς προέκυψε η τεχνολογία των microservices, σε συνέντευξη του ο γνωστός Software Architect James Lewis [1] υποδεικνύει κάποια από τα σημεία που οδήγησαν τους σύγχρονους σχεδιαστές και αρχιτέκτονες λογισμικού σε μια μη μονολιθική λύση. Χαρακτηριστικά αναφέρεται ότι η διαρκής ανάπτυξη λογισμικού στις web εφαρμογές και οι αυξανόμενες απαιτήσεις σε πολυπλοκότητα, λειτουργικότητα και χρόνους απόκρισης έχει οδηγήσει σε εφαρμογές μεγάλου μεγέθους. Όταν αυτό

αποτελεί πια κανονικότητα, η διαχείριση μιας μονοκόμματης εφαρμογής τελικά αρχίζει να γίνεται ένα δύσκολο έργο. Η προσθήκη νέων λειτουργικοτήτων, η αναβάθμιση του κώδικα, και η προσαρμογή του στην εκάστοτε συνθήκη δυσχεραίνεται όσο το μέγεθος αυξάνεται. Για κάθε αλλαγή, οποιασδήποτε έκτασης, ολόκληρος ο κώδικας πρέπει να ελεγχθεί, και ενδεχομένως να επηρεαστούν σημεία τα οποία δεν θα ήθελε ο προγραμματιστής να αλλάζουν. Επίσης, απαιτείται σε έναν βαθμό η αντίληψη του συνολικού λογισμικού, κάτι το οποίο ορισμένες φορές είναι αδύνατο. Τέλος, στην εποχή της διαρκούς ανάπτυξης στον κλάδο των υπολογιστών και των εφαρμογών, εφαρμογές τέτοιου είδους αρχιτεκτονικής, αδυνατούν να “προσαρμοστούν” εύκολα σε έργα και απαιτήσεις μεγαλύτερης κλίμακας (scalability issue).

Πρώιμες προσπάθειες για ανάπτυξη μικρής κλίμακας, ευέλικτων, αυτοτελών services απαντώνται από το 1997 με την εισαγωγή του Enterprise Java Bean (EJB) από την IBM, παρά τα προβλήματα τα οποία ακόμα δεν είχαν βρει λύσεις [2]. Επόμενη σοβαρή προσπάθεια εντοπίζεται από την Microsoft το 1995, με την εισαγωγή στην κυκλοφορία του Simple Object Access Protocol (SOAP), ενώ κάποια χρόνια αυτή η προσέγγιση εγκαταλείφθηκε λόγω των προϋποθέσεων που απαιτούνταν, και υπήρξε μεταστροφή στην Representational State Transfer (REST) προσέγγιση το 2008. Το 2011-2012 ο όρος microservices πια οικειοποιείται και εγκαθιδρύεται [2]. Στο σήμερα, βαρυσήμαντα παραδείγματα όπως αυτά του Docker Swarm και του Netflix [3], δείχνουν πώς η τεχνολογία των microservices αποτελεί πια παρόν, και βρίσκεται σίγουρα στο άμεσο μέλλον.

Σαν σχόλια και αρχικές ιδέες σύλληψης της ιδέας των microservices συναντάει κανείς την λογική των UNIX, ‘κάνε ένα πράγμα και κάν’ το καλά’ [3], αλλά και άλλα όπως ‘μην εμπιστεύεσαι κανέναν’ [3], ‘κάν’ το όσο πιο απλό γίνεται’ [2].

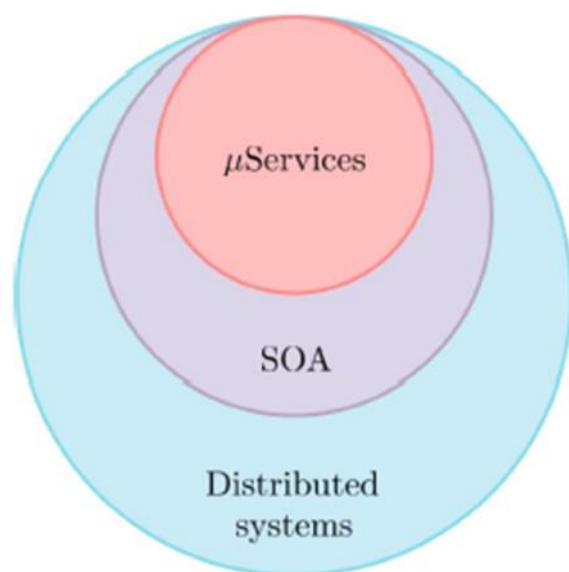


Fig. 1. Microservice architecture in perspective. Microservices are an implementation approach to SOA. SOA is a subclass of distributed systems.

Εικόνα 2.1.: Τεχνολογία microservices ως υποσύνολο σχεδιαστικών λύσεων. [3]

Στην όψη της συνεχούς ανάκυψης νέων απαιτήσεων οι οποίες έφερναν μαζί τους τις παραπάνω δυσκολίες, οι αρχιτέκτονες λογισμικού σταδιακά αντιλαμβάνονταν και αντιλαμβάνονται την ανάγκη μιας αρχιτεκτονικής όχι τόσο μονοκόμματης, όχι τόσο μονολιθικής. Επιπλέον, η χρήση διάφορων γλωσσών προγραμματισμού και τεχνολογιών – με διαφορετική εξειδίκευση και των προγραμματιστών αλλά και των ίδιων των τεχνολογιών ανά περίπτωση, και η αδυναμία να συνδυαστούν αυτά εύκολα σε ένα μοναδικό σύστημα. Σε επέκταση της λογικής των χωριστών μπλοκ κώδικα στο επίπεδο της αρχιτεκτονικής, η ιδέα των κατανεμημένων συστημάτων αρχίζει να επανέρχεται οδηγώντας στη λογική της SOA τεχνολογίας και τελικά στα microservices. Στην Εικόνα 1.1. παρατηρείται το που τελικά τοποθετείται η τεχνολογία των microservices σε σχέση με τις παραπάνω τεχνολογίες αρχιτεκτονικής.

2.2 Ορισμός Αρχιτεκτονικής Microservices

Για να οριστεί ορθά και ξεκάθαρα η διαφορετικότητα των microservices απαραίτητο είναι να αποσαφηνιστούν οι έννοιες κατανεμημένα συστήματα και SOA αρχιτεκτονικές. Αρχικά, ο όρος μονολιθική εφαρμογή σημαίνει ότι όλη η υλοποίηση στεγάζεται κάτω από ένα σύστημα. Ο κώδικας εσωτερικά μπορεί να είναι γραμμένος και οργανωμένος σε μικρότερα, αυτόνομα, εξατομικευμένα (modular) αλλά αλληλένδετα κομμάτια, τα οποία αποτυγχάνουν συνολικά. Η ιδέα των κατανεμημένων συστημάτων “σπάει” τις εν δυνάμει αυτόνομες λειτουργίες σε διαφορετικά συστήματα, ώστε να γίνεται πιο ξεκάθαρος ο σκοπός του κάθε συστήματος. Η ιδέα των συστημάτων-σταθμών εργασίας (nodes) υπόσχεται τα εξής εντός του πλέγματός τους [3]:

1. Κάθε σύστημα γνωρίζει μόνο τη δική του κατάσταση, και όχι τη συνολική κατάσταση του συνολικού συστήματος.
2. Οι εξατομικευμένοι κόμβοι λαμβάνουν αποφάσεις βάσει και έχουν πρόσβαση μόνο σε τοπικά διαθέσιμες πληροφορίες (πληροφορίες δηλαδή προς στις οποίες είναι εξουσιοδοτημένος ο κόμβος ή εισέρχονται από διασυνδεδεμένους κόμβους).
3. Η αρχιτεκτονική του συνολικού συστήματος πρέπει να είναι τέτοια ώστε η αποτυχία ή καταστροφή ενός κόμβου να μην επηρεάζει τους υπόλοιπους κόμβους και την λειτουργία τους [8] (σύγχρονα παραδείγματα επιτρέπουν την αντικατάσταση ενός κόμβου σε τέτοια περίπτωση από κάποιον κλώνο του, ή την αυτόνομη επανεκκίνηση-θεραπεία του πληγέντα σταθμού εργασίας).

Ενώ τα κατανεμημένα συστήματα επιτρέπουν την αυτονομία και την διαφορετικότητα στην επιλογή τεχνολογιών και λογικής ανάπτυξης, στα μειονεκτήματά τους μπορούν να αναφερθούν η μεγαλύτερη πολυπλοκότητα στην αρχική σχεδίαση, η δυσκολία στην διαδικασία του debugging και στην διαδικασία του ελέγχου και της συνολικής παρακολούθησης (monitoring) [3].

Αξίζει να αναφερθεί εδώ, ότι δίπλα στις έννοιες κατακευμασμένα ρυστήματα και microservices πολύ ρυχνά απαντώνται οι έννοιες του Cloud και των Containers, όπου οι αρχές ρχεδιασμού αρχιτεκτονικής ρυμπίπτουν: υπηρεσιογκεντρικός ρχεδιασμός (SOA, ακριβώς παρακάτω), εικονική υλοποίηση (virtualization), αβεβαιότητα-μη ρταθερότητα με την έννοια της αυτονομίας, της πολυρρησίας, της κλίμακας και της διαφορετικότητας αναφορικά με τις τεχνολογίες που ρρησιμοποιούνται τα οποία εισάγουν μια έννοια αβεβαιότητας, και η προσαρμοστικότητα που μπορούν να έχουν τα αυτοτελή, αυτόνομα microservices [4]. Ωστόσο, δεν αναφέρονται ρτο ίδιο πράγμα, παρόλο που οι έννοιες ρυχνά ρυγγέονται και ρρησιμοποιούνται ορισμένες φορές για να περιγράψουν μία από τις καταστάσεις, ή και μια ενδιαμέρρη κατάσταση. Σημαντικός είναι ο διαρρωρισμός των εννοιών: τα microservices αποτελούν μία προσέγγιση ρτην εφαρμογή της λογικής του Cloud και την υλοποίησή του, ενώ οι Containers αποτελούν έναν τρόπο-μέρσο εφαρμογής της αρχιτεκτονικής λογικής των microservices.

Σε μια πιο ρυγκεκριμένη προσέγγιση, η Αρχιτεκτονική Προσανατολισμένη ρτις Υπηρεσίες, SOA (Service-Oriented Architecture) επιδιώκει τον ρχεδιασμό των ρυστημάτων γύρω από τις επιχειρησιακές υπηρεσίες που πρέπει να παρέχει μια εφαρμογή. Κάθε υπηρεσία-ρύστημα μπορεί να είναι διαφορετική, να έχει τα δικά της ετερογενή ραρακτηριστικά, να είναι αυτόνομη, να ανήκει σε διαφορετικούς ρειριστές, και όλα μαζί να ρυντελούν ρτη δημιουργία ενός μεγαλύτερου ρυστήματος. Στο άρθρο [3], όπου γίνονται ραφής και οι διαφορές μεταξύ SOA και microservices, διαγράφονται οι βασικές υποθέσεις της SOA αρχιτεκτονικής:

- Υπηρεσίες - Services: Στα πλαίσια της SOA αρχιτεκτονικής μια υπηρεσία-service αποτελεί μια πράγματωση μίας επιχειρησιακής λειτουργικότητας αυτοτελής σε ένα σημείο. Κάθε υπηρεσία, πέρα από αυτοτελής, μπορεί και ρυγκεντρώνει και τα παρακάτω ραρακτηριστικά: αυτονομία, ανιχνεύσιμη/ορατή (με εφαρμογή διάφορων κανόνων), επαναρρησιμοποιήσιμες, όχι ρταθερή κατάσταση (state) ανά κλήρη, ρταθερή λειτουργικότητα, ρύνθετο, προσαρμύσιμο ανά κλήρη.

- Διαλειτουργικότητα – Interoperability: Η συγκεκριμένη υπόθεση έγκειται στην δυνατότητα κάθε service να μπορεί να τρέχει και να λειτουργεί σε διαφορετικές πλατφόρμες. Αποτελεί μια από της πιο σημαντικές αρχές της SOA αρχιτεκτονικής, επιτρέποντας την πολλαπλότητα, την αυτονομία και την ετερογενή ανάπτυξη των διάφορων services.
- “Χαλαρή” σύνδεση - Loose coupling: Η αρχή της χαλαρής σύνδεσης των κόμβων – που έρχεται μαζί με την έννοια της αυτονομίας – υπονοεί την μη εξάρτηση μεταξύ τους. Κάτι τέτοιο είναι απαραίτητο ώστε να μπορεί να επιτευχθεί η ευελιξία που υπόσχεται η SOA αρχιτεκτονική, η εύκολη προσαρμογή σε μεγαλύτερης κλίμακας απαιτήσεις (scalability), και η ανοχή στα σφάλματα, στις περιπτώσεις που ένας κόμβος αποτύχει ή καταστραφεί. Στόχος αυτής της αρχής είναι να μην υπάρχει αλληλοεξάρτηση μεταξύ των κόμβων, ώστε και οι αλλαγές να μπορούν να γίνονται μεμονωμένα σε κάθε service, χωρίς να επηρεάζονται τα υπόλοιπα, είτε επικοινωνούν με τον συγκεκριμένο κόμβο είτε όχι.

Γίνεται διακριτό το πώς οι SOA αρχιτεκτονικές αποτελούν μια συγκεκριμένη περίπτωση των κατανεμημένων συστημάτων, καθώς εστιάζουν κυρίως σε μία αφήγηση που ανακύπτει από τελικές λειτουργικότητες σχετικές με το επιχειρησιακό πλαίσιο της εφαρμογής. Σε σύγκριση με την αρχιτεκτονική των microservices, πολλές φορές όπως αναφέρεται και στα [1,3] υπάρχει μία σύγχυση στο σε ποια σημεία διαφοροποιείται η τελευταία. Όπως εύστοχα αποδίδεται στην Εικόνα 1.1., τα microservices αποτελούν μία περίπτωση εφαρμογής της λογικής της SOA.

Προχωρώντας στον προσδιορισμό και την διαφοροποίηση των microservices, απαντώνται διάφοροι ορισμοί. Παρακάτω παρατίθενται δύο από αυτούς [3]:

- **Ορισμός του Newman:** *Μικρά αυτόνομα services-υπηρεσίες τα οποία υλοποιούνται και πραγματώνονται ακολουθώντας τις παρακάτω αρχές: οι υπηρεσίες χτίζονται γύρω*

από επιχειρησιακά σενάρια-υποθέσεις, υιοθετώντας μια κουλτούρα αυτοματισμού, αποκρύπτοντας τις εσωτερικές λεπτομέρειες υλοποίησης, σε μια αποκεντρωμένη λογική, απομονώνοντας κάθε περίπτωση αποτυχίας ή αστοχίας. Επιπλέον, κάθε service πρέπει να μπορεί να αναπτυχθεί ανεξάρτητα από τα υπόλοιπα και να είναι εύκολα ορατό/αναγνωρίσιμο από τα υπόλοιπα. [3, ελεύθερη μετάφραση]

- Ορισμός των Lewis και Fowler: Η προσέγγιση ανάπτυξης μιας μοναδικής εφαρμογής ως μιας ακολουθίας μικρών services-υπηρεσιών, κάθε μία από τις οποίες μπορεί να τρέχει τις δικές της διαδικασίες και να επικοινωνεί με “ελαφρούς” μηχανισμούς, συνήθως μέσω HTTP resource API. Αυτές οι services-υπηρεσίες αναπτύσσονται γύρω από επιχειρησιακές δυνατότητες και λειτουργούν ανεξάρτητα από πλήρως αυτοματοποιημένους μηχανισμούς υλοποίησης. Υπάρχει μια ελάχιστη κεντρική διαχείριση αυτών των services-υπηρεσιών, οι οποίες μπορούν να είναι γραμμένες σε διαφορετικές γλώσσες προγραμματισμού και να χρησιμοποιούν διαφορετικές τεχνολογίες αποθήκευσης δεδομένων. [3, ελεύθερη μετάφραση]

Και στους δύο ορισμούς διαφαίνονται τα βασικά χαρακτηριστικά και δυνατότητες των microservices και η ουσία, η νοοτροπία της λογικής αυτής της αρχιτεκτονικής. Τα διακριτά τους χαρακτηριστικά όπως προκύπτει από τους ορισμούς και τις διαφοροποιήσεις τους με τους προδρόμους τους και τις ευρύτερες κατηγορίες στις οποίες ανήκουν όπως το SOA και τα κατανεμημένα συστήματα είναι τα παρακάτω¹ [3]:

Distributed composition: Τα microservices χτίζονται και υλοποιούνται βασισμένα σε άλλα microservices και επικοινωνούν μέσω του διαδικτύου.

Modularity: Τα microservices τείνουν να προσφέρουν μικρότερης λεπτομέρειας-κλίμακας APIs, κάτι το οποίο τα οδηγεί σε ευέλικτη επαναχρησιμοποίηση.

Encapsulation: Τα services-υπηρεσίες είναι σε μεγάλο βαθμό

¹ Οι όροι δεν μεταφράζονται εδώ, για αποφυγή λανθασμένης απόδοσης των ιδιοτήτων. Το νόημα των όρων στα Ελληνικά προκύπτει από την περιφραστική τους περιγραφή.

ενθυλακωμένα και απομονωμένα από τα υπόλοιπα services-υπηρεσίες, και μπορεί να είναι γραμμένα σε διαφορετικές γλώσσες προγραμματισμού, χωρίς να επηρεάζουν οι τεχνολογικές αποφάσεις σχετικά με κάποιο άλλο service-υπηρεσία.

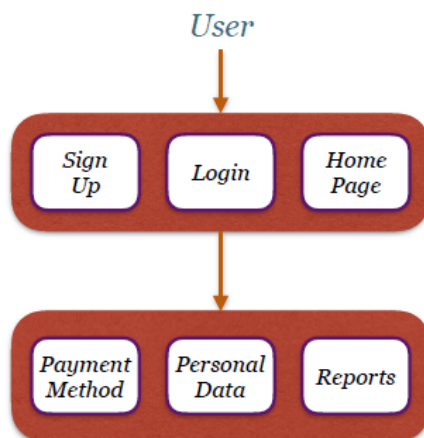
Network service: Τα services-υπηρεσίες είναι προσβάσιμα από το διαδίκτυο, και η επαναχρησιμοποίησή τους συμβαίνει με την επικοινωνία με το εκάστοτε service-υπηρεσία, παρά με την εγκατάσταση βιβλιοθηκών και σύνδεση με αυτές (σε αντίθεση με την κλασική προσέγγιση σχεδιασμού του modular design).

2.3 Συγκριτικά Παραδείγματα Μονολιθικών Εφαρμογών και Microservices

Αφού έχει οριστεί η τεχνολογία των microservices, και έχοντας γίνει σαφείς οι διαφοροποιήσεις από άλλες τεχνολογίες και έννοιες, παρακάτω παρουσιάζονται δύο σχηματικά παραδείγματα εφαρμογής τους σε αντιδιαστολή με την μονολιθική προσέγγιση. Το ένα παράδειγμα αφορά ένα σύστημα online πληρωμών και το άλλο την εφαρμογή ενός ψηφιακού συστήματος διαχείρισης μιας κλινικής ζώων (Spring PetClinic, μιας Java εφαρμογής ανοιχτού κώδικα που δημιουργήθηκε και διατίθεται για σκοπούς έρευνας και παρουσίασης λύσεων-demonstration).

Σε μία παρουσίαση της εταιρίας Battery Ventures (BV) για την σχεδιαστική λύση των microservices [5] παρουσιάζεται μια σχηματική απλοϊκή συγκριτική απεικόνιση μεταξύ της μονολιθικής αρχιτεκτονικής και των microservices. Στην Εικόνα 1.2. φαίνεται πως με την μονολιθική αρχιτεκτονική οι διαδικασίες της εισόδου στο σύστημα και των ενεργειών εντός του συστήματος είναι αλληλοεξαρτόμενες, γραμμικά συνδεδεμένες,

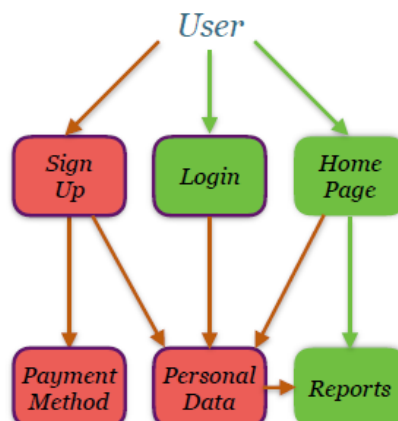
και ανήκουν συνολικά σε συγκεκριμένες ομάδες (teams-owners), Δεν υπάρχει ευελιξία, και η εφαρμογή λειτουργεί μονοκόμματα και γραμμικά. Επίσης, μέσω των πλαισίων, διευκρινίζεται το πώς οι συγκεκριμένες λειτουργικότητες-υπηρεσίες εξαρτώνται η μία από την άλλη, και αν αποτυγχάνει η μία, αποτυγχάνουν τελικά όλες.



Εικόνα 2.2.: Η μονολιθική προσέγγιση ενός συστήματος online πληρωμών. [5]

Στον αντίποδα, παρουσιάζεται η ευέλικτη λύση των microservices, όπου κάθε λειτουργικότητα είναι απομονωμένη, αυτοτελής και αυτόνομη. Επίσης, το πώς μπορούν να επικοινωνούν οι υπηρεσίες-services μεταξύ τους με πιο ελεύθερο τρόπο. Παραδείγματος χάριν, αν η λειτουργικότητα “SignUp” για κάποιο λόγο βρίσκεται εκτός λειτουργίας, ένας χρήστης που θέλει απλά να εισέλθει στην εφαρμογή, δεν αντιμετωπίζει ούτε αντιλαμβάνεται κάποιο πρόβλημα.

Τέλος, μέσω του χρωματικού κώδικα, διακρίνεται και ένα άλλο είδος ευελιξίας: οι ομάδες προγραμματισμού και ανάπτυξης λογισμικού έχουν την ευχέρεια να έχουν υπό την ιδιοκτησία (ownership) και έλεγχο τους αντίστοιχα microservices με την εξειδίκευση της ομάδας. Στην συγκεκριμένη περίπτωση, τα microservices με το κόκκινο χρώμα απαιτούν αποθήκευση και διαχείριση των δεδομένων και περισσότερους ελέγχους σχετικά με την ασφάλεια. Αντίστοιχα, αυτά με το πράσινο χρώμα εστιάζουν περισσότερο στην εμπειρία του χρήστη και μπορούν να παράγουν περισσότερα σενάρια χρήσης.



Εικόνα 2.3.: Η προσέγγιση της αρχιτεκτονικής microservices του αντίστοιχου με την Εικόνα 1.2. συστήματος online πληρωμών. [5]

Το επόμενο παράδειγμα, αφορά μια υλοποίηση από τον ακαδημαϊκό χώρο, για την έρευνα σε σχέση με την εφαρμογή αρχών ασφαλείας (security) στα microservices [6]. Στο κομμάτι της ασφάλειας θα αναλωθεί το επόμενο κεφάλαιο, στο συγκεκριμένο σημείο απλά επιχειρείται να γίνει διακριτή η διαφοροποίηση στην αρχιτεκτονική.

Στην μονολιθική προσέγγιση του ψηφιακού συστήματος του PetClinic παρατηρείται μια πιο λεπτομερής καταγραφή σε σχέση με την παρουσίαση της BV. Ο χρήστης αρχικά θέλει να εισέλθει στο σύστημα. Επειδή ο Application Server είναι υπεύθυνος για όλες τις λειτουργικότητες, απαραίτητη είναι η προσθήκη ενός LoadBalancer ώστε να μπορεί να διαχειριστεί το φορτίο όταν εισάγονται ταυτόχρονα πολλοί χρήστες. Στην μονολιθική υλοποίηση του Application Server, εσωτερικά υλοποιούνται όλες οι λειτουργικότητες: η διεπαφή χρήστη, η εξυπηρέτηση πελατών, η διαχείριση προσωπικού λογαριασμού, η διαχείριση στοιχείων των κατοικίδιων, η διαχείριση στοιχείων των κτηνιάτρων, η υπηρεσία ραντεβού και επισκέψεων. Επίσης, σημαντικό στοιχείο αποτελεί ότι η μονολιθική υλοποίηση οδηγεί στην χρήση μίας βάσης δεδομένων για όλες τις περιπτώσεις. Ο Application Server επικοινωνεί με την βάση για κάθε λειτουργικότητα που θέλει να εκπληρώσει, και όλα τα δεδομένα, όσο και αν δεν συνδέονται μεταξύ τους βρίσκονται στο ίδιο μέρος (δεν προκύπτει αναγκαιότητα για

διαχωρισμό τους).

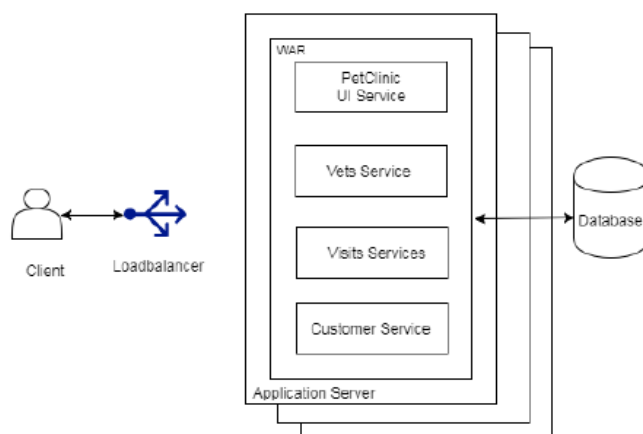


Figure 1: Monolithic Spring PetClinic

Εικόνα 2.4.: Μονολιθική προσέγγιση σχεδιασμού ψηφιακού συστήματος PetClinic. [6]

Σε αντίθεση με το προηγούμενο παράδειγμα που προσφέρει την οπτική του διαχωρισμού ανά είδος των λειτουργικοτήτων, στο παράδειγμα της κλινικής, φαίνεται το πώς η εφαρμογή της λογικής των microservices οδηγεί στην σύλληψη μιας πιο ευέλικτης και διαφορετικής αρχιτεκτονικής. Κάθε υπηρεσία διαχωρίζεται και λειτουργεί αυτόνομα, έχοντας πρόσβαση μόνο στα δεδομένα τα οποία της είναι απαραίτητα, τα οποία πλέον οργανώνονται σε διαφορετικές βάσεις. Μια υπηρεσία δεν έχει πρόσβαση σε δεδομένα τα οποία δεν θα χρησιμοποιήσει. Η υπηρεσία που διαχειρίζεται τα στοιχεία των κτηνιάτρων, δεν χρειάζεται να έχει πρόσβαση στα στοιχεία των πελατών.

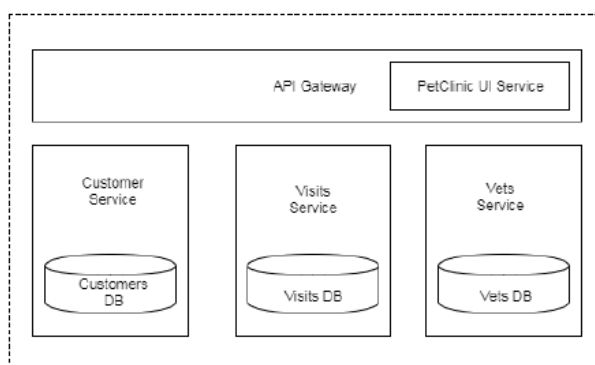


Figure 2: Evolved Architecture of PetClinic with CNA Support Functions

Εικόνα 2.5.: Αντίστοιχη προσέγγιση με microservices προσέγγιση για σχεδιασμού ψηφιακού συστήματος PetClinic της Εικόνας 1.4. [6]

Η διαφορετική αρχιτεκτονική σε σχέση με τα δεδομένα, αλλά και η αυτονομία των διάφορων services απλοποιεί την κατανόηση του συνολικού συστήματος, το οποίο εξακολουθεί να λειτουργεί συνολικά ως μία εφαρμογή. Οι διάφορες υπηρεσίες επικοινωνούν μεταξύ τους με βάση την λογική του συστήματος: για να οριστεί μια επίσκεψη πρέπει να ερωτηθούν οι υπηρεσίες που διαχειρίζονται τα δεδομένα των κτηνιάτρων και των πελατών, και να δημιουργήσει ένα στιγμιότυπο-εγγραφή στη βάση ενός ραντεβού. Επιπλέον, οι αλλαγές σε κάθε service δεν επηρεάζουν τα υπόλοιπα.

Τα παραπάνω παραδείγματα ολοκληρώνουν την παρουσίαση της τεχνολογίας των microservices. Τα πλεονεκτήματα και τα μειονεκτήματα, οι διαφοροποιήσεις στον τρόπο λειτουργίας, εργασίας των ανθρώπων που εμπλέκονται στην ανάπτυξη λογισμικού, αρχιτεκτονικής λογισμικού και σχεδιασμού βάσεων, οριοθετούν το τοπίο για να αναδειχθεί το ζήτημα της ασφάλειας (security) στην τεχνολογία των microservices στο επόμενο κεφάλαιο.

3

Security των Microservices

Έχοντας ορίσει την τεχνολογία των microservices, σε αυτό το κεφάλαιο παρουσιάζονται τα θέματα που ανακύπτουν στον τομέα της παρακολούθησης-monitoring και της ασφάλειας-security. Η διαφορετική δομή, η πολυμορφικότητα των τεχνολογιών και του τρόπου υλοποίησης, το πολύπλοκο δίκτυο επικοινωνίας, η ανεξαρτησία των υπηρεσιών και άλλα χαρακτηριστικά που θα αναλυθούν στην συνέχεια, ορίζουν νέες παραμέτρους και προβλήματα που πρέπει να ληφθούν υπόψη, για τον σχεδιασμό των παραπάνω διαδικασιών.

Αναζητώντας τις αιτίες που το πρόβλημα είναι τόσο σύγχρονο όσο και επιτακτικό, συναντάει κανείς ένα σύνηθες φαινόμενο. Όπως συμβαίνει συνήθως με τις νέες τεχνολογίες και τεχνολογικές λύσεις, τα χαρακτηριστικά αυτά που προσφέρουν νέα πλεονεκτήματα εγείρουν νέα προβλήματα, τα οποία διακρίνονται και αντιμετωπίζονται σε δεύτερο χρόνο. Ακόμη και σήμερα, πάνω από μία δεκαετία μετά την εισαγωγή και την ταχύτατη επικράτηση των microservices σε υλοποιήσεις μεγάλης κλίμακας, η βιβλιογραφία υστερεί σε θέματα που αφορούν την ασφάλεια, και δεν υπάρχουν ευρέως καθιερωμένες μεθοδολογίες και οδηγίες προς τους developers [7]. Ωστόσο, υπάρχουν προσπάθειες να εντοπιστούν τα ευαίσθητα σημεία, και κάποια μεγάλα εγχειρήματα (Docker Swarm, Netflix) έχουν αναπτύξει λύσεις αναφορικά με την ασφάλεια [3], οι οποίες θα εξεταστούν και θα παρουσιασθούν στην συνέχεια.

Το παρόν κεφάλαιο δομείται ως εξής: αρχικά, πρέπει να γίνει μια αναφορά στις διαφορές της νέας αρχιτεκτονικής, εντοπίζοντας αυτή τη φορά τα διάφορα σημεία τα οποία εγείρουν νέα – και παλιά – θέματα ασφάλειας. Στην συνέχεια, με βάση την κατηγοριοποίηση που προτείνουν οι **Yarygina και Bagge [3]** **θα διευκρινιστούν τα διαφορετικά επίπεδα, και τα ευάλωτα σημεία που παρουσιάζουν.** Επιχειρείται μια κατηγοριοποίηση αυτών των χαρακτηριστικών και πεδίων, όπως και μία καταγραφή των λύσεων που υπάρχουν. Τέλος, μια βιβλιογραφική έρευνα για το κατά πόσο υπάρχουν αυτές οι λύσεις, κατά πόσο είναι διαθέσιμες και τα σημεία επίθεσης όπου δίνεται περισσότερη έμφαση.

3.1 Διαφοροποιήσεις αρχιτεκτονικών microservices

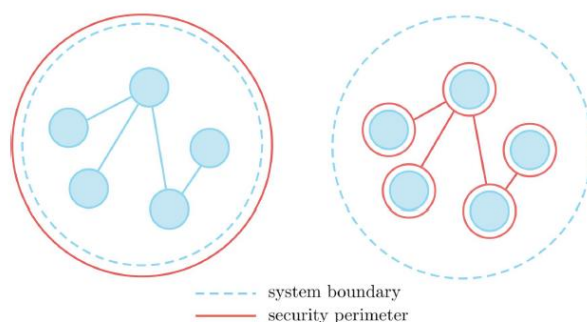
Η κύρια διαφορά που φέρνει η τεχνολογία των microservices είναι η επέκταση της προγραμματιστικής λογικής των μπλοκ κώδικα – μεμονωμένα κομμάτια κώδικα που εκτελούν συγκεκριμένες αυτοτελείς διαδικασίες (π.χ. συναρτήσεις) – προσδίδοντάς τους το στοιχείο της αυτονομίας και της αυτόνομης επικοινωνίας. Ο ρόλος που παίζει συνήθως η *main()* σε διάφορες γλώσσες προγραμματισμού, δηλαδή η σύνθεση και η κλήση ενός ιεραρχημένου, ενθυλακωμένου κώδικα, ανατίθεται σε κάποια microservices επιφορτισμένα με αυτόν τον ρόλο, αλλά ενίοτε μοιράζεται και στα επιμέρους microservices. Εξετάζοντας τα νέα αυτά χαρακτηριστικά και την δομή και ευελιξία αυτής της αρχιτεκτονικής, προκύπτουν οι νέες σημαντικές διαφορές, με επιπτώσεις στην προσέγγιση του θέματος της ασφάλειας:

- **Επικοινωνία σημείων μεταξύ τους.** Από την αρχή της σύλληψης της ιδέας των κατανεμημένων συστημάτων, των SOA και των microservices, ένα κύριο έμφυτο χαρακτηριστικό αποτελεί η επικοινωνία μεταξύ των διαφορετικών κομματιών της εφαρμογής. Κάτι τέτοιο είναι δεμένο με τη “φύση” των microservices και συνάπτονται επικοινωνίες μεταξύ τους πάρα πολλές φορές

ακόμα και μέσα σε πολύ μικρό χρονικό διάστημα. Πέρα από το θέμα του συγχρονισμού και της ενορχήστρωσης (orchestration), τα οποία αποτελούν επίσης ένα σημαντικό θέμα για τον σχεδιασμό της συγκεκριμένης αρχιτεκτονικής, τα πολλαπλά σημεία εισόδου-εξόδου, αποτελούν πιθανά σημεία επίθεσης. Η χρήση περίπλοκων κωδικών πρόσβασης, πιστοποιητικών, η διαδικασία της πιστοποίησης και της εξουσιοδότησης που συνάδουν με μία σύναψη επικοινωνίας, αποτελούν κρίσιμα σημεία στον σχεδιασμό των microservices [8].

- **Συντονισμός σημείων.** Ενώ σε μία μονολιθική προσέγγιση τα πράγματα συμβαίνουν υπό την επίβλεψη μίας και μοναδικής εφαρμογής, στην περίπτωση των microservices, το χαρακτηριστικό της αυτονομίας τους, γεννά την απαίτηση για συντονισμό. Το πώς όλα τα διαφορετικά services θα λειτουργούν αυτόνομα και συντονισμένα, αποτελεί μια ενδιαφέρουσα σπαζοκεφαλιά και απαιτεί προσεκτικό σχεδιασμό. Ωστόσο, όσο η κλίμακα της εφαρμογής μεγαλώνει, τόσο πιο πολύπλοκο γίνεται αυτό το έργο. Η διαταραχή του συντονισμού σε ένα πλέγμα microservices θα μπορούσε να φέρει καταστροφικά αποτελέσματα όσον αφορά στην λειτουργία και την υγεία της εφαρμογής.
- **Διάδοση πληροφορίας (και λάθους).** Πρόκειται για επέκταση του πρώτου φαινομένου. Ενώ ένα από τα χαρακτηριστικά των microservices είναι το να αποτυγχάνουν μονομερώς και να ανακύπτουν αυτόνομα, χωρίς να καταρρέει ολόκληρη η εφαρμογή, στον αντίποδα, σε περίπτωση κακόβουλης επίθεσης, αντί της χρήσιμης πληροφορίας και μηνυμάτων, μπορεί να διαδοθεί μια επίθεση, σε περίπτωση “νόσησης” ενός microservice. Για την αποφυγή αυτού του φαινομένου, τα πρωτόκολλα επικοινωνίας και πιστοποίησης-εξουσιοδότησης πρέπει να είναι πολύ αυστηρά και να έχουν προβλέψει την περίπτωση κακόβουλης επικοινωνίας, ώστε να αποφευχθεί η κακή χρήση ολόκληρης της εφαρμογής, ευαίσθητων δεδομένων, κ.α.
- **Έκθεση σημείων του κώδικα.** Η μονολιθική προσέγγιση στον σχεδιασμό λογισμικού, είχε σαν λογική ότι κάποια “ενδότερα” κομμάτια κώδικα, δεν θα

είναι ποτέ ούτε ορατά, ούτε προσβάσιμα, ούτε προσπελάσιμα εξωτερικά, παρά μόνο εντός της εφαρμογής. Διασπάζοντας-κατανέμοντας την αρχική δομή σε μικρότερα μέρη, κομμάτια τα οποία δεν ήταν σχεδιασμένο ούτε επιθυμητό να είναι προσβάσιμα εξωτερικά, βρίσκονται εκτεθειμένα και σαν σημεία εισόδου-εξόδου, δηλαδή σημεία επικοινωνίας με γειτονικά microservices, άρα και ορατά μέσω του δικτύου [3]. Η νέα αυτή αρχιτεκτονική, δημιουργεί λοιπόν πολλαπλά και νέα σημεία πρόσβασης-σύνδεσης, τα οποία πρέπει να επιστρατεύσουν αυστηρούς κανόνες ασφάλειας για την επικοινωνία τους (όπως αναφέρθηκε και προηγουμένως), καθώς μπορούν να αποτελούν ευαίσθητα, τρωτά σημεία.



Εικόνα 3.1.: Η αλλαγή στην προσέγγιση της αρχιτεκτονικής από μονολιθική σε κατανεμημένη, επιφέρει αλλαγές στα σημεία της εφαρμογής που εγείρουν ζητήματα ασφάλειας. [3]

- **Χρήση διαφορετικών τεχνολογιών.** Η ευελιξία που παρέχουν τα microservices στην επιλογή του cloud παρόχου, αλλά και η επιλογή της γλώσσας προγραμματισμού, τεχνολογίας βάσεων και του ευρύτερου περιβάλλοντος, φέρνει και μια ποικιλομορφία στις απαιτήσεις ασφάλειας. Κάθε γλώσσα προγραμματισμού έχει τις δικές της απαιτήσεις και τα δικά της ευάλωτα σημεία [6]. Ένα σύγχρονο παράδειγμα μεγάλης κλίμακας, μπορεί να θεωρηθεί ένα από τα πιο γνωστά bugs της Java (Apache), “Log4J Bug”, το οποίο εντοπίστηκε στα τέλη του 2021 και έπρεπε να αντιμετωπιστεί με άμεσο Update της έκδοσης της Java (Apache)² που δεν περιείχε την συγκεκριμένη τρωτότητα. Κάτι τέτοιο,

²Άρθρο για το Log4j Java Bug: <https://www.swarmnetics.com/blog/apache-log4j-vulnerability-explained/>

στην περίπτωση των microservices, θα επηρέαζε προφανώς τοπικά μόνο τις εφαρμογές που ήταν γραμμένες σε Java και χρησιμοποιούσαν την συγκεκριμένη βιβλιοθήκη (χωρίς να εξετάζεται η περίπτωση της διάδοσης λάθους σε γειτονικά microservices). Στην συγκεκριμένη περίπτωση θα απαιτούσε άμεση δράση αναφορικά με τα συγκεκριμένα microservices, και όχι σε σχέση με το σύνολο microservices της εφαρμογής.

3.2 Επίπεδα ασφάλειας και ευάλωτα σημεία των Microservices

Όλα τα παραπάνω, αποτελούν χαρακτηριστικά των microservices, τα οποία εξετάστηκαν υπό τον προβληματισμό της ασφάλειας. Η ίδια η αρχιτεκτονική εγείρει θέματα τα οποία στην περίπτωση των μονολιθικών εφαρμογών είτε δεν υπήρχαν (έκθεση ευάλωτων σημείων κώδικα), είτε αντιμετωπίζονταν μονοδιάστατα, στο σύνολο της εφαρμογής (χρήση συγκεκριμένης τεχνολογίας και κοινής υποδομής). Στην συνέχεια, σε επέκταση αυτής της σκέψης παρουσιάζεται μια κατηγοριοποίηση ανά επίπεδο αρχιτεκτονικής – ακολουθώντας σε ένα βαθμό τη λογική των δικτύων, των επτά επιπέδων του OSI – όπως προτείνεται από τους Yarygina και Bagge [3].

Πού προκύπτει η επικοινωνία; Πώς εδραιώνεται μια σύνδεση μεταξύ δύο microservices; Με πόσους τρόπους μπορεί να προσβληθεί ένα microservice; Γιατί κρύβονται κίνδυνοι στη χρήση πολλαπλών τεχνολογιών και παρόχων; Για να απαντηθούν αυτά και άλλα ερωτήματα μεθοδικά, και να εντοπιστούν στο σύνολό τους τα ζητήματα ασφάλειας μιας microservices αρχιτεκτονικής, πρέπει πρώτα να υιοθετηθεί ένα μοντέλο όπου θα μπορέσουν να γίνουν αντιληπτά τα διαφορετικά επίπεδα και οι κίνδυνοι που εγγυμονεί το κάθε ένα από αυτά. Έπειτα, μπορούν ευκολότερα να μεθοδευτούν λύσεις και στρατηγικές σχεδιασμού με γνώμονα την ασφάλεια. Τα επίπεδα

τα οποία θα υιοθετηθούν για την προσέγγιση [3] είναι τα εξής:

- Hardware – Επίπεδο Υλικού
- Virtualization – Επίπεδο Εικονικής Μηχανής
- Cloud – Επίπεδο Νέφους
- Communication – Επίπεδο Επικοινωνίας
- Service/Application – Επίπεδο Υπηρεσίας/Εφαρμογής
- Orchestration – Επίπεδο Ενορχήστρωσης

Παρακάτω πραγματοποιείται μια γενική περιγραφή του κάθε επιπέδου, καθώς και κάποια παραδείγματα επιθέσεων και ευάλωτων σημείων.

Επίπεδο Υλικού (Hardware) Παρόλο που οι προγραμματιστές-developers δεν έχουν κάποια άμεση πρόσβαση στον σχεδιασμό του υλικού, και των διάφορων ευάλωτων σημείων, πρέπει εδώ να αναφερθεί ως επίπεδο, καθώς αποτελεί έναν ιδιαίτερα μεγάλο κίνδυνο και μπορεί να επηρεάσει και την ασφάλεια των υπόλοιπων επιπέδων. Σε περίπτωση που ο σχεδιασμός υλικού δεν συνάδει με τον προγραμματισμό της εκάστοτε εφαρμογής (όπως π.χ. στα ενσωματωμένα συστήματα) η ασφάλεια αυτού του επιπέδου έγκειται στην φερεγγυότητα του κατασκευαστή και την χρήση-επιλογή του κατάλληλου υλικού.

Επίπεδο Εικονικής Μηχανής (Virtualization) Πλέον σχεδόν οι περισσότερες εφαρμογές δεν υλοποιούνται απευθείας σε κάποιο φυσικό μηχάνημα, αλλά αντιθέτως σε κάποια εικονική μηχανή (Virtual Machine – VM) ή αντίστοιχα σε containers. Αυτή η τεχνική αποτελεί μια ριζοσπαστική ιδέα, όπου σε ένα φυσικό μηχάνημα μπορούν με την εγκατάσταση ή ενεργοποίηση του αντίστοιχου λογισμικού να δημιουργούνται εικονικά μηχανήματα, με ανεξάρτητο λειτουργικό, ρυθμίσεις, ip, περιορισμούς, cache memory, μνήμη, σκοπό-είδος χρήσης κλπ. Ο server που διαχειρίζεται τις μηχανές ονομάζεται hypervisor, ενώ η διαχείριση τους αποτελεί μια κρίσιμη διαδικασία για την ομαλή λειτουργία των εφαρμογών, ενώ υπάρχουν διαφορετικά λογισμικά μεγάλων εταιριών που υλοποιούν αυτήν την ιδέα (VMWare, Virtual Box, Hyper-V και άλλα). Η χρήση της εικονοποίησης υψώνει ακόμη ένα τοίχος προστασίας, καθώς υπάρχει πιο ξεκάθαρος

χειρισμός, εύκολη ανάκτηση, και σχεδιασμός των ειδικών χαρακτηριστικών κάθε εικονικής μηχανής, με μια απόσταση από το αντίστοιχο υλικό. Στην περίπτωση των microservices, κάθε microservice μπορεί να υλοποιείται σε διαφορετικό μηχάνημα. Οι κίνδυνοι που εγκυμονούν σε αυτό το επίπεδο αφορούν την παράκαμψη των περιορισμών (sandbox escape), την προσβολή του χειριστή των εικονικών μηχανών, την κακόβουλη χρήση της μοιρασμένης-κοινής μνήμης, κινδύνους που ελλοχεύουν στην δημιουργία μηχανών με την χρήση images που έχουν προσβληθεί ή περιέχουν κακόβουλο λογισμικό. Πέρα από την επιλογή ασφαλών φυσικών μηχανημάτων (προηγούμενο επίπεδο) σε αυτό το επίπεδο, οι ρυθμίσεις κατά την δημιουργία των εικονικών μηχανών, η χρήση επικυρωμένων images και η διαχείρισή τους από φερέγγυους παρόχους, αποτελούν κρίσιμα σημεία ασφάλειας. Επίσης, η διαχείριση και σχεδιασμός ώστε να αποφευχθεί η χρήση κοινής μνήμης, κοινών βιβλιοθηκών, κοινής κρυφής μνήμης είναι καίριας σημασίας, όπως και οι διαρκείς αναβαθμίσεις των εικονικών μηχανών.

Επίπεδο Νέφους (Cloud) Στο επίπεδο του Νέφους οι κίνδυνοι κυρίως αφορούν τον απόλυτο έλεγχο του παρόχου υπηρεσιών νέφους πάνω στο περιεχόμενο (λογισμικό, δεδομένα) και άλλους κινδύνους αναγνωρισμένους στο cloud computing. Η ασφάλεια που μπορεί κάποιος να έχει, έγκειται σε περιορισμένες τεχνικές επιλογές και ρυθμίσεις, για την προστασία από ενδεχόμενη κακόβουλη χρήση ή πρόθεση του παρόχου και εξωτερικές επιθέσεις.

Επίπεδο Επικοινωνίας (Communication) Πρόκειται για ένα από τα πιο καίρια προβλήματα στον σχεδιασμό των microservices, όπως έχει παρουσιαστεί και στις προηγούμενες σελίδες. Όταν ξεκινάει η συζήτηση γύρω από την επικοινωνία μεταξύ των microservices, εγείρονται ίσως τα πιο σημαντικά προβλήματα ασφάλειας της συγκεκριμένης αρχιτεκτονικής, λόγω της ιδιαίτερης πολύπλοκης δομής και του δικτύου επικοινωνίας που αυτή συνθέτει. Επιθέσεις στο δίκτυο, στα πρωτόκολλα δικτύου, επιθέσεις σε μηνύματα που αφορούν την κλοπή μηνύματος, αποκρυπτογράφηση και μετάδοση διαφορετικού μηνύματος ή υποκλοπή του (π.χ. Man in the middle), χρήση ψευδών πιστοποιητικών, διακοπή-καταστροφή ή παραβίαση κάποιου ανοιχτού session [8], μπλοκάρισμα-άρνηση υπηρεσίας (Denial of Service) και άλλες κλασικές επιθέσεις

που αφορούν στην επικοινωνία μέσω δικτύου, απειλούν την ασφάλεια των microservices. Τρόποι, μεθοδολογίες και παραδείγματα για την αντιμετώπιση και την προστασία της επικοινωνίας των microservices μέσω του δικτύου (και συγκεκριμένα διαδικτύου) θα αναλυθούν με λεπτομέρεια στη συνέχεια. Αξίζει να σημειωθεί ότι από αυτό το επίπεδο και στα επίπεδα που ακολουθούν, όσο από το υλικό και τους παρόχους υπηρεσιών εικονικών μηχανών και cloud η συζήτηση προχωράει στο σχεδιασμό λογισμικού και χρήση πρωτοκόλλων, ο σχεδιασμός του λογισμικού όχι μόνο συμβάλει στην ασφάλεια, αλλά αρχίζει να έχει τον πρωτεύοντα ρόλο.

Επίπεδο Υπηρεσίας/Εφαρμογής (Service/Application) Σε αυτό το επίπεδο, αφού η επικοινωνία έχει επιτευχθεί, οι κίνδυνοι αφορούν κυρίως διάφορες εισβολές, παρεμβολές, και διαρροή-κακόβουλη επεξεργασία-καταστροφή δεδομένων εντός της εφαρμογής ή υπηρεσίας. Τέτοιου είδους απειλές αφορούν αρχικά την απόκτηση πρόσβασης και ελέγχου εντός της εφαρμογής, διαπερνώντας ή παρακάμπτοντας τις διαδικασίες πιστοποίησης. Επιπλέον, σαν απειλές σε αυτό το επίπεδο απαντώνται οι SQL injections και η διαρροή, έκθεση ευαίσθητων δεδομένων [8]. Τεχνικές που μπορούν να επιστρατευθούν για μεγαλύτερη ασφάλεια θα συζητηθούν και για αυτό το επίπεδο στη συνέχεια.

Επίπεδο Ενορχήστρωσης (Orchestration) Έχει αναφερθεί και νωρίτερα ότι σε ένα πλέγμα που αποτελείται από microservices ο συντονισμός τους, η διαχείρισή τους και η αυτοματοποίηση διάφορων ενεργειών είναι ζωτικό κομμάτι της συγκεκριμένης αρχιτεκτονικής. Η ορατότητα, μη ορατότητα των microservices, ο εντοπισμός τους, η διαρκής αναδιαμόρφωση του πλέγματος και των συνάψεων επικοινωνίας (π.χ. αλλαγή ip κάποιου microservice, ανανέωση πιστοποιητικών ασφαλείας-certifications), αποτελεί ρουτίνα σε μια εφαρμογή που χρησιμοποιεί αυτή την αρχιτεκτονική λύση. Εδώ οι απειλές που αφορούν στην ασφάλεια, αποτελούν επίθεση στο συντονιστικό έργο, στον εντοπισμό των διάφορων microservices, η αναδρομολόγηση κάποιου μηνύματος ή επικοινωνίας προς ή από κάποιο κακόβουλο service (δηλαδή η “εισβολή” κάποιου service σχεδιασμένο να προκαλέσει κάποια βλάβη, υποκλοπή, ή κακόβουλη διαχείριση της εφαρμογής στο πλέγμα). Παρόλο που αποτελεί ένα από τα κυριότερα θέματα ασφαλείας, ακόμα στο συγκεκριμένο επίπεδο δεν έχει πραγματοποιηθεί εκτενής έρευνα

με μεθοδευμένες λύσεις. Όντας ένα από τα πραγματικά πρωτοεμφανιζόμενα χαρακτηριστικά των microservices, ο ασφαλής συντονισμός, η ασφαλής ενορχήστρωσή τους δεν έχει ακόμη εξαντληθεί και δομηθεί επαρκώς σαν μεθοδολογία και προτεινόμενες λύσεις.

Όπως γίνεται αντιληπτό, στα τρία τελευταία επίπεδα ο ρόλος του σχεδιαστή λογισμικού είναι πολύ σημαντικός για το ζήτημα της ασφάλειας, είτε αφορά το επίπεδο της επικοινωνίας, όπου απαντώνται πρωτόκολλα δικτύου και η ορθή χρήση τους, είτε τα επίπεδα υπηρεσίας/εφαρμογής και ενορχήστρωσης όπου πια η συγγραφή του κώδικα της εφαρμογής έχει τον πρώτο λόγο.

Ξεκινώντας με το επίπεδο μεταφορών (αντιστοιχία με τα επίπεδα που έχουν τεθεί στην παρούσα διπλωματική, του επιπέδου επικοινωνίας), παρουσιάζονται δύο δημοφιλείς λύσεις: της εταιρείας Netflix και της Docker Swarm, αναφορικά με την διαδικασία της πιστοποίησης στην εσωτερική επικοινωνία μεταξύ των microservices και την χρήση του πρωτοκόλλου TLS [3,8]. Στη συνέχεια, παρουσιάζεται και η χρήση tokens ασφαλείας (JSON Web Tokens, JWT) για την ταυτοποίηση του εκάστοτε χρήστη της εφαρμογής.

Και στις δύο περιπτώσεις, υιοθετείται το πρωτόκολλο MTLS (Mutual Transport Layer Security), μια προέκταση του προϋπάρχοντος πρωτοκόλλου TLS (Transport Layer Security), το οποίο όμως ήταν σχεδιασμένο στη λογική server-client και όχι peer-to-peer, microservice to microservice. Επίσης, και στις δύο περιπτώσεις χρησιμοποιείται μία δομή διαχείρισης δημόσιου κλειδιού ανά microservice (self-hosted Public Key Infrastructure – PKI). Έπειτα από την μονομερή πιστοποίηση-αυθεντικοποίηση (TLS) των επιμέρους microservices από ένα κεντρικό service επιφορτισμένο με αυτήν την ευθύνη, στο πρωτόκολλο MTLS, τα microservices πλέον μπορούν να ανοίγουν διάυλο αμφίδρομης επικοινωνίας μεταξύ τους. Μια αναπαράσταση της γενικής αρχής λειτουργίας του πρωτοκόλλου MTLS, φαίνεται στην Εικόνα 2.2..

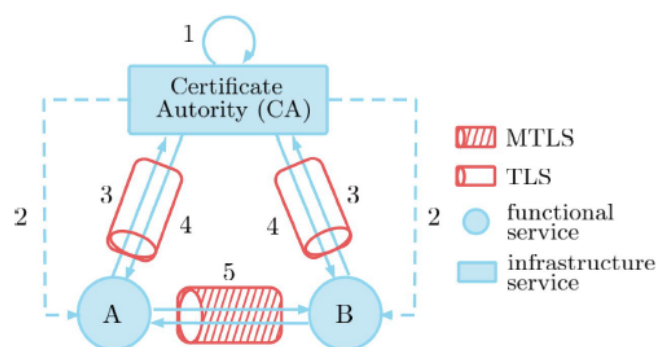


Fig. 3. A generic solution for microservice trust based on MTLS. 1) CA service generates a self-signed root certificate and a shared secret. 2) A cryptographic hash of the certificate and a shared secret are extracted from the CA service and provisioned into the new benign service either automatically or manually. 3) A benign service establishes a one-way TLS connection with the CA service, verifies CA identity using the provisioned hash of the CA certificate, and if successful submits a CSR and shared secret. 4) Upon successful verification of the received CSR and shared secret, CA service signs and sends back the newly issued certificate. 5) Two benign services communicate over MTLS.

Εικόνα 3.2.: Γενική απεικόνιση πρωτοκόλλου MTLS (συμπεριλαμβάνεται και η αρχική λεζάντα για την επεξήγηση του πρωτοκόλλου). [3]

Οι δύο λύσεις παρουσιάζονται ξεχωριστά, καθώς ενώ ξεκινούν με την ίδια σύλληψη, καταλήγουν σε διαφορετικές εφαρμογές.

Η περίπτωση της Docker Swarm [3]

Η λύση της Docker Swarm παρουσιάζει μεγάλο ενδιαφέρον, καθώς αποτελεί μια δημοφιλή πλατφόρμα ανάπτυξης αρχιτεκτονικών καταναμεμένων συστημάτων και microservices, αλλά έχει και μια πληθώρα επιλογών στην επιλογή ρυθμίσεων ασφαλείας. Σε γενικότερα θέματα αρχιτεκτονικής, τα microservices χωρίζονται σε κόμβους εργασίας (worker nodes) και σε κόμβους διαχείρισης (manager nodes). Οι κόμβοι διαχείρισης είναι υπεύθυνοι για την υλοποίηση των πρωτοκόλλων ασφαλείας, και συγκεκριμένα του MTLS με την χρήση PKIs. Περιγράφεται η διαδικασία σε βήματα:

- Ο πρώτος κόμβος διαχείρισης δημιουργεί ένα νέο πηγαίο Certificate Authority, υπογεγραμμένο από τον ίδιο, μαζί με ένα ζεύγος κλειδιών.
- Ένα hash του πηγαίου Certificate Authority μαζί με ένα τυχαίο αριθμό εσωκλείονται σε ένα token, το οποίο μοιράζεται στους υπόλοιπους κόμβους κατά τη διαδικασία του deployment.
- Για να εισαχθεί στο πλέγμα της εφαρμογής, ένας κόμβος ταυτοποιεί τον κόμβο διαχείρισης βάσει του token που έχει στη διάθεσή του, και παράγοντας ο ίδιος ένα αρχικό πιστοποιητικό, το οποίο αποστέλλει στον κόμβο διαχείρισης μαζί με το token, για να αιτηθεί πιστοποίησης.
- Αφού γίνει η ταυτοποίηση-επικαιροποίηση του μυστικού αριθμού από το token από την πλευρά του κόμβου διαχείρισης, ο τελευταίος, αποστέλλει τελικά ένα πιστοποιητικό στον αντίστοιχο κόμβο εργασίας.
- Για την επικοινωνία μεταξύ των κόμβων, ανοίγεται μία σύνδεση TLS, χρησιμοποιώντας τα πιστοποιητικά που τους έχουν διανεμηθεί μετά την διαδικασία ταυτοποίησης-πιστοποίησης-επικαιροποίησης από τον κόμβο διαχείρισης.

Ακολουθώντας αυτά τα βήματα, ο κόμβος διαχείρισης εγγυάται την εισαγωγή των microservices που θα έπρεπε να είναι στο πλέγμα, και διαβεβαιώνει μια ασφαλή επικοινωνία μεταξύ τους. Η ανανέωση των πιστοποιητικών – που αποτελεί και μια καίρια τακτική ασφάλειας – εξ ορισμού συμβαίνει κάθε τρεις μήνες, αν και ανάλογα την περίπτωση μπορεί να μεγαλώσει η συχνότητα, και ασύγχρονα.

Η περίπτωση της Netflix [3]

Αρχικά, ενώ στην περίπτωση της Docker Swarm ο κώδικας είναι ανοιχτός, εδώ δεν

διατίθεται η λύση δημόσια. Αυτός είναι και ο λόγος που οι διαθέσιμες πληροφορίες για την μεθοδολογία που χρησιμοποιείται και οι λεπτομέρειες αναφορικά με την υλοποίηση είναι περιορισμένες.

Η διαφοροποίηση στην περίπτωση της εταιρίας Netflix, έγκειται στην υιοθέτηση μιας τακτικής χρήσης πιστοποιητικών μεγάλης και μικρής-σύντομης διάρκειας. Η γενική ιδέα είναι ότι για την επικοινωνία των microservices χρησιμοποιούνται τα πιστοποιητικά μικρής διάρκειας, τα οποία λήγουν και ανανεώνονται με μεγάλη συχνότητα. Αυτά παράγονται και διανέμονται με βάση τα πιστοποιητικά μεγάλης διάρκειας, τα οποία ανανεώνονται πιο σπάνια.

Αξιίζει να αναφερθεί εδώ, ότι ένα από τα βασικά μειονεκτήματα της χρήσης μικρής διάρκειας πιστοποιητικών που χρήζουν συχνής ανανέωσης, είναι η δυσκολία χρήσης μηχανισμών ανάκτησης, σε περίπτωση αποτυχίας κάποιου κόμβου και επαναφοράς του, καταστροφής κάποιου δίαυλου επικοινωνίας και σε άλλες περιπτώσεις. Επιπλέον, όσον αφορά την επικοινωνία μεταξύ των microservices, πολλές φορές ανάλογα με την λειτουργικότητα που επιτελούν μπορούν να χωρίζονται σε κατηγορίες (π.χ. αν εκτελούν κάποιο ερώτημα χρήστη προς τη βάση, αν εκτελούν κάποιον εσωτερικό υπολογισμό). Υπάρχει και η λογική του πιστοποιητικού ανά κατηγορία microservice, και το πώς οι διάφορες κατηγορίες επιτρέπεται ή όχι να επικοινωνούν μεταξύ τους [3], προτού γίνει πιο συγκεκριμένο το αν κάποιο microservice επικοινωνεί με κάποιο άλλο.

Συνεχίζοντας στην επόμενη θεματική, εκτός από την γεφύρωση επικοινωνίας και αναγνώρισης των microservices μεταξύ τους, ζητήματα αναγνώρισης, αυθεντικοποίησης και πιστοποίησης προκύπτουν και απέναντι στον εκάστοτε χρήστη. Τα microservices που θα εκτελούν τις αιτήσεις-ενέργειες του χρήστη, πρέπει σε κάθε χρονική στιγμή να γνωρίζουν τι συμβαίνει με τον συγκεκριμένο χρήστη (αν έχει ταυτοποιηθεί-πιστοποιηθεί, ποια είναι τα δικαιώματά του, αν έχει λήξει το session κ.ο.κ.). Αναπόφευκτα, αυτές οι πληροφορίες πρέπει να δίνονται στα microservices πολλές φορές, ακόμα και να προωθούνται από τον ένα κόμβο στον άλλον [3].

Για να μπορεί κάθε microservice να έχει διαθέσιμες αυτές τις πληροφορίες, απαραίτητη είναι η ανταλλαγή κάποιων μηνυμάτων τα οποία να τις περιέχουν. Με την δημιουργία tokens ασφαλείας-αυθεντικοποίησης που περιέχουν τα στοιχεία του χρήστη και πληροφορίες για το session που έχει ανοίξει κάτι τέτοιο επιτυγχάνεται. Δημοφιλή tokens που μπορούν να επιτελέσουν αυτόν τον σκοπό είναι τα JSON Web Tokens (JWT), τα οποία χρησιμοποιούνται ευρέως στην αρχιτεκτονική των microservices και ακολουθούν τις βασικές αρχές της κρυπτογραφίας (public-private key). Αυτά περιέχουν πληροφορίες αναφορικά με την ταυτότητα του χρήστη, την ομάδα χρηστών στην οποία ανήκει (δηλαδή πληροφορίες για τα δικαιώματα πρόσβασης και ενεργειών που έχει) και πληροφορίες σχετικά με το session του χρήστη. Επιπλέον και πολύ σημαντικό, τα JWT περιέχουν την ακριβή στιγμή λήξης τους, ώστε να μην μπορούν να χρησιμοποιηθούν εκτός χρόνου και να μπορεί μεμονωμένα κάθε microservice να ελέγχει αν το αίτημα είναι και χρονικά έγκυρο. Όλη η παραπάνω σύλληψη επιτρέπει στα microservices να μπορούν να λαμβάνουν αποφάσεις βάσει των πληροφοριών που έχουν τοπικά διαθέσιμες [3].

Τα θέματα ασφαλείας που ενδεχομένως προκύπτουν από αυτήν την προσέγγιση αφορούν τον συγχρονισμό των ρολογιών των microservices για τον ορθό έλεγχο χρονικής εγκυρότητας των πιστοποιητικών, η ασφαλής διακίνηση και σωστή κρυπτογράφηση των tokens ώστε να μην μπορεί κάποιος να αποκρυπτογραφήσει το μήνυμα και να στέλνει μηνύματα κακόβουλου περιεχομένου προσποιούμενος κάποιον πιστοποιημένο έγκυρο χρήστη, και η ασφάλεια της επικοινωνίας σε επίπεδο μεταφορών [3].

Στο επίπεδο υπηρεσίας/εφαρμογής, ο αρχιτέκτονας λογισμικού και ο προγραμματιστής έχουν την πλήρη ευθύνη για το πως θα γραφεί η εφαρμογή, για τον σχεδιασμό της διαδικασίας ελέγχου (testing) και για διάφορες τεχνολογικές επιλογές. Τα ακόλουθα είναι κάποια σημαντικά σημεία στα οποία χρειάζεται ιδιαίτερη προσοχή και “φροντίδα” στο θέμα της ασφάλειας [3].

- **Ανάλυση του κώδικα:** Και πριν την συγγραφή και κατά την διάρκειά της, και των επιπλέον διορθώσεων, ή προσθηκών, η ανάλυση αποτελεί μια πρώτη κρούση με το θέμα της ασφάλειας. Η σκέψη πάνω στο πως θα σχεδιαστεί ο κώδικας, η ανάλυσή του μετέπειτα ώστε να εντοπιστούν οι διάφορες καταστάσεις-κενά-σφάλματα-θέματα ασφάλειας που μπορούν να προκύψουν. Η ανάλυση είναι μέρος ζωής του μηχανικού και του σχεδιαστή λογισμικού, και επιτρέπει την συγγραφή ενός πιο καθαρού κώδικα, με όλο και λιγότερα θέματα ασφάλειας, λειτουργικότητας και απόδοσης.
- **Σχεδιασμός του κώδικα – εσωτερικά σημεία:** Όπως έχει αναφερθεί και νωρίτερα, η λογική των microservices είναι αυτόνομες, αυτοτελής εφαρμογές που έχουν ως σκοπό την παροχή μιας και μοναδικής συγκεκριμένης ενέργειας-υπηρεσίας. Το πώς αυτός ο κώδικας θα σχεδιαστεί και θα χωριστεί αρχικά είναι κάτι πάρα πολύ σημαντικό και σε επίπεδο ασφάλειας. Κάθε microservice πρέπει να έχει τον έλεγχο για τα δικά του στοιχεία εισόδου, την εσωτερική ροή του, και την διαχείριση εσωτερικών καταστάσεων. Πιο συγκεκριμένα:
 - **Έλεγχος στοιχείων εισόδου (input validation):** Τα δεδομένα, τα μηνύματα, τα tokens, τα αιτήματα που έρχονται στην είσοδο του εκάστοτε microservice πρέπει να ελέγχονται τοπικά ως προς την εγκυρότητά τους, την αυθεντικότητά τους, την αξιοπιστία τους. Βάσει της λογικής ‘μην εμπιστεύεσαι κανέναν’, ό,τι αποτελεί στοιχείο εισόδου πρέπει να ελέγχεται αυτόνομα.
 - **Εσωτερική διαχείριση σφαλμάτων (error handling):** Η διαχείριση σφαλμάτων είναι μια κλασική τακτική του αμυντικού προγραμματισμού. Ο σχεδιαστής λογισμικού πρέπει να έχει προβλέψει όλες τις πιθανές καταστάσεις (ή τουλάχιστον τις περισσότερες) στις οποίες μπορεί να βρεθεί κάποια στιγμή η ροή του κώδικα. Οι ανεπιθύμητες καταστάσεις διαχειρίζονται με συγκεκριμένο τρόπο, ώστε να μην καταρρέει η εφαρμογή, αλλά και να αποφευχθεί η διάδοση λαθών. Σε επίπεδο ασφάλειας, κάτι τέτοιο είναι πολύ σημαντικό, και για την ομαλή ροή, ώστε να μην υπάρχει

διευρυμένη αποτυχία του πλέγματος των microservices, αλλά και για να μην προωθηθεί εσφαλμένη-μολυσμένη πληροφορία σε κάποιον επόμενο κόμβο.

- Συγγραφή σωστών εγχειριδίων (documentation): Στην σύγχρονη εποχή, η συγγραφή κώδικα έχει κλιμακωθεί, έχει γίνει πιο περίπλοκη και μεγάλη σε όγκο, έχει εισχωρήσει σε πάρα πολλά επίπεδα της παραγωγής και της καθημερινής ζωής, έχει περισσότερες απαιτήσεις, έχει δυναμικό χαρακτήρα – και ως προς τις τεχνολογίες και τις απαιτήσεις που συνεχώς αναβαθμίζονται, αλλά και ως προς τα άτομα που διαχειρίζονται μια συγκεκριμένη εφαρμογή. Το θέμα της συγγραφής κατάλληλων εγχειριδίων ώστε η σημαντική πληροφορία για τον κώδικα να είναι διαθέσιμη ανά πάσα στιγμή, είναι πολύ σημαντική για την ασφάλεια. Όταν όλα τριγύρω είναι δυναμικά και πρέπει να αλλάζουν διαρκώς, χωρίς να αφήνουν κενά ασφαλείας, και όταν το μέγεθος του κώδικα/εφαρμογής δεν επιτρέπει εύκολα σε κάποιον να έχει υπό την εποπτεία του το σύνολό του/της σε κάθε μορφή που αλλάζει καθημερινά, το να υπάρχει ένας οδηγός ο οποίος να μπορεί να εξηγήσει το πως η εφαρμογή έχει σχεδιαστεί, και συμπερασματικά ποια είναι τα πιθανά ευάλωτα σημεία της καταλήγει να αποτελεί αναγκαιότητα. Ως εκ τούτου, η διαρκής ενημέρωση των εγχειριδίων, τείνει να γίνει τρόπος ζωής για τους σύγχρονους προγραμματιστές.
- Συγγραφή καθαρού κώδικα (clear code): Όπως και με την συγγραφή σωστών εγχειριδίων, ο κώδικας στον τρόπο που συγγράφεται πρέπει να είναι ξεκάθαρος, βοηθητικός, σπασμένος σε λογικά κομμάτια, ευανάγνωστος, με βοηθητικά ονόματα και σχολιασμό. Ο καθαρός κώδικας βοηθάει και στην γρήγορη κατανόησή του, και στην αποφυγή ‘κρυφών σημείων’ τα οποία μπορεί να αυξάνουν την δυσκολία κατανόησης της ροής χωρίς κάποιον λόγο, και να εμπεριέχουν κινδύνους οι οποίοι δεν είναι άμεσα εμφανείς.

- Ανασκόπηση του κώδικα (code review): Η διαδικασία της ανασκόπησης του κώδικα, αποτελεί μέρος της διαδικασίας συγγραφής κώδικα πλέον, ειδικά στο σύστημα agile. Σε σχέση με την ασφάλεια, το να παίρνει ο κώδικας έγκριση από πολλά και διαφορετικά άτομα, μέσα από συλλογική δουλειά, μπορεί να συνεισφέρει στην αποκάλυψη κενών σημείων που δεν είχαν ληφθεί αρχικά υπόψη. Επιπλέον, η ανασκόπηση του κώδικα σε δεύτερο χρόνο (αφού είναι ήδη στην παραγωγή) μπορεί επίσης να δείξει σημεία που πρωτύτερα δεν θεωρούνταν ευάλωτα, ή σημεία που δεν ήταν ορατά κατά την συγγραφή του κώδικα.
- Προστασία δεδομένων: Η προστασία των δεδομένων αφορά από μόνης της ένα θέμα υψίστης σημασίας για την ασφάλεια. Ήδη την τελευταία δεκαετία, νομοθετικά έχουν γίνει πολλά βήματα προς αυτήν την κατεύθυνση, ιδιαίτερα στην Ευρώπη με την σύνταξη του νόμου περί γενικού κανονισμού για την προστασία δεδομένων (General Data Protection Regulation, GDPR) το 2018³. Στην αρχιτεκτονική των microservices, όπου απαντώνται ποικίλες τεχνολογίες βάσεων και επικοινωνία του εκάστοτε κόμβου με την δική του, απομονωμένη βάση (κατά σχεδιασμό, βάσει της αρχής της αυτονομίας, της τοπικότητας, και του δόγματος ‘μην εμπιστεύεσαι κανέναν’) πρέπει να λαμβάνονται τα κατάλληλα μέτρα από τους προγραμματιστές. Κάθε βάση δεδομένων πρέπει να έχει ρόλους, και δίνει δικαιώματα σε συγκεκριμένους χρήστες να έχουν πρόσβαση στα δεδομένα. Αυτή είναι μια πάγια τακτική για τον έλεγχο πρόσβασης στα δεδομένα μιας βάσης. Αμέσως μετά, αναφέρονται δύο χαρακτηριστικές τακτικές για την προστασία δεδομένων, αναφορικά με την κρυπτογράφησή τους:
 - Data Protection At Rest – Προστασία δεδομένων “εν στάσει”: Πρόκειται για μία από τις πιο εύκολες αλλά σημαντικές τακτικές για την προστασία δεδομένων. Η προστασία – δηλαδή η κρυπτογράφηση της βάσης – όταν αυτή δεν χρησιμοποιείται, ή για όσους δεν την χρησιμοποιούν αποτελεί ένα

³Προστασία δεδομένων στην ΕΕ, από την επίσημη ιστοσελίδα της Ευρωπαϊκής Επιτροπής, link: https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_el

δεύτερο σημαντικό τείχος ασφαλείας, μετά τον ορισμό των ρόλων και των δικαιωμάτων πρόσβασης. Η βάση κρυπτογραφείται, ώστε σε περίπτωση διαρροής των δεδομένων, ο κακόβουλος προγραμματιστής να μην μπορεί να την αποκρυπτογραφήσει αν δεν έχει στην κατοχή του τα κατάλληλα πιστοποιητικά (certifications), κωδικούς (passwords) και κλειδιά.

- Column Encryption – Κρυπτογράφηση συγκεκριμένων πεδίων: Τα δεδομένα που αποθηκεύονται σε μία βάση, όση πληροφορία και αν έχουν συνδυαστικά, μπορούν να διαχωριστούν σε ευαίσθητα και μη. Για παράδειγμα, ο αριθμός λογαριασμού τραπεζής και το ονοματεπώνυμο του κατόχου αποτελούν ευαίσθητα δεδομένα, χωρίς τα οποία το πεδίο στην βάση που περιέχει το ποσό του λογαριασμού, δεν μπορεί να φανερώσει κάποια πληροφορία που να μπορεί να χρησιμοποιηθεί κακόβουλα. Η τάση κρυπτογράφησης συγκεκριμένων ευαίσθητων πεδίων αποτελεί πλέον πραγματικότητα, ώστε μόνο εξουσιοδοτημένοι χρήστες και εφαρμογές να μπορούν να αποκρυπτογραφήσουν αυτά τα πεδία και να αιτηθούν να δουν την πληροφορία που αυτά περιέχουν.
- Επιλογή κατάλληλων τεχνολογιών – Αποφυγή τεχνολογιών με τρωτότητες: Σαν τελευταίο αλλά καθόλου αμελητέο σημείο για το ζήτημα της ασφάλειας, αναφέρεται η επιλογή της βέλτιστης τεχνολογίας ανά περίπτωση. Ένας έμπειρος και ενημερωμένος αρχιτέκτονας λογισμικού ή προγραμματιστής είναι σε θέση να γνωρίζει και να μπορεί να διεξάγει έρευνα κάθε χρονική στιγμή για την επιλογή κατάλληλων εργαλείων και τεχνολογιών (π.χ. βάσεις δεδομένων, γλώσσες προγραμματισμού, πάροχοι, ολοκληρωμένες τεχνολογικές λύσεις). Στην σημερινή εποχή παρέχεται πληθώρα τεχνολογικών λύσεων και επιλογών, κάθε μία από τις οποίες έχει τα δικά της χαρακτηριστικά, και τα δικά της μειονεκτήματα και πλεονεκτήματα. Ανά χρονικές περιόδους, κάποιες τεχνολογίες αποδεικνύονται πιο ισχυρές στο κομμάτι της ασφάλειας και σε άλλα – αν και είναι κάτι που μπορεί και αλλάζει δυναμικά. Κατά τις κρίσιμες αποφάσεις και τις μεγάλες αλλαγές, μια ενδελεχής έρευνα για το ποιες τεχνολογίες είναι πιο ασφαλείς ανά περίπτωση, στον κόσμο της ποικιλομορφίας

των microservices, είναι ένα πολύ σημαντικό βήμα προς μια ασφαλέστερη υλοποίηση.

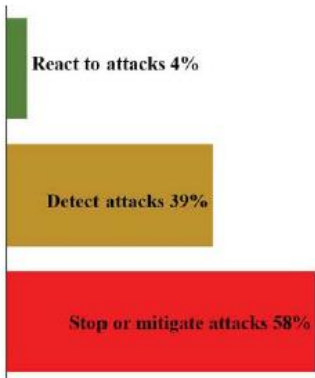
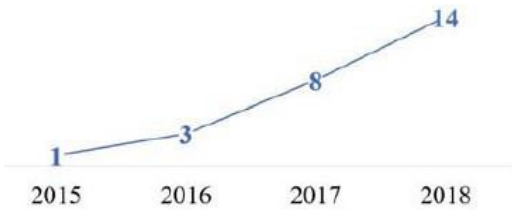
3.3 Αποτύπωση της ασφάλειας των microservices στην βιβλιογραφία

Ολοκληρώνοντας την ανασκόπηση των μεθόδων και τεχνικών που αποσκοπούν στην ασφάλεια κατά την ανάπτυξη και υποστήριξη-συντήρηση λογισμικού, προκύπτει ότι το θέμα της ασφάλειας είναι ποικιλόμορφο, πολυεπίπεδο, επίκαιρο και καίριο. Διαφαίνεται επίσης το πόσο σημαντικό ρόλο παίζει ο σωστός σχεδιασμός λογισμικού και η ετοιμότητα-ενημέρωση του προγραμματιστή. Το θέμα παραμένει πολύπλοκο, και αξίζει σε αυτό το σημείο να γίνει μια αναφορά σε ένα μείζον πρόβλημα που αφορά την ελλιπή βιβλιογραφία, αλλά και την πολωμένη εστίαση της έρευνας πάνω σε συγκεκριμένες πτυχές.

Σε βιβλιογραφική έρευνα του 2019 [7] παρατηρείται το γεγονός ότι κύριος στόχος των ερευνών είναι να σταματήσουν μια ενδεχόμενη επίθεση. Η διαδικασία ανάκαμψης και αντίδρασης σε μία επίθεση έρχεται τελευταία, ενώ η ανίχνευση της επίθεσης έρχεται δεύτερη. Το γεγονός ότι η αντίδραση σε μία επίθεση δεν αποτελεί κύρια προτεραιότητα, μπορεί να εξηγηθεί από την λογική των microservices. Συγκεκριμένα, ένα microservice καλό είναι να αποτυγχάνει εύκολα και λόγω του μικρού μεγέθους και της αυτονομίας του, να επαναδημιουργείται, χωρίς να προκαλεί κάποια εκτεταμένη διαταραχή στην λειτουργία της ευρύτερης εφαρμογής.

Όσον αφορά στην ποικιλομορφία του ζητήματος της ασφάλειας, στην ίδια έρευνα φαίνεται ότι το μεγαλύτερο ενδιαφέρον επικεντρώνεται στην περίπτωση διάρρηξης της διαδικασίας της πιστοποίησης, αυθεντικοποίησης και της χρήσης πιστοποιητικών για κακόβουλη χρήση. Ακολουθούν θέματα που αφορούν την σύναψη επικοινωνίας. Το φαινόμενο αυτό είναι απολύτως λογικό για δύο λόγους: (1) λόγω των μεταβλητών της

έρευνας που διεξήχθη [7], (2) οι συγκεκριμένες δημοσιεύσεις εστιάζουν στα δύο επίπεδα (επικοινωνίας, υπηρεσίας/εφαρμογής) στα οποία έχουν κύριο λόγο οι δημιουργοί και συντηρητές των εφαρμογών. Η ασφάλεια στα υπόλοιπα επίπεδα, όπως αυτά έχουν οριστεί για τις ανάγκες της εργασίας στο παρόν κεφάλαιο, δεν άπτονται της δικαιοδοσίας του αρχιτέκτονα λογισμικού ή του προγραμματιστή που διαχειρίζονται την εφαρμογή. Εξετάζονται υπό το πρίσμα της ασφάλειας, αλλά σε άλλους τομείς, όπως για παράδειγμα σχεδιασμός υλικού. Ωστόσο, είναι αξιοπερίεργο το γεγονός ότι δεν υπάρχει κάποιο ενδιαφέρον ως προς την ασφάλεια στο κομμάτι της εντοπισμού, αν και λόγω πολυπλοκότητας πρόκειται για ένα πρόβλημα μεγάλης δυσκολίας, ακόμα και ως προς τον ορισμό του.

 Fig. 5. Classification of articles according to Architectural security strategies in microservices-based systems.	 Fig. 2. Number of relevant publications per year, from 2015 to 2018.
Εικόνα 3.3.: Θεματική των άρθρων που μελετήθηκαν από τους Pereira-Vale et al. σε σχέση με την προοπτική μιας ενδεχόμενης επίθεσης. [7]	Εικόνα 3.4.: Πλήθος άρθρων ανά χρονιά, όπως αυτά επιλέχθηκαν και μελετήθηκαν από τους Pereira-Vale et al. το 2019. [7]

Παρόλο που πολλά ζητήματα μένουν ακόμα ανεξερεύνητα ή δεν υπάρχει σχετική βιβλιογραφία, στο συγκεκριμένο άρθρο παρατηρείται η αύξηση ενδιαφέροντος από την ακαδημαϊκή κοινότητα τα προηγούμενα χρόνια, με τις σχετικές δημοσιεύσεις να παρουσιάζουν ανοδική εκθετική πορεία. Το γενικότερο συμπέρασμα, είναι ότι το θέμα

της ασφάλειας των microservices αποτελεί ένα ανερχόμενο ερευνητικό πεδίο, με πολύ ανεξερεύνητο υλικό, χώρο για νέες ιδέες και λύσεις.

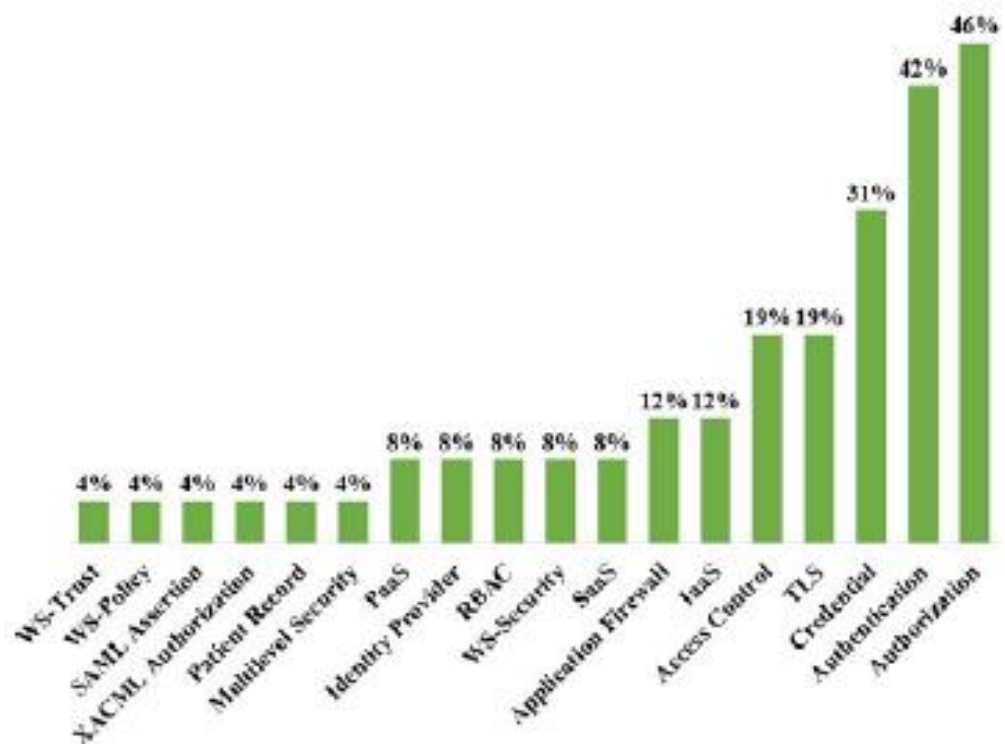


Fig. 4. Security mechanisms reported in the reviewed articles.

Εικόνα 3.5.: Περιοχή ενδιαφέροντος των άρθρων που μελετήθηκαν από τους Pereira-Vale et al. σε σχέση με το είδος μιας ενδεχόμενης επίθεσης. [7]

4

Συστήματα Monitoring στην τεχνολογία των Microservices

Στο προηγούμενο κεφάλαιο αναδείχθηκαν τα θέματα ασφάλειας που προκύπτουν λόγω της διαφορετικής προσέγγισης και της κατανεμημένης αρχιτεκτονικής των microservices. Στο παρόν κεφάλαιο, έχοντας αναδείξει το θέμα της ασφάλειας και έχοντας επιμείνει στις ιδιαιτερότητες και τις νέες προκλήσεις της συγκεκριμένης αρχιτεκτονικής, παρουσιάζεται η σημασία του monitoring για την λειτουργία του συστήματος, και κατ' επέκταση για την ασφάλεια.

Το monitoring ενώ συνήθως νοείται ως η ύπαρξη ηλεκτρονικών συστημάτων σαν μια προσθετική διαδικασία, τα τελευταία χρόνια έχει αναχθεί σε μία από τις πιο σημαντικές λειτουργικότητες που πρέπει να έχει ένα σύστημα, είτε αυτό είναι κάποια γραμμή παραγωγής, είτε cloud εφαρμογές, είτε βάσεις δεδομένων με ταχύ ρυθμό προσπέλασης, ανανέωσης. Τα συστήματα τείνουν να γίνονται πολυπλοκότερα, πιο σύνθετα και μεγάλης κλίμακας. Η “υγεία” ενός συστήματος πρέπει να ελέγχεται, καθώς η λανθασμένη λειτουργία του σε κάποιο σημείο μπορεί να είναι καταστροφική (π.χ. περιβαλλοντική καταστροφή: λειτουργία πυρηνικού αντιδραστήρα, καταστροφή δεδομένων: μια βάση που χάνει ή καθυστερεί ανανεώσεις, καταστροφή προϊόντος: λάθος σε γραμμή παραγωγής).

Η “υγεία” του συστήματος είναι συνδεδεμένη με την ασφάλεια (security) αλλά και με την απόδοση (performance) και ακρίβεια (accuracy). Όπως έγινε ξεκάθαρο νωρίτερο, η ασφάλεια δεν αφορά μόνο κακόβουλες επιθέσεις με στόχο την καταστροφή ή την υποκλοπή δεδομένων, αλλά και την σωστή λειτουργία, την απόδοση και γενικότερα όλα τα υποσυστήματα της εφαρμογής. Η λειτουργικότητα του monitoring-παρακολούθησης επιτρέπει στους κατόχους και υπεύθυνους της εφαρμογής να μπορούν να παρακολουθούν ανά πάσα στιγμή την κατάσταση του συστήματος, και να μπορούν εγκαίρως να επέμβουν, να επαναφέρουν το σύστημα και να αποτρέψουν καταστροφές μεγάλης εμβέλειας.

Το συγκεκριμένο κεφάλαιο δομείται ως εξής. Αρχικά, παρουσιάζεται η έννοια του monitoring, τι αυτή περιλαμβάνει, πως προέκυψε, τεχνικές και μέσα που χρησιμοποιεί. Στην συνέχεια, παρουσιάζεται το πώς εφαρμόζεται το monitoring στην τεχνολογία των microservices, πως η ιδιαιτερότητες αυτής της αρχιτεκτονικής φέρνουν νέες προκλήσεις για τον σχεδιασμό ενός λειτουργικού τρόπου παρακολούθησης. Τέλος, παρουσιάζονται παραδείγματα και εργαλεία και τεχνολογίες που χρησιμοποιούνται για την εφαρμογή ενός συστήματος monitoring στην τεχνολογία των microservices.

Σκοπός του κεφαλαίου είναι να αποδείξει το πόσο σημαντικός είναι ο σωστός σχεδιασμό ενός συστήματος monitoring, ώστε στο επόμενο κεφάλαιο να αναδειχθεί η σύνδεση του με το ζήτημα της ασφάλειας. Επιπλέον, αποτελεί μια προσπάθεια αναζήτησης στη βιβλιογραφία των τεχνικών και μεθοδολογιών που χρησιμοποιούνται, ώστε στο επόμενο κεφάλαιο να μπορέσουν να παρουσιαστούν έχοντας το θεωρητικό υπόβαθρο κάποιες περιπτώσεις εφαρμογής του monitoring σε microservices αρχιτεκτονικές για την ενίσχυση της ασφάλειας.

4.1 Εισαγωγή στην διαδικασία του monitoring

Ξεκινώντας πρέπει να αποσαφηνιστεί η έννοια του monitoring στα πλαίσια των συστημάτων υπολογιστών. Στο εγχειρίδιο *The Art of Monitoring* του Turnbull [9]

δίνεται ο παρακάτω ορισμός:

“Από την πλευρά της τεχνολογίας, το monitoring αποτελεί εργαλεία και διαδικασίες με τις οποίες μετράμε και διαχειριζόμαστε τα τεχνολογικά (IT – Information Technology) συστήματα μας.” [9, ελεύθερη μετάφραση]

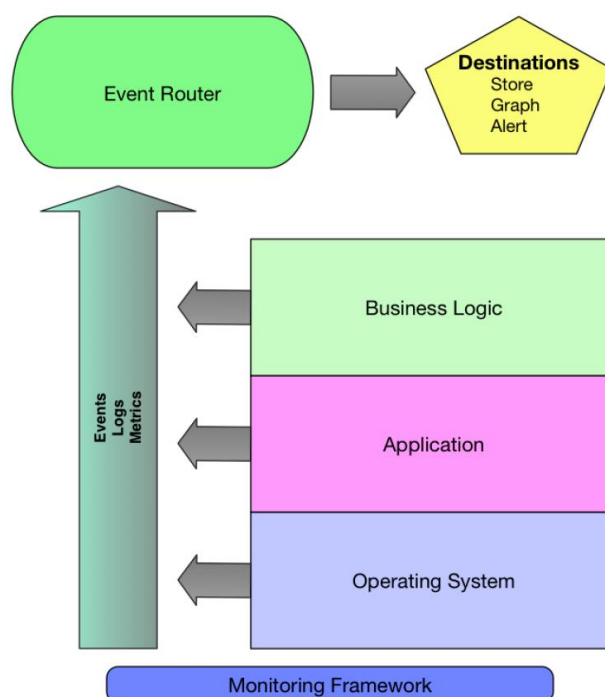
Ωστόσο, όπως αναφέρει και ο ίδιος, το monitoring δεν είναι απλά αυτό, αλλά μια δίοδος να μετατραπούν-εμφανιστούν-μεταφραστούν οι μετρικές του συστήματος σε μια μετρήσιμη εμπειρία του χρήστη, και να αποτυπώσουν την αξία τους στο επίπεδο της επιχείρησης – business value. Αποτελεί το πρώτο βήμα παροχής πληροφορίας ανατροφοδότησης – feedback – για την εικόνα και την λειτουργία του συστήματος. Σε γενικές γραμμές ο Turnbull αναφέρει ότι το σύστημα monitoring απευθύνεται σε δύο κατευθύνσεις, σε αυτήν της επιχείρησης και σε αυτήν του τεχνολογικού τμήματος – IT. Από την πλευρά της επιχείρησης στόχος του monitoring είναι να εξασφαλίζει ότι το σύστημα λειτουργεί σωστά, και παρέχει στον χρήστη μια εμπειρία της ζητούμενης ποιότητας, ενώ βοηθάει στην λήψη επιχειρησιακών αποφάσεων. Από την πλευρά του τεχνολογικού τμήματος, το monitoring τους παρέχει την δυνατότητα να παρακολουθούν το σύστημα, αν αυτό λειτουργεί όπως θα έπρεπε, να εντοπίζουν τα προβλήματα που προκύπτουν και να τα αντιμετωπίζουν, δηλαδή να συντηρούν την εφαρμογή και να μπορούν να βλέπουν πως επηρεάζουν οι διάφορες αλλαγές το σύστημα.

Η διαδικασία του monitoring πάντοτε υπήρχε με την έννοια του ελέγχου και την αναζήτηση της κατάστασης των διάφορων συστημάτων (αν τρέχουν ή όχι) και η επέμβαση για διόρθωσή τους. Ωστόσο, με την πολυπλοκότητα και την κλίμακα που έχουν αναπτύξει τα σύγχρονα συστήματα, ο απλός καθημερινός έλεγχος είναι ανέφικτος και από άποψη χρόνου και όγκου, αλλά και από άποψη ανθρώπινων πόρων (θα χρειαζόνταν πάρα πολύ άνθρωποι για να μπορούν να ελέγχουν τα διάφορα μέρη των μεγάλων υπολογιστικών συστημάτων). Παρουσιάζεται λοιπόν στη σύγχρονη εποχή ένας διαχωρισμός των συστημάτων monitoring ανάλογα με την εμπλοκή του ανθρώπινου παράγοντα: ο χειροκίνητος τρόπος (manual monitoring), ο διαδραστικός τρόπος (reactive monitoring), ο ενεργός τρόπος (proactive monitoring) [9].

Στον χειροκίνητο τρόπο, ο κλασικός έλεγχος διενεργείται, δεν υπάρχουν αυτοματισμοί, όλα γίνονται κατ' εντολή, και σε ορισμένες περιπτώσεις το monitoring δεν νοείται ως σύστημα και δεν έχει κάποια συγκεκριμένη υλοποίηση. Αυτός ο τρόπος αναγκαστικά εγκαταλείπεται όταν η πολυπλοκότητα του συστήματος αυξάνεται, αλλά παραμένει σε πρότζεκτ μικρού όγκου και πολυπλοκότητας ή σε πολύ παλιές υλοποιήσεις που παραμένουν ενεργές. Ο διαδραστικός τρόπος, ή αλλιώς η λογική του “δράση-αντίδραση”, είναι και ο πιο δημοφιλής και κατέχει τη μερίδα του λέοντος, καθώς αφορά έναν συνδυασμό, όπου κάποιες λειτουργίες του monitoring είναι πλήρως αυτοματοποιημένες ενώ κάποια κομμάτια πάλι είναι χειροκίνητα. Το γεγονός αυτό είναι πολύ λογικό σε μεγάλης κλίμακας εφαρμογής, καθώς υπάρχει συνδυασμός πολλών τεχνολογιών σε διάφορα επίπεδα, και περιοχές που δεν έχουν ενδιαφέρον ως προς τη διαρκή επίβλεψη, αλλά υπάρχει ανά περιπτώσεις ανάγκη ελέγχου και βραχυπρόθεσμης παρακολούθησης. Στον τρίτο πιο σύγχρονο τρόπο, τον ενεργό τρόπο, όλη η διαδικασία του monitoring είναι αυτοματοποιημένη, διενεργούνται έλεγχοι και αποστέλλονται προειδοποιήσεις αυτόματα, ενώ διατηρείται και αρχείο. Στην τελευταία περίπτωση είναι εμφανές ότι ο σχεδιασμός και η πρόβλεψη του συστήματος monitoring πρέπει να είναι υπερβολικά προσεκτικός, καθώς απουσιάζει ο ανθρώπινος παράγοντας, τουλάχιστον σε καθημερινή βάση. Σε κάθε περίπτωση, η αυτοματοποίηση του monitoring είναι η τάση που παρατηρείται, όσο και να υπάρχει ακόμα απόσταση από την συνολική αυτοματοποίηση.

Άλλη μία διαφοροποίηση, η οποία έχει ιδιαίτερη σημασία και για την αρχιτεκτονική που εξετάζεται στην παρούσα εργασία, είναι το αν το monitoring αντιμετωπίζει τα διάφορα μέρη ενός συστήματος ως ‘μαύρα κουτιά’ – black boxes – ή όχι. Αν το monitoring ενδιαφέρεται για το τι συμβαίνει εσωτερικά μιας διαδικασίας και αντλεί δεδομένα για την λειτουργία και την απόδοση εσωτερικά, τότε η προσέγγιση ονομάζεται ‘λευκού κουτιού’ – white box, το αντίθετο δηλαδή από το ‘μαύρο κουτί’. Στην αντίθετη περίπτωση, τα διάφορα αυτοτελή μέρη, δεν εξετάζονται εσωτερικά, αλλά σημασία έχει η εξωτερική τους εικόνα, το πως επικοινωνούν με το υπόλοιπο σύστημα, οι συνολικοί χρόνοι απόκρισης κλπ.

Πώς όμως λειτουργεί το monitoring, από που αντλεί δεδομένα και πότε τα παρουσιάζει; Μια γενική εικόνα του πλαισίου λειτουργίας ενός συστήματος monitoring φαίνεται στην Εικόνα 4.1.. Αρχικά το εκάστοτε σύστημα που παρακολουθείται, χωρίζεται σε τρία επίπεδα: το επίπεδο του λειτουργικού συστήματος (π.χ. επίδοση των servers, κύκλοι CPU), το επίπεδο εφαρμογής (π.χ. λειτουργία κώδικα, συνδέσεις) και το επίπεδο επιχειρησιακής λογικής (π.χ. εμπειρία του χρήστη, UIs). Από κάθε επίπεδο αντλούνται δεδομένα και προωθούνται προς το επόμενο επίπεδο. Τα δεδομένα αυτά μπορούν να είναι συμβάντα – events, αρχεία καταγραφής διαδικασιών – logs, και μετρικές – metrics, που αντλούνται για να μετρηθούν διάφορες μεταβλητές. Στο επόμενο επίπεδο, ο Event Router, επεξεργάζεται και προωθεί αυτά τα δεδομένα ανάλογα με την διαχείριση που έχει προγραμματιστεί (προορισμός δεδομένων: γραφικές παραστάσεις, dashboards, βάσεις δεδομένων, άλλοι τρόποι αποθήκευσης κλπ), και με το αποτέλεσμα συγκρίσεων (ενεργοποίηση αποστολής ειδοποιήσεων – alerts).



Εικόνα 4.1.: Πλαίσιο τρόπον λειτουργίας ενός συστήματος monitoring. [9]

Η ροή λοιπόν ενός συστήματος monitoring αφορά στην συλλογή, επεξεργασία, προώθηση, διαχείριση και τελικά προβολή δεδομένων και ενεργοποίηση ειδοποιήσεων.

Ο διαχωρισμός των επιπέδων προδίδει ότι σε κάθε επίπεδο απαιτείται άλλη διαδικασία για την συλλογή των δεδομένων, καθώς επίσης ότι τα δεδομένα και οι μεταβλητές που ενδιαφέρουν σε κάθε επίπεδο είναι διαφορετικά. Ωστόσο, σε υψηλό επίπεδο, σχηματικά η Εικόνα 4.1. περιγράφει αρκετά απτά το πως το σύστημα monitoring αγκαλιάζει το σύστημα που καλείται να παρακολουθεί.

Συνεχίζοντας στον σκοπό ύπαρξης του, ο τομέας του monitoring μπορεί να θεωρηθεί ότι στοχεύει στο να καλύψει και να βοηθήσει σε τέσσερις βασικές λειτουργίες [10], όπως αναφέρεται και πολλές φορές από λογισμικά monitoring⁴ και στην αρθρογραφία της βιομηχανίας⁵. Για όλες αυτές τις διαδικασίες, το monitoring παίζει κυρίαρχο ή συνοδευτικό ρόλο, αλλά σε κάθε περίπτωση στον σύγχρονο σχεδιασμό είναι παρόν.

- **Alerting** – Εκτέλεση ελέγχων και αποστολή ειδοποιήσεων. Σε ένα σύστημα είναι σημαντικό να γνωρίζουν οι υπεύθυνοι, οι προγραμματιστές και οι μηχανικοί λογισμικού πότε έχει συμβεί κάποιο σφάλμα, πότε έχει ξεκινήσει κάποια σειρά μη αποδεκτών γεγονότων, πότε κάποια συστήματα δεν λειτουργούν ή δυσλειτουργούν. Το monitoring είναι συνυφασμένο με αυτή τη λειτουργία. Ένα σύστημα παρακολουθείτε και ελέγχεται για να μπορούν οι υπεύθυνοι να γνωρίζουν πότε πρέπει να επέμβουν ώστε να μην επέλθει κάποια καταστροφή ή πτώση του συστήματος. Εντοπίζοντας εγκαίρως το πρόβλημα και ενημερώνοντας για αυτό, οι κατάλληλοι άνθρωποι μπορούν να λάβουν άμεσα δράση ώστε να αποτρέψουν ή να διορθώσουν μια μη επιθυμητή κατάσταση.
- **Debugging** – Αναζήτηση λαθών και διόρθωση. Έπειτα από την ειδοποίηση των κατάλληλων ανθρώπων μετά τον εντοπισμό κάποιας ανωμαλίας στη λειτουργία του συστήματος, αυτοί πλέον μπορούν να ερευνήσουν, να αναλύσουν το τι συνέβη, να βρουν την αιτία του προβλήματος και να προβούν σε ενέργειες απαραίτητες για την επαναφορά του συστήματος ή του σημείου του συστήματος. Και εδώ το monitoring μπορεί να βοηθήσει, καθώς υποδεικνύει πότε και που εντοπίστηκε το πρόβλημα/ανωμαλία. Επιπλέον σε περιβάλλοντα

⁴Επίσημη σελίδα του λογισμικού Prometheus: <https://prometheus.io/>

⁵Επίσημη σελίδα O'Reilly: <https://www.oreilly.com/library>

ανάπτυξης κώδικα (δηλαδή ότι στο περιβάλλον της παραγωγής) μπορεί να χρησιμοποιηθεί συνδυαστικά, για την αναπαραγωγή του προβλήματος και την παρακολούθησή του, ώστε να ελεγχθεί ότι η αιτία που εντοπίστηκε είναι όντως αυτή που προξένησε το πρόβλημα ή την ανωμαλία.

- **Trending** – Παρακολούθηση τάσεων στο σύστημα. Πέρα από την ειδοποίηση για τον εντοπισμό λαθών ή ανωμαλιών και η επέμβαση για τη διόρθωση αυτών το monitoring μπορεί να φανερώσει και προβλήματα σε βάθος χρόνου. Η παρακολούθηση ενός συστήματος σε μεγάλο χρονικό διάστημα, μπορεί να φανερώσει τάσεις σε διάφορες μετρικές. Το monitoring βασίζεται στην αποθήκευση και παρουσίαση χρονοσειρών. Οι τάσεις που μπορεί να είναι ανοδικές, σταθερές ή καθοδικές, και αφορούν μετρικές που αλλάζουν ή όχι μέσα στο χρόνο μπορεί να αποδειχθεί μια πολύ χρήσιμη πληροφορία, και να οδηγήσει σε αποφάσεις σχεδιασμού ή αλλαγή στρατηγικών. Ένα πολύ σύνηθες παράδειγμα είναι η εξάντληση του χώρου και η επί τοις εκατό διαθεσιμότητα της CPU. Όταν ένα σύστημα χρησιμοποιείται και μεγαλώνει σε κλίμακα, κάποια στιγμή θα εξαντλήσει τους πόρους του. Αν αυτό δεν εντοπιστεί εγκαίρως, μπορεί να προξενήσει μεγάλη δυσλειτουργία, μέχρι και απενεργοποίηση της εφαρμογής, η οποία δυνητικά μπορεί να διαρκέσει για μεγάλο καταστροφικό χρονικό διάστημα. Η πρόβλεψη, ο επανασχεδιασμός, και οι στρατηγικές κινήσεις που μπορούν να προκύψουν από θέματα που φανερώνουν οι τάσεις των μετρικών του συστήματος είναι μία από τις πιο σημαντικές παροχές του monitoring.
- **Plumbing** – Διαμόρφωση και Αξιοποίηση Αγωγών Συστήματος. Το σύστημα monitoring τελικά είναι η παράθεση κάποιων δεδομένων, που είτε προκύπτουν από υπάρχουσες ροές δεδομένων είτε γίνεται η εξόρυξή τους για τον συγκεκριμένο και μόνο σκοπό. Σαν τελευταίο χαρακτηριστικό, διαδικασία, αναφέρεται εδώ αυτή η διαπίστωση. Από την στιγμή που το σύστημα monitoring αντλεί δεδομένα από το σύστημα που καλείται να επιβλέπει, από το να δημιουργούνται μόνο νέες ροές, μπορεί να γίνει αξιοποίηση και των ήδη υπάρχουσών ροών για να αντληθούν τα απαραίτητα δεδομένα. Η αξιοποίηση

του συστήματος όπως ήδη υπάρχει και η προσαρμογή του συστήματος monitoring σε αυτό, είναι κάτι αρκετά συνηθισμένο και συνήθως ελαττώνει όχι μόνο το κόστος σε ανάπτυξη του συστήματος και στην κατανάλωση πόρων (χρόνος, χώρος) αλλά και τα ευάλωτα σημεία του συστήματος. Με το τελευταίο εννοείται ότι αν υπάρχουν λιγότερες ροές δεδομένων – ειδικά αν περιέχονται ευαίσθητα δεδομένα, ή ένας ακόμα τρόπος πρόσβασης σε αυτά – ο κίνδυνος για κάποια υποκλοπή γίνεται μικρότερος. Βέβαια, η διαχείριση των προσβάσεων πρέπει να γίνεται πάντα πολύ προσεκτικά, και σίγουρα θα υπάρχουν δεδομένα τα οποία η ίδια η εφαρμογή δεν θα τα χρειάζεται, οπότε θα πρέπει να αντληθούν (π.χ. χρόνοι απόκρισης, κύκλοι CPU).

Έχοντας πλέον μια αρκετά σαφή εικόνα της έννοιας του monitoring αναφορικά με τα συστήματα και τις εφαρμογές υπολογιστών, τι αυτό περιλαμβάνει και ποιος είναι ο σκοπός του, στη συνέχεια παρουσιάζονται κάποια ζητήματα και ιδιαίτερα χαρακτηριστικά που προκύπτουν κατά την εφαρμογή ενός συστήματος monitoring στην αρχιτεκτονική των microservices.

4.2 Monitoring και Microservices

Σε αυτό το υποκεφάλαιο στόχος είναι να απαντηθεί το εξής ερώτημα: πώς η λογική του monitoring εφαρμόζεται στην αρχιτεκτονική των microservices και ποια είναι τα σημεία στα οποία διαφέρει σε σχέση με την εφαρμογή σε μονολιθικές αρχιτεκτονικές; Ξεκινώντας κανείς πρέπει να αναλογιστεί την διαδικασία ανάπτυξης των microservices, και έπειτα από την ιδιαιτερότητα της κατανεμημένης λογικής που ακολουθούν το πώς τα πράγματα διαφοροποιούνται και που συγκεντρώνεται τελικά το ενδιαφέρον για έλεγχο και επίβλεψη.

Μία εφαρμογή η οποία χρησιμοποιεί την αρχιτεκτονική των microservices ακολουθεί

τρία βασικά στάδια, τα οποία ενδεχομένως απαντώνται και στον κύκλο ζωής των μονολιθικών εφαρμογών. Τα στάδια τα οποία ακολουθούνται για την ανάπτυξη μιας εν λόγω εφαρμογής μπορούν να διακριθούν σε τρία διαφορετικά [11]. Στο πρώτο στάδιο, αφορά στη σχεδίαση της εφαρμογής. Η εφαρμογή αρχικά σχεδιάζεται ως μονολιθική, δηλαδή αποτυπώνεται η λειτουργία της και τα βασικά της μέρη ως κάτι ενιαίο. Στην συνέχεια, αφού έχουν οριστεί οι βασικές λειτουργικότητες που πρέπει να εξυπηρετούνται και τα βασικά μέρη τα οποία θα τις καταστήσουν εφικτές, η διαδικασία της σχεδίασης επισέρχεται στην αποτύπωση των συγκεκριμένων μερών και λειτουργικοτήτων σε κατανεμημένη λογική. Σε αυτήν την διαδικασία απαραίτητος είναι ο προσδιορισμός των επιμέρους υπηρεσιών, η πρόβλεψη και σχεδιασμός των κατά περίπτωση καταστάσεων και συνδέσεων μεταξύ των υπηρεσιών, ώστε να μπορέσει η μονολιθική οπτική να κατανεμηθεί σε ένα λειτουργικό δίκτυο υπηρεσιών. [11]

Το δεύτερο στάδιο αφορά την υλοποίηση όλου του σχεδιασμού και την παράδοσή του στην παραγωγή. Οι διάφορες στρατηγικές που μπορούν να ακολουθηθούν συζητούνται, δοκιμάζονται και τελικά επιλέγεται η καλύτερη προοπτική. Τεχνικές λεπτομέρειες όπως οι επιλογές των τεχνολογιών, ο σχεδιασμός των microservices συνδεδεμένος ή όχι με την λογική των containers, το πώς θα επικοινωνούν τα διάφορα τμήματα αποτελούν το ένα μέρος. Άλλα σημεία απόφασης αποτελεί η στρατηγική με την οποία κάτι θα δοθεί στην παραγωγή, ξεκινώντας συνήθως από τις βασικές λειτουργίας και προσθέτοντας σταδιακά νέες βελτιώσεις και λειτουργικότητες, αντιμετωπίζοντας το εκάστοτε πρότζεκτ από την οπτική μιας agile μεθοδολογίας. Επιπλέον, σε αυτό το στάδιο εντάσσεται και ο έλεγχος-testing ώστε να σταθεροποιηθούν βασικά στοιχεία και να μπορέσει να παραδοθεί η βασική ροή. Τέλος, σε οποιαδήποτε στρατηγική κι αν επιλεγθεί, ο σχεδιασμός έργου πρέπει να είναι τέτοιος ώστε να ενισχύει την διαρκή ανάπτυξη κώδικα και ελέγχου, διαρκών παραδόσεων βελτίωσης (updates), αποφεύγοντας μια λογική σχεδιασμού καταρράκτη (waterfall), ακολουθώντας την λογική του κατανεμημένου, ελαστικού, δυναμικού χαρακτήρα των microservices. [11]

Στο τρίτο και τελευταίο στάδιο, θεωρείται ότι η βασική υλοποίηση έχει βγει στην παραγωγή, η εφαρμογή τρέχει και εξυπηρετεί, ενώ ακόμη μπορούν να προστίθενται

νέες λειτουργικότητες και να γίνονται διορθωτικές επεμβάσεις ή εργασίες συντήρησης. Το τρίτο στάδιο είναι το σημαντικότερο από την πλευρά του monitoring, καθώς σε αυτό το στάδιο, που η εφαρμογή είναι διαθέσιμη προς χρήση, το σημαντικότερο είναι η παρατήρησή της, ώστε να διαπιστώνεται η ορθή λειτουργία της, ώστε να αποφευχθούν προβλήματα, σφάλματα, αποτυχία μερών ή ολόκληρης της εφαρμογής, αλλά και να εντοπίζονται και να αντιμετωπίζονται εγκαίρως επιθέσεις [11]. Επιπλέον, όντας πια μια ολοκληρωμένη εφαρμογή, δίνεται η δυνατότητα μέσω της παρακολούθησης και της παρατήρησης να γίνει ορατή η “συμπεριφορά” της εφαρμογής, να ανακαλυφθεί το αν τελικά το δίκτυο των μικροϋπηρεσιών λειτουργεί όπως αναμενόταν ή αν υπάρχουν νέα θέματα προς επίλυση, που πέφτει το μεγαλύτερο βάρος και πολλά άλλα φαινόμενα που γίνονται ορατά μόνο μετά την παράδοση ενός έργου.

Σημαντικό είναι εδώ να αναφερθεί ότι ενώ το monitoring σαν διαδικασία εμφανίζεται αφού έχει υλοποιηθεί ένα αρκετά μεγάλο μέρος της εφαρμογής, πρέπει να λαμβάνεται υπόψη ως σύστημα και στο στάδιο του σχεδιασμού, και στο στάδιο της υλοποίησης. Το ποια δεδομένα θα παρέχονται, ποιες μετρικές θα είναι διαθέσιμες και πώς αυτό θα μπορεί να πραγματοποιηθεί χωρίς επιβάρυνση ή αλλαγή του αρχικού σχεδιασμού, είναι κριτικής σημασίας.

4.2.1. Βασική Μεθοδολογία

Ποια είναι τα κρίσιμα σημεία που πρέπει να παρακολουθούνται σε μια αρχιτεκτονική microservices; Πώς αλλάζει η οπτική στο σχεδιασμό του συστήματος παρακολούθησης και ελέγχου; Σε ποιο επίπεδο πρέπει να δίνεται περισσότερη προσοχή και ποιοι είναι υπεύθυνοι για το κάθε επίπεδο; Πώς διαφοροποιείται το σύστημα monitoring στην συγκεκριμένη αρχιτεκτονική, ποιες νέες οπτικές φέρνει και ποια ερωτήματα εγείρει;

Αναζητώντας μια οργανωμένη μεθοδολογία, κάποιες αρχές αναφορικά με το πώς θα ήταν προτιμότερο να εφαρμόζεται και να σχεδιάζεται το σύστημα παρακολούθησης (monitoring system) ανακύπτουν κάποιες βασικές αρχές [12]. Αυτές οι αρχές δεν

αφορούν τεχνικές λεπτομέρειες, αλλά περισσότερο μια κατεύθυνση για το τί και σε ποιο επίπεδο είναι σημαντικό να ελέγχεται και να παρακολουθείται, αν είναι σημαντικό κάποια πράγματα να ελέγχονται συνδυαστικά ή μεμονωμένα, το είδος του ελέγχου και την κλίμακα. Παρακάτω παρουσιάζονται πέντε βασικές αρχές, όπως αυτές παρουσιάζονται στον κόσμο της βιομηχανίας και των εταιριών που επιχειρούν να εφαρμόσουν και να αναπτύξουν τεχνολογίες αιχμής.

1. **Παρακολούθηση εσωτερικής λειτουργίας των containers:** Η έννοια των containers έχει αποσαφηνιστεί στο Κεφάλαιο 2. Ένας container είναι μια αυτόνομη συλλογή όλων αυτών των στοιχείων (βιβλιοθήκες λογισμικού, compilers, κώδικας κ.α.) τα οποία είναι απαραίτητα για να στηθεί και να λειτουργήσει ένα σύνολο λειτουργικοτήτων. Στο σύγχρονο κόσμο συγγραφής κώδικα, οι διάφοροι containers κερδίζουν διαρκώς έδαφος λόγω της αυτονομίας τους και της ευκολίας συγγραφής τους απομονωμένα, καθιστώντας τους ένα “μαύρο κουτί” το οποίο επιτελεί μια συγκεκριμένη λειτουργία της οποίας οι λεπτομέρειες δομής και λειτουργίας δεν είναι απαραίτητες για την υπόλοιπη εφαρμογή και για τους προγραμματιστές που δεν έχουν την ευθύνη του. Μια πιο επεξηγηματική τοποθέτηση των containers αναφορικά με το project αλλά και την υλοποίηση των microservices φαίνεται στην Εικόνα 4.2..

Ωστόσο, στην αρχιτεκτονική των microservices, όπου υπάρχουν πολλές αυτόνομες μικρής κλίμακας υπηρεσίες οι οποίες τρέχουν και εκτελούν συγχρονισμένα, η παρακολούθηση της λειτουργίας κάθε μιας από αυτές είναι σημαντική για να επιτευχθεί μια ομαλή λειτουργία από ένα συντονισμένο υγιές σύνολο. Ενώ το πλεονέκτημα του αυτόνομου μαύρου κουτιού δίνει νέες ευκαιρίες και δυνατότητες στον σχεδιασμό και στην ανάπτυξη μεγάλης κλίμακας λογισμικού, από την πλευρά της παρακολούθησης, δεν πρέπει να παραβλέπεται. Προφανώς, σε ένα σύστημα με πολύ μεγάλο αριθμό από microservices δεν είναι δυνατή η οριζόντια παρακολούθησή τους. Αλλά απαιτείται βαθιά γνώση της εσωτερικής λειτουργίας, των μετρικών, και των θεμάτων επίδοσης του κάθε αυτόνομου μαύρου κουτιού. Δηλαδή, το σύστημα παρακολούθησης αντιμετωπίζει νέες προκλήσεις για να μπορέσει να σχεδιαστεί σωστά και να παρακολουθεί τα σημεία που παρουσιάζουν κίνδυνο ή ενδιαφέρον σε επίπεδο αυτόνομων υπηρεσιών μικρής κλίμακας. Σε αυτήν την περίπτωση, προτείνονται δύο κατευθύνσεις. Η μία είναι

εξαρχής τα μικρά αυτόνομα μέρη του συστήματος να σχεδιάζονται από τους προγραμματιστές – ή να οργανώνονται κατάλληλα στη συνέχεια – με την προοπτική να παρέχουν προς το σύστημα παρακολούθησης όλα τα απαραίτητα δεδομένα και μετρικές που είναι απαραίτητα για την παρακολούθηση της λειτουργίας και της υγείας τους. Η άλλη κατεύθυνση αφορά στην παρακολούθηση σε επίπεδο πυρήνων, ώστε να εξετάζεται η λειτουργία των containers σε επίπεδο host. Σε κάθε περίπτωση, το συμπέρασμα της πρώτης αρχής είναι η σημασία που έχει για την σωστή εικόνα του συστήματος η παρακολούθηση σε επίπεδο microservice.

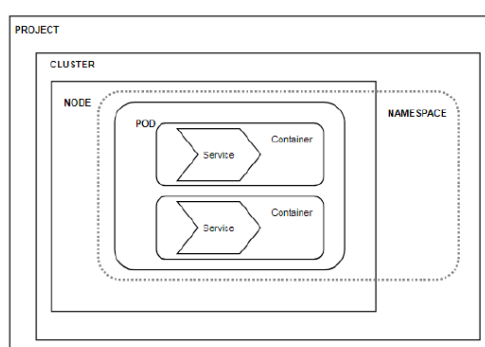


Figure 2. Encapsulation of Microservices inside the Container [12]

Εικόνα 4.2.: Η θέση των containers συγκριτικά με το project και την υλοποίηση των microservices (λεζάντα όπως στο πρωτότυπο στην εικόνα). [11]

2. Παρακολούθηση των συστημάτων ενορχήστρωσης και τοποθέτηση ειδοποιήσεων-alerts αναφορικά με την επίδοση-performance των υπηρεσιών-services: Πέρα από το “χαμηλό” επίπεδο του κάθε microservice, μεγάλης – και για κάποιους μεγαλύτερης – σημασίας είναι η παρακολούθηση της εφαρμογής-υπηρεσίας ως συνολικής οντότητας και της παρακολούθησης των microservices συνολικά. Αυτό μπορεί να αναλυθεί προς δύο προοπτικές. Η μία αφορά τον εντοπισμό των μεγαλύτερων προβλημάτων και των σημείων μεγαλύτερου ενδιαφέροντος (π.χ. τα πιο χρονοβόρα αιτήματα προς την βάση, συχνότητα επισκεψιμότητας κάποιων συνδέσμων, συχνότητα χρήσης

συγκεκριμένων microservices). Όταν τα microservices παρακολουθούνται συνολικά μπορεί να εντοπιστεί πιο ενδεχομένως καθυστέρει, πιο καταναλώνει περισσότερους πόρους, πιο χρειάζεται διόρθωση ή προσοχή. Η δεύτερη αφορά την παρακολούθηση της ενορχήστρωσης του συστήματος. Όσο σωστά και να λειτουργεί η κάθε υπηρεσία μικρής κλίμακας, αν όλες δεν λειτουργούν συντονισμένα, με προγραμματισμένους χρόνους απόκρισης, η εφαρμογή τελικά δεν θα είναι λειτουργική. Το κομμάτι του συντονισμού και της ενορχήστρωσης έχει αναφερθεί και νωρίτερα ως μία από τις μεγαλύτερες προκλήσεις στα κατανεμημένα συστήματα. Διάφορα λογισμικά επιμερίζονται αυτήν την ευθύνη, ενώ διαρκώς βελτιώνονται και νέες λύσεις παρουσιάζονται. Επιπλέον και τονίζοντας τη σημασία του ελέγχου και της παρακολούθησης τα περισσότερα εργαλεία ενορχήστρωσης παρέχουν μετρικές και δεδομένα ή κάποια στοιχειώδη λειτουργία monitoring (dashboards), καθώς προσπαθούν να διαχειριστούν μια πολύπλοκη διάταξη με πολλές παραμέτρους. Για τον τελευταίο λόγο, και επειδή η κλίμακα των εφαρμογών διαρκώς μεγαλώνει, τα monitoring dashboards από μόνα τους δεν θα μπορούσαν να ανταποκριθούν. Η τοποθέτηση alerts, δηλαδή συνθηκών και κατωφλίων που όταν παραβιαστούν αποστέλλουν κάποια ενημέρωση (email, sms, call) για άμεση εξέταση του συμβάντος. Σαν σύνοψη της συγκεκριμένης αρχής, αναδεικνύεται ως υψίστης σημασίας η παρακολούθηση της εφαρμογής ως συνόλου, ως προς την ενορχήστρωση και της συνολικής κατάστασης των microservices αλλά και της συνολικής υπηρεσίας.

3. **Ετοιμότητα για την διαχείριση υπηρεσιών οι οποίες είναι ελαστικές-δυναμικές και βρίσκονται κατανεμημένες εσωτερικά σε διαφορετικές τοποθεσίες:** Στα πλεονεκτήματα της αρχιτεκτονικής έγινε αναφορά στο πως μια τέτοια διάταξη είναι ευέλικτη στην προσαρμογή σε απαιτήσεις μεγαλύτερης κλίμακας αλλά και σε διαρκείς αλλαγές, βελτιώσεις, διορθώσεις και προσθέσεις νέων λειτουργικοτήτων. Στο ζήτημα του monitoring όπως ακριβώς και στο ζήτημα της ασφάλειας, μια τέτοια ευελιξία και δυναμικότητα, εγείρει προβληματισμούς και προκλήσεις υψηλής πολυπλοκότητας. Σε ένα σύστημα που διαρκώς αλλάζει, μεταβάλλεται σε μέγεθος και είναι κατανεμημένο σε

διάφορα σημεία, οποιαδήποτε αλλαγή, δεν θα πρέπει να λαμβάνεται απήφιστα από την πλευρά της ασφάλειας αλλά και της παρακολούθησης. Νέες μετρικές πρέπει να λαμβάνονται υπόψη, νέες λειτουργικότητες προστίθενται και πρέπει να παρακολουθούνται εξίσου, διορθώσεις σε ήδη υπάρχοντα κομμάτια πρέπει να εξετάζονται στο κομμάτι του πως αλλάζει αυτό τις παραμέτρους, μετατοπίσεις μικρών κομματιών σε άλλα σημεία ή ακόμα και πλατφόρμες πρέπει να παρακολουθούνται και να προσαρμόζουν το σύστημα παρακολούθησης. Όπως και οι εφαρμογές έτσι και τα συστήματα ασφαλείας και παρακολούθησης αναγκάζονται να είναι δυναμικά, ελαστικά, και να προσαρμόζονται διαρκώς. Προφανώς, όλη αυτή η διαρκής διαδικασία προσαρμογής δεν μπορεί να γίνεται χειροκίνητα, αλλά να υπάρχει πρόβλεψη τόσο από την πλευρά των σχεδιαστών λογισμικού στον αρχικό σχεδιασμό αλλά και των προγραμματιστών στην προώθηση διάφορων αλλαγών στην παραγωγή, όσο και στον αρχικό σχεδιασμό του συστήματος monitoring, την προσεκτική επιλογή τεχνολογιών monitoring, την επικοινωνία με τις ομάδες προγραμματιστών, devops, και άλλων, ώστε να μπορεί το σύστημα παρακολούθησης να ανταποκριθεί στις απαιτήσεις και διαρκώς νέες ανερχόμενες προκλήσεις της εφαρμογής.

4. **Παρακολούθηση APIs:** Η παρακολούθηση των Διεπαφών Προγραμματισμού Εφαρμογών (APIs – Application Programming Interfaces) με την σειρά της είναι μια σημαντική διαδικασία, καθώς η λειτουργία των APIs αντικατοπτρίζει την εμπειρία του χρήστη της εκάστοτε εφαρμογής (χρόνοι απόκρισης, συχνότητα αιτημάτων), που αποτελεί τελικά και τον σκοπό της ύπαρξής της. Οι διεπαφές είναι τα σημεία στα οποία τα διάφορα κομμάτια της εφαρμογής επικοινωνούν μεταξύ τους, και τα μόνα ορατά σημεία στις διάφορες ομάδες προγραμματιστών, πέρα από την δική τους δικαιοδοσία. Αποτελούν λοιπόν ένα κρίσιμο σημείο για το κατά πόσο τα διάφορα μέρη λειτουργούν ομαλά και συντονισμένα μεταξύ τους. Η παρακολούθησή τους εξασφαλίζει την παρακολούθηση της ομαλής συνεργασίας, τον εντοπισμό των διεπαφών που χρησιμοποιούνται περισσότερο και λιγότερο, καθώς και τις ενδεχόμενες καθυστερήσεις (εντοπισμός bottle-necks) σε διάφορα σημεία [13]. Η

διαφοροποίηση με τον έλεγχο της ενορχήστρωσης αφορά στον έλεγχο των συνδέσεων, ανταλλαγών μηνυμάτων και δεδομένων, και όχι στην συντονισμένη λειτουργία. Για παράδειγμα, μία διασύνδεση μπορεί να είναι πιο αργή καθώς πρέπει να εξυπηρετήσει περισσότερα αιτήματα. Αυτή η περίπτωση θα πρέπει να μελετηθεί, να αναλυθεί και να προταθούν νέες λύσεις (πχ περισσότεροι κλώνοι ενός microservice, παραλληλοποίηση της διαδικασίας) για να υπερκαμφθεί το πρόβλημα. Για αυτόν τον λόγο αναφέρεται ως ο εντοπισμός που βρίσκεται πιο κοντά στην εμπειρία του χρήστη. Συμπερασματικά, έπειτα από τον έλεγχο των επιμέρους microservices, της ενορχήστρωσης και την δυναμική φύση του monitoring, προστίθεται η αναγκαιότητα του ελέγχου των διεπαφών, δηλαδή των διάφορων συνδέσεων.

5. **Σχεδιασμός του Monitoring σε επίπεδο δομής οργανισμού:** Όσο οι αρχιτεκτονικές λογισμικού τείνουν προς μια πιο κατανεμημένη προσέγγιση, και η δομή των οργανισμών ακολουθεί, με την εφαρμογή της δομής και μεθοδολογίας agile να εμφανίζεται σε όλο και περισσότερες εταιρίες, ο σχεδιασμός του monitoring εξυπηρετεί να ακολουθεί αντίστοιχα αυτή τη δομή. Κάθε ομάδα που είναι υπεύθυνη για ένα microservice, μπορεί και γνωρίζει καλύτερα τη σημασία των δεδομένων και των μετρικών του, την κατάσταση ορθής λειτουργίας του και τα ευαίσθητα σημεία του. Η παροχή εργαλείων monitoring (όπως dashboards, alerts κ.α.) στους προγραμματιστές τους επιτρέπει να έχουν καλύτερη εικόνα για το αντικείμενο το οποίο αναπτύσσουν, να το ελέγχουν, να εντοπίζουν λάθη και να το βελτιώνουν. Αλλά επιπλέον, και να μπορούν να παρέμβουν πιο άμεσα και στοχευμένα. Χωρίς να αναιρείται η σημασία της συνολικής εικόνας, σε επιμέρους σημεία οι άνθρωποι που έχουν την εσωτερική γνώση μπορούν να καταλάβουν και να διαχειριστούν καλύτερα την διαδικασία του ελέγχου του προϊόντος που οι ίδιοι έχουν κατασκευάσει. Επομένως, η δική τους γνώση μπορεί να αξιοποιηθεί για τους σκοπούς του monitoring επιτρέποντας παράλληλα και στους ίδιους να έχουν μια πιο σαφή εικόνα της λειτουργίας του κώδικά τους.

Συνοψίζοντας, οι βασικές αρχές που παρουσιάστηκαν αναδεικνύουν την πολυεπίπεδη και πολύπλευρη οπτική που πρέπει να έχει ένα σύστημα monitoring όταν εφαρμόζεται σε ένα κατακεντρωμένο σύστημα. Από την μικροκλίμακα του εσωτερικού της κάθε μικροϋπηρεσίας μέχρι την ενορχήστρωση του συστήματος, από τους υπεύθυνους του συνόλου της εφαρμογής μέχρι τους προγραμματιστές του εκάστοτε τμήματος, το σύστημα monitoring παίρνει κι αυτό την μορφή της αρχιτεκτονικής που εξυπηρετεί: πολυδιάστατο, δυναμικό, ελαστικό, πολυεπίπεδο, κατακεντρωμένο.

4.2.2. Περιοχές Ενδιαφέροντος

Εστιάζοντας λοιπόν, κατέχοντας πια τη μεθοδολογία και την γενική οπτική, στα σημεία τα οποία πρέπει να μετρούνται, να ελέγχονται και να παρατηρούνται, διακρίνονται κάποια ξεχωριστά πεδία. Πέρα από τον διαχωρισμό που φαίνεται στην Εικόνα 4.1., η πηγή των δεδομένων μπορεί να ταξινομηθεί και ως εξής διατηρώντας μια υποτυπώδης αντιστοιχία. Τα πεδία μπορούν να χωριστούν σε τρεις περιοχές: λειτουργία ‘υλικού’ (Power and Performance Monitoring), λειτουργία δικτύου (Network Monitoring), λειτουργία σε επίπεδο δεδομένων χρήστη (User-space Data Collection) [13].

Ξεκινώντας από την περιοχή του ‘υλικού’, τα δεδομένα που συλλέγονται εδώ αφορούν την επίδοση της μηχανής. Δηλαδή, οι μετρικές αφορούν στην χρήση της μνήμης, στους κύκλους και την χρήση της CPU και γενικότερα των υπαρκτών πόρων. Η σημασία της παρακολούθησης των συγκεκριμένων μετρικών έχει σημειωθεί και νωρίτερα στο κομμάτι της παρακολούθησης της τάσης του συστήματος. Όσο κατακεντρωμένο κι αν είναι ένα σύστημα και να ισχυρίζεται ότι χρησιμοποιεί cloud τεχνολογίες, τελικά θα εξυπηρετείται σε κάποια πραγματική μηχανή ή σε κάποια εικονική μηχανή με περιορισμένους πόρους. Η παρακολούθηση αυτών των μετρικών αποκτά όλο και περισσότερη σημασία, καθώς πολλές φορές η κλιμάκωση των εφαρμογών είναι δεδομένη και πρέπει να προβλέπεται στο χρόνο.

Η περιοχή του δικτύου, αφορά όλα τα σημεία επαφής των διάφορων microservices. Τα microservices υλοποιούνται μέσω διαδικτύου, που σημαίνει ότι επικοινωνούν μεταξύ τους με την χρήση διάφορων αιτημάτων. Η παρακολούθηση του δικτύου αφορά στον

έλεγχο αυτών των επικοινωνιών: αν εκτελούνται, τότε αποτυγχάνουν, τότε καθυστερούν κλπ. Εφόσον η συγκεκριμένη αρχιτεκτονική αποτελεί ένα δίκτυο του οποίου οι κόμβοι είναι καταναμημένοι σε διαφορετικές τοποθεσίες και συνήθως επικοινωνούν και συνδέονται μέσω του διαδικτύου, η συγκεκριμένη μετρική είναι πολύ σημαντική. Τόσο στο αν υπάρχει η σύνδεση όσο στο αν αυτή η σύνδεση λειτουργεί όπως αναμένεται.

Τέλος, στο επίπεδο δεδομένων χρήστη, μπορούν να διακριθούν δύο αναγνώσεις. Η μία είναι αυτή που προσπαθεί να εστιάσει στην γενική απόσπαση μετρικών που μπορούν να αποτυπώσουν σε ένα επίπεδο την εμπειρία του χρήστη: παρακολούθηση συγκεκριμένων endpoints, χρόνοι απόκρισης της εφαρμογής, παρακολούθηση της λειτουργίας των APIs. Από την άλλη πλευρά, η συγκεκριμένη κατεύθυνση μπορεί να ασχολείται με το πως αντανακλά η είσοδος ενός χρήστη στα προηγούμενα επίπεδα: πως διαχειρίζεται η κοινή μνήμη, ποια η επιβάρυνση στην CPU, ποια η λειτουργικότητα και επιτυχία της χρήσης κοινών βάσεων και κρυφής μνήμης.

Κλείνοντας το συγκεκριμένο κεφάλαιο, σημειώνεται ότι οι περιοχές ενδιαφέροντος – όπως και να ορίζονται σε κάθε υλοποίηση ή προσέγγιση – πρέπει να αιτιολογούνται επαρκώς, και να προσπαθούν να αγκαλιάσουν το σύνολο της εφαρμογής. Βεβαίως, διαφοροποιήσεις υπάρχουν ανάλογα με την προσέγγιση (μαύρο-λευκό κουτί κ.α.). Κάθε περίπτωση υλοποίησης διαφέρει, οπότε ακόμα και αν υπάρχουν καθολικά μοντέλα και μεθοδολογίες, πάντα στο τέλος η διαδικασία θα εμπεριέχει κάποια στάδια προσαρμογής – customization.

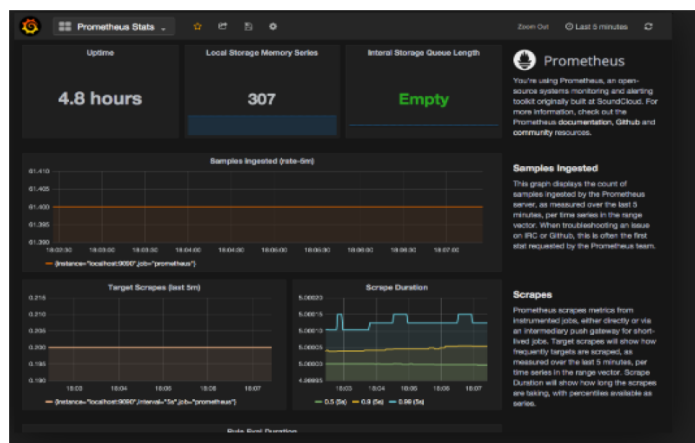
4.2.3. Παραδείγματα Συστημάτων

Έχοντας αναλύσει τις διάφορες κατευθύνσεις και την επικρατούσα μεθοδολογία, σε αυτό το υποκεφάλαιο παρουσιάζονται κάποιες περιπτώσεις εφαρμογής του monitoring σε τεχνολογίες microservices, τα σημεία ενδιαφέροντος, τα μειονεκτήματα και τις προοπτικές της κάθε υλοποίησης. Τρία παραδείγματα αναλύονται σε αυτό το σημείο. Επιλέχθηκαν με στόχο την παρουσίαση διαφορετικών οπτικών και ανάλυσης, διαφορετικών τεχνολογιών υλοποίησης, διαφορετικών οθονών monitoring,

γραφημάτων και τρόπων αναπαράστασης. Επιπρόσθετα, στην πρώτη περίπτωση παρουσιάζεται η προσπάθεια αναβάθμισης συστήματος monitoring με χρήση νέων τεχνολογιών, στην δεύτερη μια θεωρητική ανάλυση και πρόταση ενός συνολικού μοντέλου monitoring, ενώ στην τρίτη ένας σχεδιασμός-υλοποίηση που αναλύει όλα τα σημεία στα οποία μπορεί να εφαρμοστεί ένα σύστημα monitoring. Η ανάλυση των συγκεκριμένων παραδειγμάτων θα χρησιμοποιηθεί και στο επόμενο κεφάλαιο ως παροχές ή απουσία πρόληψης της ασφάλειας.

Στην πρώτη περίπτωση [10] παρουσιάζεται η χρήση λογισμικών ανοιχτού κώδικα, συγκεκριμένα ο συνδυασμός της οπτικοποίησης μέσω Grafana και συλλογή δεδομένων, monitoring και ρύθμιση ελέγχων-alerts μέσω του Prometheus. Ο συγκεκριμένος συνδυασμός τεχνολογιών έχει αρχίσει να χρησιμοποιείται αρκετά συχνά, καθώς αποτελούν ελεύθερα λογισμικά, και υπάρχει ανάπτυξη προς την όλο και καλύτερη συνέργεια των δύο. Σημειώνεται ότι το λογισμικό Prometheus κερδίζει διαρκώς έδαφος λόγω των λύσεων που παρέχει για ένα ολοκληρωμένο σύστημα monitoring.

Σκοπός της συγκεκριμένης μελέτης ήταν η αναβάθμιση του συστήματος monitoring, η εισαγωγή νέων λειτουργικοτήτων όπως ειδοποιήσεων-alerts και η ενημέρωση σε πραγματικό χρόνο του συστήματος monitoring, και η μεταστροφή ενός παλαιού τρόπου monitoring προς ένα σύγχρονο, δυναμικό και διαρκώς εξελισσόμενο σύστημα μέσω του Prometheus. Η ευκολία τοποθέτησης ειδοποιήσεων, ο εύκολος τρόπος συλλογής και καταγραφής μετρικών και αντίστοιχων χρονοσειρών και η γενικότερη ευχέρεια που παρέχει το Prometheus για προσαρμογή στις απαιτήσεις του συστήματος, είναι κάποια από τα πλεονεκτήματα που αναφέρονται σε σχέση με την προηγούμενη υλοποίηση. Ενισχύεται έτσι η επιχειρηματολογία αναφορικά με την καταλληλότητα του συγκεκριμένου λογισμικού για την περίπτωση των microservices.



Εικόνα 4.3.: Οθόνη monitoring για την παρακολούθηση μετρικών του server της εφαρμογής με χρήση τεχνολογιών Prometheus και οπτικοποίηση μέσω εργαλείου Grafana. [10]

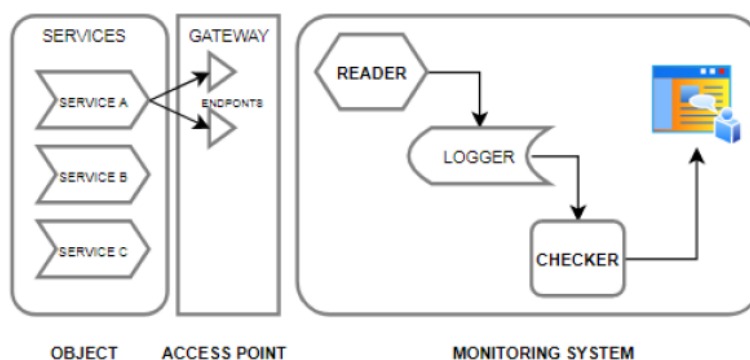
Η δημιουργία οθόνης monitoring με την χρήση Grafana έγινε στην συνέχεια χρησιμοποιώντας τα δεδομένα που είχε συλλέξει το Prometheus στην βάση. Στην Εικόνα 4.3. φαίνεται ένα παράδειγμα μίας οθόνης για το σύστημα με σκοπό την παρακολούθηση μετρικών του server της εφαρμογής.

Στο δεύτερο παράδειγμα που παρουσιάζεται, γίνεται μία προσπάθεια σχεδιασμού ενός γενικού μοντέλου monitoring για αρχιτεκτονικές microservices εστιάζοντας στα εξωτερικά σημεία, σημεία σύνδεσης, σημεία εισόδου-εξόδου – endpoints. Σε αντίθεση με την πρώτη περίπτωση, δεν πρόκειται για εφαρμογή κάποιας τεχνολογίας monitoring σε μία συγκεκριμένη εφαρμογή, αλλά η ανάλυση, μελέτη και γενικός σχεδιασμός, με την προοπτική παρουσίασης ενός μοντέλου εφαρμόσιμου σε κάθε σύστημα microservices, στηρίζοντας θεωρητικά το μοντέλο ως μία συνολική λύση. [11]

Οι τεχνολογίες και ο συνδυασμός τους που προτείνεται για την πιθανή υλοποίηση είναι ή μπορούν να είναι όπως αναφέρεται: “front-end: HTML/JavaScript, Java, back-end: PHP/Python and Apache web server, Java/PHP , data storage: file-based, message broker, database file, file-based” [11]. Το σύστημα που προέκυψε στην συγκεκριμένη εργασία ονομάστηκε MonitorFix, και χρησιμοποιεί διάφορες οθόνες για παρακολούθηση του συστήματος. Δεν παρέχονται λεπτομέρειες υλοποίησης αλλά κυρίως λεπτομέρειες ως προς την στρατηγική σχεδιασμού που αναλύονται αμέσως

παρακάτω.

Ακολουθώντας βασικές αρχές του monitoring, ο αρχικός σχεδιασμός οδήγησε την ομάδα στο διάγραμμα της Εικόνας 4.4., όπου φαίνονται τα microservices που πρέπει να παρακολουθούνται (objects), τα σημεία στα οποία δίνεται η πρόσβαση στα δεδομένα-μετρικές προς παρακολούθηση (access point), και έπειτα από την εξασφάλιση των δεδομένων, το πως αυτά διαβάζονται, καταγράφονται και επεξεργάζονται, μέχρι να είναι έτοιμα προς απεικόνιση (monitoring system). Ιδιαίτερα για το μέρος Checker, αξίζει να αναφερθεί ότι ακολουθείται μια λογική ελέγχου της παραμέτρου-μετρικής ώστε να διαπιστωθεί σε ποια κατάσταση βρίσκεται το εν λόγω microservice. Οι καταστάσεις ορίστηκαν ως ετικέτες: *Success*, *Warning*, *Danger*, *Off*, οι οποίες υπολογίζονται με βάσει την σύγκριση του αριθμού επιτυχιών και αποτυχιών όπως εμφανίζονται στη συλλογή δεδομένων από τα σημεία πρόσβασης σε χρονική σειρά-συνέχεια.



Εικόνα 4.4.: Βασικό σχεδιάγραμμα ροής προτεινόμενου συστήματος monitoring. [11]

Η στοχοθεσία του συγκεκριμένου παραδείγματος είναι συγκεκριμένη: ένα σύστημα monitoring πρώτης γραμμής και γενικής εφαρμογής. Για τον λόγο αυτό, κάποιες στρατηγικές αποφάσεις αναφέρονται, και έκαναν δυνατό ένα γενικό σχεδιασμό. Αρχικά, τα microservices αντιμετωπίζονται ως μαύρα κουτιά – black boxes. Δεν προβλέπεται κάποια ενέργεια monitoring που να αφορά την εσωτερική λειτουργία των microservices ή των containers. Επειδή οι λειτουργικότητες γίνονται πιο συγκεκριμένες ανάλογα με

την εφαρμογή, αυτό το κομμάτι δεν μπορεί να εντάσσεται σε ένα γενικό μοντέλο, παρόλο που θα μπορούσε να υπάρχει πρόβλεψη για παρακολούθηση της κατάστασης των hosts-servers. Ακολουθώντας τον ίδιο τρόπο σκέψης, το σύστημα monitoring σχεδιάστηκε σε επίπεδο εφαρμογής, και όχι σε επίπεδο containers. Τέλος, η παρακολούθηση των μετρικών επιλέχθηκε να γίνει τόσο σε γενικό όσο και σε συγκεκριμένο επίπεδο, και αναπαράσταση των δεδομένων τόσο σε μετρήσιμες μεταβλητές όσο και ως ετικέτες όπως προαναφέρθηκε.

Services About 4 result (0.19 seconds) [+ Register new service](#)

Categories

- Microservices 4
- Endpoints 14

List of Services

ID	MICROSERVICES	ENDPOINT(S)	STATUS
1	User	4	1 Off 1 Danger 1 Warning 1 Success
2	Post	4	2 Danger 2 Warning
3	Product	4	4 Warning
4	Analytic	2	2 Success

Endpoints About 10 result (0.19 seconds) [+ Register new service](#)

Categories

- Microservices 4
- API Endpoints 10

Filter

- ☒ All
- ☐ Off
- ☐ Danger
- ☐ Warning
- ☐ Success

[Confirm changes](#)

List of Endpoints

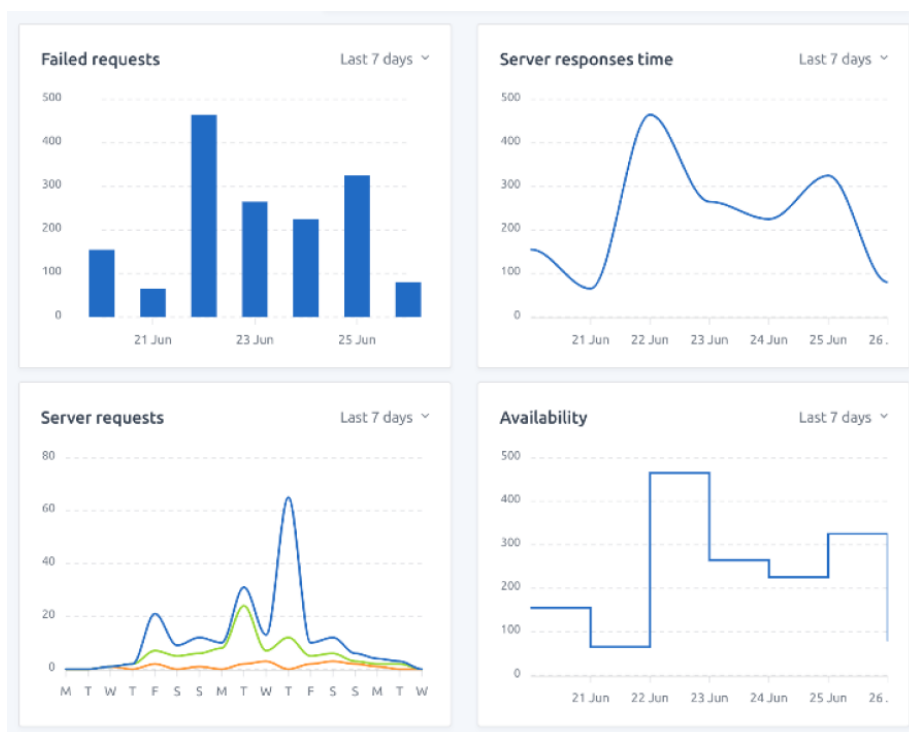
ID	METHODS	ENDPOINTS	SERVICES	STATUS
1	GET	/user/{username}	Service A	Off
2	POST	/user	Service A	Danger
3	PUT	/user/{username}	Service A	Danger
4	DELETE	/post/{postId}	Service B	Danger
5	DELETE	/log/{username}	Service A	Warning
6	GET	/news/{newsId}	Service A	Warning
7	PUT	/post/{postId}	Service B	Warning
8	DELETE	/product/{productId}	Service C	Warning
9	GET	/news/{newsId}	Service A	Success
10	POST	/analytic	Service D	Success

(α)

(β)

Εικόνα 4.5.: Dashboards καταστάσεων: (α) προτεινόμενο dashboard για κύρια οθόνη monitoring της συνολικής κατάστασης των microservices, (β) προτεινόμενο dashboard για κύρια οθόνη κατάστασης των επιμέρους endpoints. [11]

Πέρα από το αναλυτικό θεωρητικό πλαίσιο και την αξία σχεδιασμού ενός ευρείας εφαρμογής, η συγκεκριμένη μελέτη παρουσιάζει και ορισμένους τρόπους οπτικοποίησης, για ευκολότερη λήψη των πληροφοριών από τον άνθρωπο, και με πιο απτή αναπαράσταση της κατάστασης του συστήματος, αλλά και των επιμέρους μερών του.



Εικόνα 4.6.: Μέρος του dashboard για την αναλυτική παρουσίαση των τεσσάρων διαφορετικών παραμέτρων-μετρικών που συλλέγονται για ένα επιλεγμένο endpoint (failed requests, server requests, server responses time, availability). Η παρουσίαση γίνεται σε χρονοσειρές, με διαφορετικούς τύπους γραφημάτων ανά περίπτωση παραμέτρου (δεν συμπεριλαμβάνονται οι λεπτομέρειες που παρέχονται σε μορφή κειμένου, βλ. πρωτότυπο). [11]

Στα παραδείγματα αρχικών dashboards-οθονών της Εικόνας 4.5. παρουσιάζονται σε αθροιστικά σε επίπεδο microservices Εικόνα 4.5. (α) και σε επιμέρους endpoints Εικόνα

4.5. (β) οι καταστάσεις των endpoints βάσει των ετικετών. Η επιλογή προβολής γίνεται από τα κουμπιά με τα αντίστοιχα ονόματα πάνω δεξιά, ενώ στην δεύτερη περίπτωση εμφανίζεται η επιλογή φίλτρων, αν παραδείγματος χάρη κάποιος θέλει να ερευνήσει ποια endpoints βρίσκονται σε κίνδυνο ή είναι ανενεργά, για να δοθεί εντολή διερεύνησης του συγκεκριμένου σημείου στους κατάλληλους τεχνικούς. Επιπλέον, στην περίπτωση των endpoints πέρα από την κατάσταση τους φαίνεται και η μέθοδος που επιτελούν, αποκαλύπτοντας σταδιακά περισσότερες λεπτομέρειες όσο το επίπεδο παρατήρησης γίνεται πιο συγκεκριμένο.

Η τελευταία παρατήρηση είναι πολύ σημαντική. Διαφαίνεται με αυτόν τον τρόπο μια βασική λειτουργικότητα των dashboards, αυτή του drill-down: με αφετηρία την γενική εικόνα, ακολουθώντας τις ενδείξεις μπορεί κάποιος να φτάσει σε χαμηλότερο επίπεδο ώστε να οδηγηθεί στην μελέτη-έλεγχο μίας συγκεκριμένης ύποπτης περίπτωσης. Συνέχεια της συγκεκριμένης λειτουργικότητας είναι και η Εικόνα 4.6. όπου μετά από επιλογή μελέτης ενός συγκεκριμένου endpoint εμφανίζονται διαγράμματα χρονοσειρών για τις διάφορες μετρικές-παραμέτρους που συλλέγονται.

Στο τρίτο και τελευταίο παράδειγμα που παρουσιάζεται εδώ, γίνεται απόπειρα σχεδιασμού και υλοποίησης μιας ολοκληρωμένης λύσης σχεδιασμένη για τον συνδυασμό Dockers-Kubernetes, υιοθετώντας την λογική της οπτικής του μαύρου κουτιού – black box. Για τις ανάγκες αυτές, αναλύεται όλο το σύστημα των microservices όπως ορίζεται από τις συγκεκριμένες τεχνολογίες, και προσεγγίζονται όλα τα πεδία στα οποία είναι χρήσιμο να εφαρμοστεί κάποιου είδους μέτρηση για να ελέγχονται. Για κάθε περίπτωση (π.χ. παρακολούθηση δικτύου, κίνησης, συνδέσεων) περιγράφεται αναλυτικά ο τρόπος μέτρησης και απόσπασης δεδομένων. [13]

Αρχικά γίνεται μία καταγραφή και γενική ανάλυση του συστήματος, και χωρίζονται οι περιοχές ενδιαφέροντος για την απόσπαση μετρικών και μεταβλητών σε τρία διαφορετικά επίπεδα. Δίνεται μεγάλη έμφαση στην διαχείριση των μετρικών για τοποθέτηση ειδοποιήσεων-alerts και δημιουργία νέων συνδυαστικών μετρικών. Αυτό ακολουθεί μια μαθηματική ανάλυση και επεξήγηση των φραγμάτων μέσω από την

βιβλιογραφία. [13]

Πραγματοποιείται λοιπόν μία περιγραφή για ανάλυση του συστήματος, όσον αφορά το λειτουργικό σύστημα και την κατανάλωση ενέργειας, και μία ανάλυση του TCP πρωτοκόλλου και το πώς αυτό μπορεί να επιβλέπεται στα διάφορα σημεία. Τέλος, αναλύεται το πώς θα συλλέγονται τα δεδομένα σε επίπεδο χρήστη. Η συγκεκριμένη εργασία διαφέρει και συμπεριλαμβάνεται για την ενδελεχή μαθηματική ανάλυση και επισκόπηση συστήματος που πραγματοποιεί. [13]

Στην συνέχεια παρουσιάζεται μία ανάλυση της ροής-pipeline που θα ακολουθείται για την συλλογή και την πορεία των δεδομένων. Ο συνδυασμός βάσεων και κόμβων επεξεργασίας αλλά και σημείων συλλογής και εξόδου δεδομένων φαίνεται αναλυτικά στο διάγραμμα της Εικόνας 4.7.. [13]

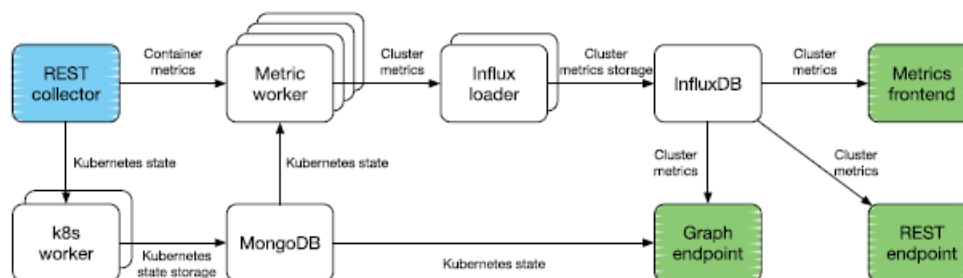


Fig. 6. Backend pipeline structure: The REST collector (highlighted in blue) is the entry point that collects all the data coming from the agents, while metrics frontend, REST endpoint, and Graph endpoint (highlighted in green) are the endpoints used to visualize and query data. Components are connected with a queuing system enabling distribution, replication, and asynchronous communication. Arrows show the data flow.

Εικόνα 4.7.: Ροή δεδομένων (συμπεριλαμβάνετε η λεζάντα του πρωτότυπου) [13]

Τέλος, αφού έχει πραγματοποιηθεί η παραπάνω ανάλυση, ορίζονται τα σημεία στα οποία θα εστιάσει η γραφική ανάλυση δεδομένων, δηλαδή ποιος θα είναι ο σκοπός των διαγραμμάτων που θα παραχθούν. Οι γράφοι εστιάζουν σε τρία σημεία: την απεικόνιση

της αλληλεπίδρασης μεταξύ των microservices, την απεικόνιση της δυνατότητας των microservices να αποκριθούν σε αιτήματα, και πως η γενική απόδοση των στοιχείων του κατανεμημένου συστήματος αντανακλά στην συνολική απόδοση. [13]

Η παρουσίαση των παραδειγμάτων είχε ως σκοπό ανάδειξη του προβλήματος του monitoring στην βιβλιογραφία και αναζήτηση διαφορετικών οπτικών στην αντιμετώπιση αυτών. Στο επόμενο κεφάλαιο θα γίνουν σχόλια σε σχέση με την ασφάλεια και θα παρουσιαστούν παραδείγματα που εστιάζουν σε αυτό.

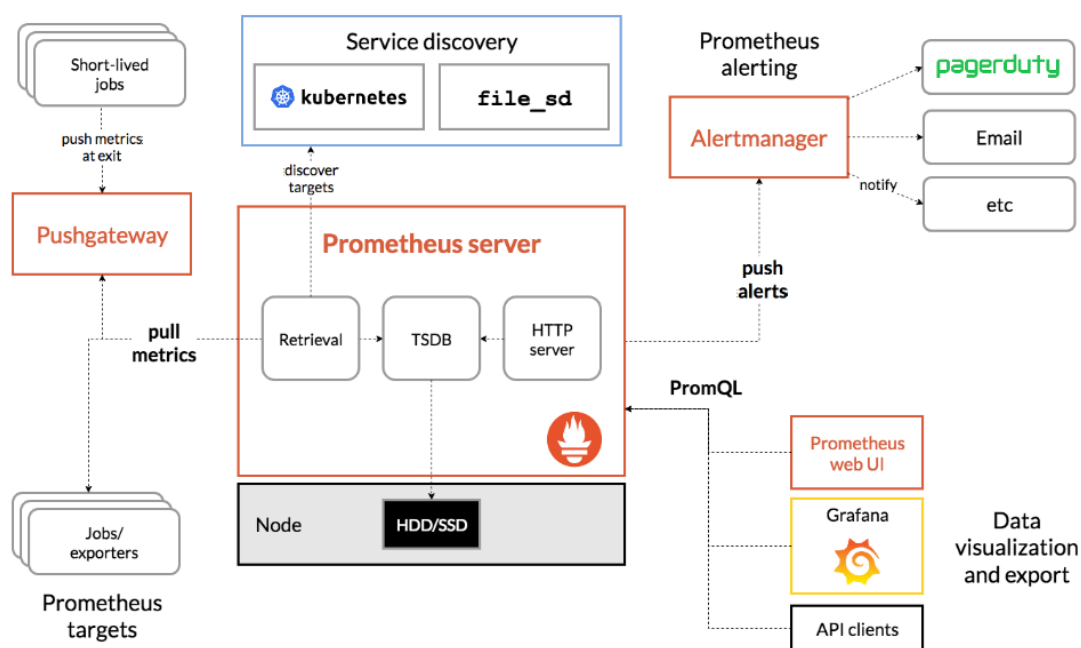
4.3 Σημαντικότερες Τεχνολογίες Monitoring για την αρχιτεκτονική Microservices

Κλείνοντας το παρόν κεφάλαιο για τα συστήματα monitoring στην αρχιτεκτονική των microservices, γίνεται μία αναφορά σε βασικές τεχνολογικές λύσεις και λογισμικά που χρησιμοποιούνται, τόσο ανοιχτού κώδικα, όσο και ολοκληρωμένες λύσεις της βιομηχανίας.

Prometheus ⁶

Το λογισμικό Prometheus αναπτύχθηκε και εξελίχθηκε ως ένα βοηθητικό λογισμικό για τους προγραμματιστές και τους υπεύθυνους εφαρμογών στην διαδικασία ανάπτυξης και παραγωγής συστημάτων, όπως διάφορων εφαρμογών, εργαλείων, βάσεων και άλλων. Στην συνέχεια μετεξελίχθηκε σε ένα από τα πιο δημοφιλή εργαλεία ανοιχτού λογισμικού ως αναπόσπαστο κομμάτι των συστημάτων monitoring. Μπορεί να εγκατασταθεί, να ρυθμιστεί και να τεθεί σε λειτουργία μέσω εντολών σε κάποιον Linux Server [14].

⁶Επίσημη σελίδα του λογισμικού Prometheus: <https://prometheus.io/>



Εικόνα 4.8.: Η αρχιτεκτονική δομή του συστήματος Prometheus και των διάφορων μερών του.⁷

Η αρχιτεκτονική του Prometheus δομείται γύρω από έναν κύριο κεντρικό server, τον Prometheus Server. Κάθε server στο σύστημα του Prometheus επιμερίζεται την παρακολούθηση ενός συγκεκριμένου αντικειμένου, διαδικασίας ή συστήματος (όπως για παράδειγμα ενός Linux Server ή μίας συγκεκριμένης διαδικασίας ή βάσης) τα οποία ονομάζονται Targets. Ο Prometheus Server συλλέγει μετρικές από τα Targets και τα εισαγάγει σε μία βάση που ακολουθεί χρονολογική ταξινόμηση-χρονοσειρές, ώστε να μπορεί να χρησιμοποιηθεί στην συνέχεια για εξαγωγή γραφημάτων, ενεργοποίηση alerts, ανάλυση χρονοσειρών και άλλες εφαρμογές. Επίσης, υπάρχει λειτουργία διάφορων alertmanagers, ώστε να διετελούνται έλεγχοι και να στέλνονται αντίστοιχες ενημερώσεις. Σε άλλα βασικά συστατικά μέρη του Prometheus βρίσκονται οι Exporters οι οποίοι χρησιμοποιούνται για την εισαγωγή μετρικών από third-party συστήματα, και την ένταξή τους στην βάση.

Στην Εικόνα 4.8. παρουσιάζεται η αρχιτεκτονική του Prometheus, πως ο Prometheus

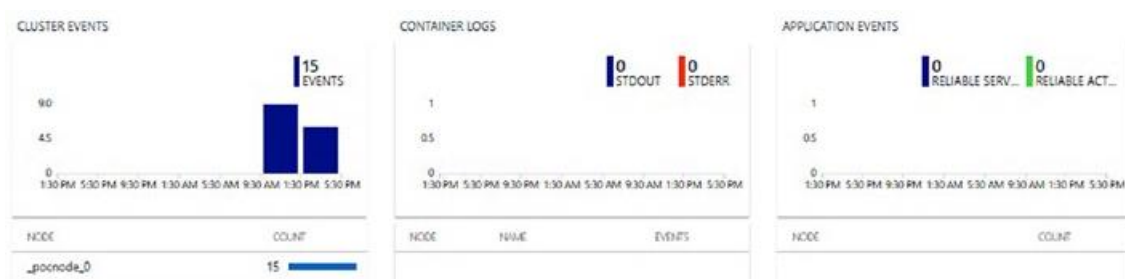
⁷Επίσημη σελίδα του λογισμικού Prometheus: <https://prometheus.io/>

Server συλλέγει μετρικές από διάφορους Targets και Exporters και αποθηκεύονται ως χρονοσειρές στην βάση, πως τροφοδοτείται ο Alertmanager και μπορεί να αποστείλει ειδοποιήσεις με διάφορους τρόπους, πως άλλα λογισμικά οπτικοποίησης δεδομένων (π.χ. BI tools) μπορούν να αντλήσουν δεδομένα για να τα χρησιμοποιήσουν για την δημιουργία οθονών monitoring.

Microsoft Azure⁸

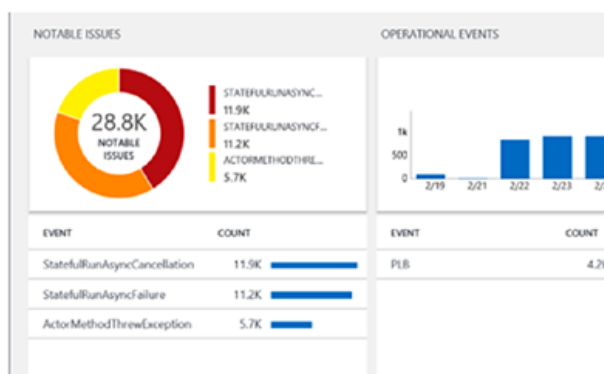
Η πλατφόρμα Azure της Microsoft αποτελεί ένα cloud περιβάλλον που χρησιμοποιεί microservices, παρέχοντας μια ολοκληρωμένη λύση για υλοποίηση εφαρμογών στην συγκεκριμένη αρχιτεκτονική. Μέσα στην πλατφόρμα συμπεριλαμβάνονται υπηρεσίες ασφαλείας και monitoring [15]. Όλο και περισσότερες εταιρίες αναζητούν πλατφόρμες που να παρέχουν ολοκληρωμένες λύσεις, και το Azure είναι μία από τις πιο δημοφιλείς επιλογές.

Από την στιγμή που η εφαρμογή έχει υλοποιηθεί στην πλατφόρμα, η ενεργοποίηση των υπηρεσιών monitoring είναι μία εύκολη διαδικασία, που μπορεί να υλοποιηθεί σε λίγα βήματα. Με πληθώρα επιλογών σχεδιασμού και έτοιμα templates, με συνδυασμό κώδικα και άλλων επιλογών ανάλυσης δεδομένων και υπολογισμών, το σύστημα monitoring σχεδιάζεται και μπορεί να προσαρμοστεί σε κάθε περίπτωση [15]. Στις Εικόνες 4.9.(α) και (β) φαίνονται παραδείγματα οθονών monitoring της πλατφόρμας.



(α)

⁸Επίσημη σελίδα του Microsoft Azure: <https://azure.microsoft.com/en-us/>



(β)

Εικόνα 4.9.: (α) Cluster Events Dashboards. (β) Παράδειγμα αρχικού γενικού Dashboard για συμβάντα. [15]

Sysdig⁹

Το λογισμικό Sysdig αναπτύχθηκε με σκοπό να παρέχει υπηρεσίες ασφαλείας και monitoring σε συστήματα που χρησιμοποιούν Kubernetes, Containers και γενικότερα αρχιτεκτονικές Cloud και microservices. Πρόκειται για μία από τις πιο δημοφιλείς ολοκληρωμένες λύσεις σε αυτόν τον τομέα, ενώ έχει αναπτυχθεί ειδικός σχεδιασμός για συμβατότητα με το λογισμικό Prometheus. Όντας ένα εργαλείο το οποίο εξ αρχής είχε ως κατεύθυνση τις τεχνολογίες Cloud, από το 2013 ασχολείται αποκλειστικά με αυτό. Ενώ τα πρώτα βήματα έγιναν προς την κατεύθυνση ανάπτυξης ενός εργαλείου monitoring σταδιακά άρχισε να παρέχει και λύσεις security.

Grafana¹⁰

Το λογισμικό Grafana ανήκει σε μία κατηγορία λογισμικών που ασχολούνται με το κομμάτι της ανάλυσης – analytics, και την οπτικοποίηση δεδομένων, με διαδραστικές οθόνες, φίλτρα και διαφορετικές γραφικές αναπαραστάσεις και τύπους διαγραμμάτων. Πρόκειται για ένα λογισμικό ανοιχτού κώδικα το οποίο πρωτοεμφανίστηκε και χρησιμοποιήθηκε το 2014. Σταδιακά άρχισε να είναι συμβατό με όλο και περισσότερες βάσεις, ενώ υπάρχει άμεση σύνδεση και προσπάθεια για διαρκή συμβατότητα με το λογισμικό Prometheus.

⁹Επίσημη σελίδα του λογισμικού Sysdig: <https://sysdig.com/>

¹⁰Επίσημη σελίδα του λογισμικού Grafana: <https://grafana.com/>

Άλλα λογισμικά οπτικοποίησης και ανάλυσης δεδομένων

Το εργαλείο Grafana παρουσιάστηκε ξεχωριστά ως ένα εργαλείο οπτικοποίησης και ανάλυσης δεδομένων λόγω της στενής συγγενείας του με το λογισμικό Prometheus. Άξια αναφοράς είναι και άλλα λογισμικά που έχουν δημιουργηθεί για τον ίδιο σκοπό, ωστόσο δεν είναι λογισμικά ελεύθερου κώδικα. Αναφέρονται τρία από τα πιο δημοφιλή, παραλείποντας το Azure της Microsoft, καθώς αποτελεί μια ολοκληρωμένη πλατφόρμα για υλοποίηση εφαρμογών σε αρχιτεκτονική cloud, microservices που παρέχει εργαλεία monitoring και ανάλυσης, οπτικοποίησης δεδομένων. Το εργαλείο SAS¹¹ είναι από τα πιο γνωστά για επεξεργασία, ανάλυση και οπτικοποίηση δεδομένων. Επιπλέον, εμπεριέχει δυνατότητα εφαρμογής αλγορίθμων μηχανικής μάθησης, ενώ το προϊόν SAS Viya είναι το ίδιο υλοποιημένο σε microservices[16], δίνοντας νέες δυνατότητες στον κόσμο του monitoring. Κλείνοντας, τα λογισμικά Tableau¹² και Qlik¹³ παρέχουν και αυτά λύσεις οπτικοποίησης και ανάλυσης, με το πρώτο να μπορεί να σχεδιάσει μέσω διάφορων προϊόντων (Tableau Prep¹⁴) ολόκληρες ροές δεδομένων και το δεύτερο να μπορεί να συνδυαστεί με διάφορα συγγενή προϊόντα.

¹¹Επίσημη σελίδα της SAS: https://www.sas.com/el_gr/home.html

¹²Επίσημη σελίδα της Tableau: <https://www.tableau.com/>

¹³Επίσημη σελίδα της Qlik: <https://www.qlik.com/us/>

¹⁴Προϊόν Tableau Prep: <https://www.tableau.com/products/prep>

5

Προτάσεις για Security Monitoring στην αρχιτεκτονική Microservices

Από την ανάλυση στο προηγούμενο κεφάλαιο και την αναζήτηση στη βιβλιογραφία προκύπτει ότι ενώ τα συστήματα monitoring θεωρούνται απαραίτητα και ζωτικής σημασίας στις σύγχρονες εφαρμογές, δεν υπάρχει μια συνολική συμπεριληπτική λύση όσον αφορά το σχεδιασμό και τα σημεία που πρέπει να εστιάζει το σύστημα. Λόγω της πολυπλοκότητας και της διαφορετικότητας των εφαρμογών, συνήθως δίνεται βάρος σε διαφορετικά σημεία, ενώ επίσης και οι τεχνολογικές λύσεις διαφέρουν. Ωστόσο, το γεγονός ότι έχει διαμορφωθεί και υιοθετηθεί μια κοινή οπτική σχεδιασμού, ανοίγει το δρόμο για διαφορετικές αξιόπιστες υλοποιήσεις.

Επιπλέον, παρατηρείται ότι το κομμάτι της ασφάλειας αυτό καθεαυτό δεν αποτελεί μία από τις προτεραιότητες, ωστόσο σημειώνεται ότι το monitoring μπορεί να βοηθήσει ιδιαίτερα στην εξασφάλιση της ασφάλειας [11]. Παρόλα αυτά, σημειώνεται ότι η ομαλή λειτουργία της εφαρμογής, αφήνει λιγότερα ευαίσθητα προς επίθεση σημεία, καθώς επίσης και μία μη ομαλή λειτουργία και απόδοση της εφαρμογής μπορεί να αποτελεί μια ένδειξη κάποιας επίθεσης η οποία έχει προξενήσει πρόβλημα στο σύστημα.

Στο παρόν κεφάλαιο, το οποίο αποτελεί και την έρευνα της συγκεκριμένης διπλωματικής εργασίας, παρουσιάζεται η εκπόνηση της έρευνας περίπτωσης, όπου

αναζητήθηκαν παραδείγματα που σε εφαρμογές που χρησιμοποιούν την αρχιτεκτονική microservices θέτουν το σύστημα monitoring στην υπηρεσία της διαχείρισης της ασφάλειας.

Σκοπός του κεφαλαίου είναι να συγκεντρώσει παραδείγματα από την σύγχρονη βιβλιογραφία και μέσω της ανάλυσης και της σύγκρισης, αλλά και της θεωρητικής γνώσης, να διακρίνει τις τάσεις και τα κενά που υπάρχουν στον συγκεκριμένο εξειδικευμένο τομέα. Τα συμπεράσματα που παρουσιάζονται στο επόμενο και τελευταίο κεφάλαιο και η έρευνα που παρουσιάζεται στο παρόν, μπορούν να αποτελέσουν έναν οδηγό προς την σχεδίαση συστημάτων monitoring υποδεικνύοντας τα ζητήματα ασφαλείας στα οποία μπορεί να συνεισφέρει. Επιπλέον, μπορεί να αποτελέσει βάση για συζήτηση, σχετικά με τα ερωτήματα που μπορούν να τεθούν σε πλαίσιο έρευνας και θεωρητικής ανάλυσης, αλλά και τα σημεία στα οποία υπάρχει χώρος για πειραματισμό και χρήζουν λύσεων, όσο η αρχιτεκτονική των microservices και των κατανεμημένων συστημάτων υιοθετείται από όλο και περισσότερες εφαρμογές και συστήματα έναντι της μονολιθικής προσέγγισης.

Η δομή του κεφαλαίου χαράζεται ως εξής. Αρχικά παρουσιάζεται η μεθοδολογία που ακολουθήθηκε στην έρευνα, και στηρίζονται με επιχειρήματα οι διάφορες επιλογές και αποφάσεις για το τι θα συμπεριλαμβάνει η έρευνα. Επίσης, παρουσιάζονται τα βήματα και οι διαδικασίες. Στην συνέχεια, αναπτύσσεται ένα θεωρητικό πλαίσιο. Αναδεικνύεται η σύνδεση των δύο συστημάτων monitoring και security, όπως επίσης και οι οπτικές και βασικές αρχές που έχουν θεσπιστεί και χρησιμοποιούνται. Θέτοντας έτσι τις θεωρητικές βάσεις, στη συνέχεια παρουσιάζονται παραδείγματα από την βιβλιογραφία, και σχολιάζονται υλοποιήσεις που αφορούν στην σχεδίαση συστημάτων monitoring με σκοπό την χρήση τους για σκοπούς ασφαλείας αλλά και συστήματα monitoring τα οποία εμπεριέχουν κομμάτια που μπορούν να βοηθήσουν στην παρακολούθηση της κατάστασης της ασφάλειας στην αρχιτεκτονική των microservices.

5.1 Μεθοδολογία

Η συγκεκριμένη έρευνα δομήθηκε ως εξής. Αρχικά έγινε μία προσπάθεια να οικοδομηθεί ένα σχετικό θεωρητικό πλαίσιο που παρουσιάζεται στα προηγούμενα κεφάλαια. Η αποσαφήνιση και ορισμός των τεχνολογιών των microservices, αλλά και του τομέα του security και του monitoring ήταν απαραίτητα ώστε να μπορεί να παρουσιαστεί η εφαρμογή του monitoring για λόγους security στην τεχνολογία των microservices. Τα ιδιαίτερα χαρακτηριστικά, τα πλεονεκτήματα και τα μειονεκτήματα, οι μεθοδολογίες, ο σκοπός ύπαρξης των παράλληλων συστημάτων security και monitoring, οι τεχνολογίες, τα λογισμικά που χρησιμοποιούνται, η ανάλυση που χρησιμοποιείται και άλλα παρουσιάστηκαν με κατεύθυνση πάντα στο θέμα της συγκεκριμένης διπλωματικής.

Συνεχίζοντας στο κομμάτι της έρευνας, αναπτύχθηκαν δύο κατευθύνσεις. Η μία αφορά την αναζήτηση σχετικής θεωρίας στην βιβλιογραφία. Το πως ο τομέας της ασφάλειας χρησιμοποιεί αλλά και αποτελεί κομμάτι του συστήματος monitoring θεωρείται ως κάτι δεδομένο και για τον λόγο αυτό έχουν παγιωθεί κάποιες βασικές αρχές, μεθοδολογίες και σημεία στα οποία θα έπρεπε να παρακολουθούνται για λόγους ασφάλειας. Η αναζήτηση έγινε τόσο σε επίσημα τεχνολογικά εγχειρίδια καθορισμού προτύπων (έγγραφα της NIST – National Institute of Standards and Technology) και σε ακαδημαϊκές δημοσιεύσεις που κάνουν νύξη για το συγκεκριμένο θέμα, αλλά και σε έγγραφα της βιομηχανίας, όπου στην προσπάθεια παροχής μίας ολοκληρωμένης λύσης monitoring και security κάνουν σαφή τη σύνδεση των δύο συστημάτων, τονίζοντας κάποιες αρχές σχεδιασμού και εφαρμογής (π.χ. εταιρία Sysdig, λογισμικό Prometheus).

Η αναγκαιότητα έρευνας αναφορικά με την θεωρία, αποτέλεσε εφόδιο για την αξιολόγηση και διερεύνηση των διαφόρων προτάσεων για εφαρμογή ενός συστήματος monitoring σε αρχιτεκτονική microservices. Οι βασικοί άξονες που ορίζονται, μπορούν κατέχοντας τη θεωρητική βάση να εντοπιστούν και να αναλυθούν.

Το παραπάνω σχόλιο αποτελεί γέφυρα για το δεύτερο σκέλος της έρευνας. Σε αυτό το κομμάτι πραγματοποιήθηκε αναζήτηση στην ακαδημαϊκή βιβλιογραφία και στα άρθρα-

έγγραφα της βιομηχανίας με σκοπό τον εντοπισμό παραδειγμάτων σχεδιασμού συστημάτων monitoring που αποσκοπούν στην συνεισφορά στην ασφάλεια του συστήματος. Πραγματοποιήθηκαν δύο αναζητήσεις με μόνο χρονολογικό φίλτρο να έχουν δημοσιευθεί τα τελευταία πέντε χρόνια, λόγω των διαρκών ταχέων εξελίξεων σε μία τεχνολογία αιχμής:

- **security + monitoring + microservices (+ system/example/deployment/dashboard)¹⁵**: Σκοπός σε αυτήν την αναζήτηση ήταν να βρεθούν παραδείγματα και αναλύσεις για την εφαρμογή του συστήματος monitoring με γνώμονα την ασφάλεια.

Σχετικά Αποτελέσματα

- Ακαδημαϊκή Βιβλιογραφία: 7
- Έγγραφα/Αρθρογραφία Επιχειρήσεων: 1
- **monitoring + microservices (+ system/example/deployment/dashboard)¹⁶**: Σε επέκταση του παραπάνω, ο όρος “security” αφαιρέθηκε, ώστε να διευρυνθεί το αποτέλεσμα της αναζήτησης. Μεθοδολογικά, σε αυτό το μέρος βάσει του θεωρητικού πλαισίου εντοπίστηκαν σημεία των προτεινόμενων συστημάτων monitoring για την αρχιτεκτονική των microservices τα οποία θα μπορούσαν να συνεισφέρουν στην παρακολούθηση της κατάστασης ασφαλείας του συστήματος. Από τα αποτελέσματα επιλέχθηκαν αυτά τα οποία κρίθηκαν ότι προσδίδουν αξία στον σκοπό της παρούσας έρευνας και είναι σχετικά με το θέμα.

Σχετικά Αποτελέσματα

- Ακαδημαϊκή Βιβλιογραφία: 2

¹⁵Στο διάγραμμα της Εικόνας 5.1. αναφέρεται ως **search 1**.

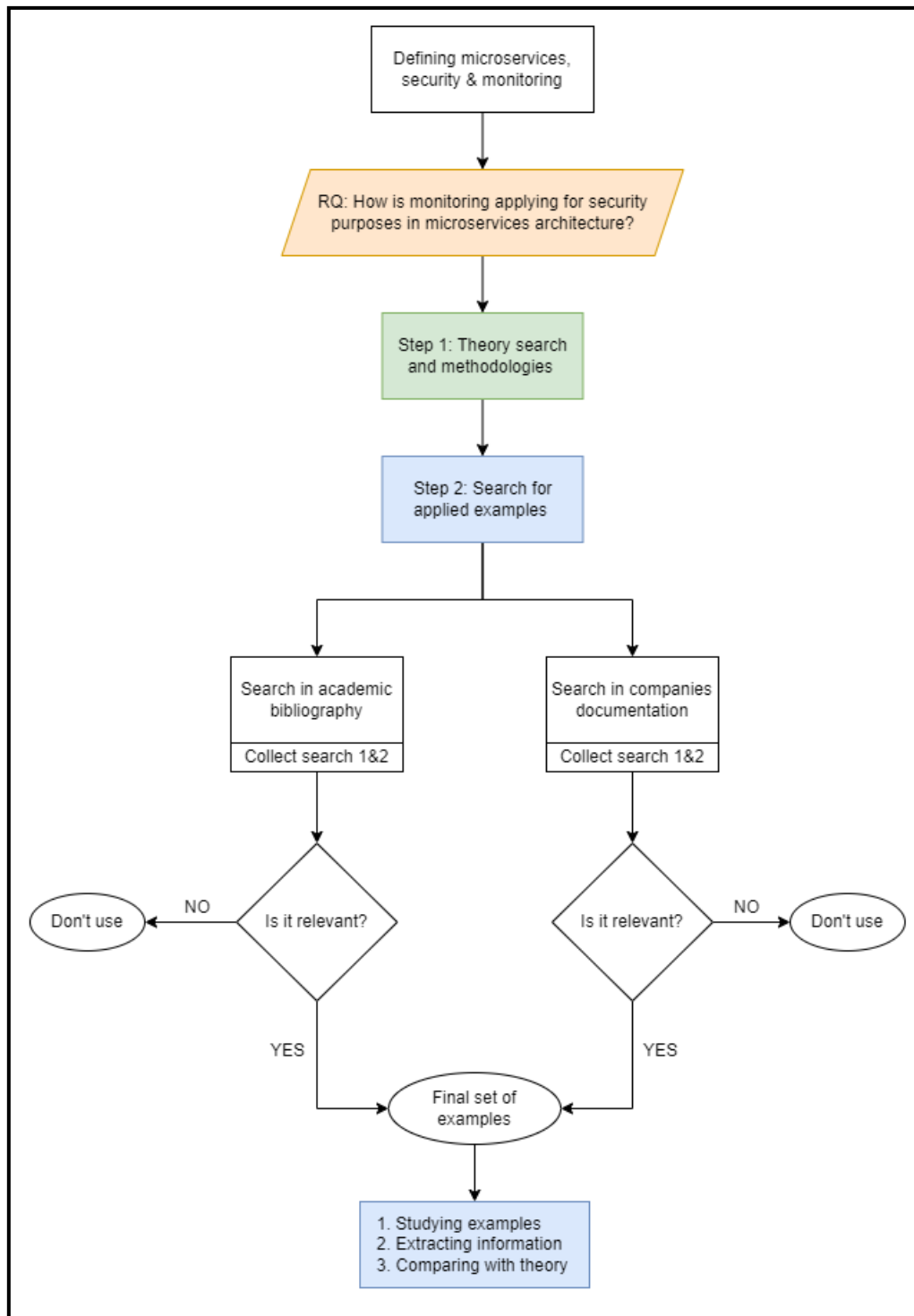
¹⁶Στο διάγραμμα της Εικόνας 5.1. αναφέρεται ως **search 2**.

Μελετήθηκαν συνολικά 10 άρθρα για συστήματα monitoring σε αρχιτεκτονικές microservices. Σε αυτά συμπεριλαμβάνονται και τα παραδείγματα που παρουσιάστηκαν στο Κεφάλαιο 4, ως παραδείγματα εφαρμογής συστήματος monitoring σε microservices, για την ανάδειξη ή όχι σημείων που σχετίζονται με την ασφάλεια. Τα κριτήρια για την επιλογή μίας πρότασης για την συμπερίληψή της στο τελικό σύνολο αποτέλεσαν τα εξής:

- Η πρόταση χρονολογείται στο διάστημα 2019-2023, δεδομένου ότι οι συγκεκριμένες τεχνολογίες και ερωτήματα αφορούν ένα πολύ σύγχρονο θέμα, και τομέα έρευνας και ανάπτυξης.
- Γίνεται αναφορά στον τομέα της ασφάλειας, είτε στο επίπεδο του σχεδιασμού, είτε σε θεωρητικό επίπεδο, είτε στο σημείο της συζήτησης.
- Η πρόταση προσφέρει μία διαφορετική οπτική σε σύγκριση με τις υπόλοιπες στο κομμάτι του σχεδιασμού, της ανάλυσης, της θεωρίας ή της πρότασης για έρευνα.
- Με βάση την θεωρία που μελετήθηκε, εντοπίζονται σημεία στα οποία ο σχεδιασμός του συστήματος monitoring μπορεί να συνεισφέρει με κάποιον τρόπο όπως είναι ή με προεκτάσεις στην παρακολούθηση της κατάστασης από την πλευρά της ασφάλειας.

Η παρουσίαση των προτάσεων αυτή ακολουθεί την εξής δομή: περιγραφή γενικής ιδέας σε υψηλό επίπεδο, παράθεση λεπτομερειών υλοποίησης (όπου αυτό έχει να προσδώσει στον σκοπό της έρευνας), σχολιασμός πάνω στο θέμα της ασφάλειας με βάσει την θεωρία που έχει παρατεθεί. Στο επόμενο κεφάλαιο τα σχόλια συλλέγονται ώστε να καταλήξουν σε κάποια ευρεία συμπεράσματα, όπως και η γνώση και η εμπειρία που αποκομίστηκε κατά την εκπόνηση της έρευνας.

Η μεθοδολογία και τα βήματα που ακολουθήθηκαν φαίνονται συνολικά σχηματικά στο διάγραμμα της Εικόνας 5.1. που παρουσιάζεται αμέσως παρακάτω. Με χρώμα φαίνονται τα βασικά σκέλη που ορίζουν και την δομή του παρόντος κεφαλαίου, αλλά και τον σκελετό της έρευνας.



Εικόνα 5.1.: Διάγραμμα ροής μεθοδολογίας.

5.2 Θεωρητικό πλαίσιο *Security Monitoring*

Ξεκινώντας το κομμάτι της έρευνας κρίθηκε απαραίτητη η αναζήτηση και ορισμός ενός θεωρητικού πλαισίου γύρω από την χρήση του συστήματος monitoring για λόγους ασφαλείας. Ενώ έχει αναφερθεί και σε προηγούμενα κεφάλαια η σύνδεση των δύο, και έχουν αναλυθεί και εξεταστεί χωριστά, ο συνδυασμός τους είναι μία από τις βασικές τακτικές ασφαλείας γενικά, αλλά και στην αρχιτεκτονική των microservices. Καθώς, αν ένα σύστημα έχει προστατευτεί με τον δυνατό καλύτερο τρόπο, ποτέ κανείς δεν μπορεί να γνωρίζει αν θα δεχθεί κάποια επίθεση, αν θα υπάρξει κάποια διαρροή ή αν θα εντοπιστεί κάποιο ευαίσθητο σημείο, οπότε η ασφάλιση ενός συστήματος δεν είναι μια διαδικασία στατική στο χρόνο, αλλά πρέπει να ακολουθεί τη δυναμική φύση των τεχνολογιών πληροφορικής.

Στο παρόν σημείο κρίθηκε αναγκαία η έρευνα αναφορικά με την θεωρία, ώστε να μπορεί να υπάρξει μία βάση αρχών, οπτικών, κατευθύνσεων, στρατηγικών και μεθοδολογιών οι οποίες έχουν καταγραφεί και ακολουθούνται από τους προγραμματιστές και τους αρχιτέκτονες λογισμικού. Αυτή η βάση θα χρησιμοποιηθεί και με σκοπό να αποδείξει την σύνδεση του monitoring και του security στις εφαρμογές υλοποιημένες σε microservices, αλλά και στην συνέχεια για την αξιολόγηση των παραδειγμάτων που επιλέχθηκαν και για την τελική εξαγωγή συμπερασμάτων.

Σκοπός λοιπόν είναι η καταγραφή αυτής της βάσης αλλά και σχολίων και αναφορών σχετικά με τα δύο συστήματα στα microservices. Επισημαίνονται τόσο μεθοδολογίες, τεχνικές και τάσεις της βιομηχανίας, αλλά και εμπόδια που καθυστερούν την χρήση του monitoring στον τομέα της ασφαλείας στη σύγχρονη εποχή.

5.2.1. Μεθοδολογίες monitoring που εφαρμόζονται για σκοπούς ασφαλείας στις αρχιτεκτονικές microservices

Αναζητώντας λοιπόν μια γενική μεθοδολογία, στρατηγικές και βασικές αρχές για το πώς το monitoring μπορεί και πρέπει να εξυπηρετεί την ασφάλεια του συστήματος,

συλλέχθηκαν διάφορα σχόλια σε αναφορές και μελετήθηκαν εγχειρίδια σχετικά με την ασφάλεια των microservices. Αξίζει να σημειωθεί, ότι οι πηγές συνήθως αφορούν την ασφάλεια των microservices και υποσημειώνουν ότι το monitoring μπορεί να χρησιμοποιηθεί για να συνεισφέρει, και σε λιγότερες αναφορές εξετάζεται το monitoring και υποσημειώνεται ότι μπορεί να χρησιμοποιηθεί για σκοπούς ασφαλείας. Ωστόσο, όπως έχει αναφερθεί και νωρίτερα, το monitoring ασχολείται με την επίβλεψη της ομαλής λειτουργίας του συστήματος, οπότε κάποια μέρη μπορούν να υπονοήσουν προειδοποίηση για χτύπημα στο κομμάτι της ασφάλειας.

Στο σχετικό εγχειρίδιο του 2019 για την ασφάλεια των microservices του NIST (National Institute of Standards and Technology – Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας των ΗΠΑ) *Security Strategies for Microservices-based Application Systems* [17] το Security Monitoring αναφέρεται ως μία από τις έξι γενικές στρατηγικές που μπορούν και πρέπει να ακολουθούνται για την εξασφάλιση της ασφάλειας. Έναν χρόνο αργότερα στο εγχειρίδιο *Building Secure Microservices-based Applications Using Service-Mesh Architecture* [18] του ίδιου ινστιτούτου, αναφέρεται ως απαίτηση η εφαρμογή του monitoring για λόγους ασφαλείας, ανίχνευσης επιθέσεων και εντοπισμό παραγόντων που βάζονται, ενώ αναφέρεται και ως μία από τις οχτώ απαραίτητες περιοχές που πρέπει να ρυθμιστούν για την ασφάλεια ενός συστήματος microservices. Επιπλέον, σε σύγχρονα άρθρα που αφορούν στην ασφάλεια των microservices σημειώνεται και υπογραμμίζεται ότι η διαρκής παρακολούθηση του συστήματος (continuous monitoring) όχι μόνο αποτελεί μία τάση που σταδιακά κυριαρχεί για λόγους ασφαλείας, αλλά αποτελεί και τον τρόπο διάγνωσης προβλημάτων ασφαλείας, κάτι που το καθιστά αναπόσπαστο κομμάτι τόσο του συστήματος monitoring όσο και του σχεδιασμού που σέβεται την ασφάλεια του συστήματος [18,19,20].

Από την άλλη πλευρά, σε άλλες πηγές, επισημαίνεται η δυσκολία της εφαρμογής ενός συστήματος monitoring για λόγους ασφαλείας στα microservices λόγω της πολυπλοκότητας της αρχιτεκτονικής [8], ενώ σε πολλές περιπτώσεις το ενδιαφέρον εστιάζεται στην επίδοση του συστήματος ως η πρωταρχική αποστολή του monitoring [10,11,21], και όχι στην ασφάλεια. Παρόλα αυτά αφήνονται αιχμές συνήθως για την χρήση του monitoring για σκοπούς ασφαλείας [11].

Ποιες όμως είναι οι περιοχές που πρέπει να παρακολουθούνται; Ποια μέρη αφορούν συγκεκριμένα την ασφάλεια και όχι την επίδοση; Σε σύγκριση με τις μονολιθικές εφαρμογές, σε εφαρμογές που έχουν υλοποιηθεί σε microservices πρέπει να παρακολουθούνται όλα τα microservices που βρίσκονται σε διαφορετικούς server, χρησιμοποιούν διαφορετικές τεχνολογίες, έχουν επωμιστεί διαφορετικές λειτουργίες και επικοινωνούν συνήθως μέσω του διαδικτύου [17]. Επομένως αλλάζει όχι μόνο ο όγκος που πρέπει να παρακολουθείται, λόγω των ετερογενών τεχνολογιών και των διαφορετικών τοποθεσιών, αλλά και τα σημεία, καθώς η επικοινωνία των microservices είναι εκτεθειμένη στο διαδίκτυο και δεν συμβαίνει εσωτερικά ενός server. Επιπλέον, λόγω της εύκολης αποτυχίας και επανεκκίνησης των microservices ο διαρκής εντοπισμός τους, απαιτεί ένα δυναμικό σύστημα monitoring, καθώς οι διευθύνσεις IP που πρέπει να παρακολουθούνται μεταβάλλονται. Παρακάτω παραθέτονται κάποιες στρατηγικές και κατευθύνσεις που αφορούν το σχεδιασμό του συστήματος monitoring υπό το πρίσμα της ασφάλειας [17]:

- Το σύστημα monitoring θα πρέπει να εστιάζει και στο επίπεδο της λειτουργίας της κάθε μικροϋπηρεσίας αλλά και στα σημεία εισόδου-εξόδου τους. Σκοπός είναι η ανίχνευση, η αποστολή προειδοποιήσεων και η άμεση αντίδραση σε κάθε μη ομαλή συμπεριφορά του συστήματος. Παραδείγματα τέτοιων γεγονότων αποτελούν πιο συγκεκριμένα η επαναχρησιμοποίηση κάποιου συνθηματικού για πρόσβαση σε κάποιο microservice – token, η προσπάθεια παρεμβολής στον κώδικα και σε βάσεις δεδομένων – injection attack, σφάλματα επικύρωσης – validation errors, σφάλματα στον τύπο παραμέτρων, κατάρρευση κάποιου microservice, αποτυχία πυρήνων και άλλα προβλήματα και επιθέσεις που αποτελούν απειλή για την ασφάλεια του συστήματος. Όλα αυτά πρέπει να καταγράφονται και να αποθηκεύονται σε αρχεία, αλλά και να μπορούν να εντοπίζονται σε πραγματικό χρόνο, ώστε να μπορούν να αντιμετωπιστούν έγκαιρα, αποτρέποντας κάποια επίθεση ή ανιχνεύοντας κάποιο κενό ασφαλείας.
- Η ύπαρξη ενός κεντρικού dashboard το οποίο παραθέτει την κατάσταση των διάφορων microservices και των μεταξύ τους συνδέσεων είναι πολύ σημαντική.

Σημειώνεται ότι το συγκεκριμένο dashboard θα πρέπει να μπορεί τουλάχιστον να δείχνει κάποιες παραμέτρους ασφαλείας, όπως την αποτυχία σε περίπτωση εισαγωγής εσφαλμένων ή μη αναμενόμενων παραμέτρων και στοιχείων επικύρωσης-αίτησης πρόσβασης, τα οποία είναι προφανής ένδειξη προσπάθειας επίθεσης στο σύστημα.

- Για την παρακολούθηση του συστήματος και τον εντοπισμό μη ομαλής λειτουργίας και μη αναμενόμενων γεγονότων, απαραίτητος είναι ο ορισμός και καταγραφή των ομαλών καταστάσεων και γεγονότων – baseline. Ο ορισμός της ομαλής κατάστασης προκύπτει τόσο από τεχνολογικής άποψης (συμβατότητα, συνδέσεις, πιστοποίηση) όσο και αναφορικά με επιχειρησιακές αποφάσεις (πώς πρέπει να λειτουργεί η εκάστοτε εφαρμογή). Έχοντας καταγράψει το τι αναμένεται από ένα σύστημα και τις πιθανές αποδεκτές καταστάσεις στις οποίες μπορεί να βρεθεί, οτιδήποτε άλλο μπορεί να χαρακτηριστεί ως μη ομαλή συμπεριφορά και να ανιχνευθεί άμεσα.

Οι στρατηγικές που παρατέθηκαν είναι αρκετά γενικές και απλές: παρακολούθηση του συνόλου των microservices σε επίπεδο microservice αλλά και συνδέσεων-δικτύου και ορισμός μιας ομαλής κατάστασης για ανίχνευση επιθέσεων και μη αναμενόμενων καταστάσεων. Τονίζεται η αναγκαιότητα ύπαρξης συστήματος monitoring για την παρακολούθηση του συστήματος, όχι για την υλοποίηση τεχνικών, όπως παραδείγματος χάρη MFA (Multiple Factor Authentication). Αποτελεί το κομμάτι της επιτήρησης του συστήματος, αν και μέσω αυτής μπορούν να ανιχνευθούν κενά και προβλήματα, τα οποία να οδηγήσουν σε σχεδιασμό και αποφάσεις για το υπόλοιπο σύστημα ασφαλείας.

Συνεχίζοντας, όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο και φαίνεται και στην Εικόνα 4.1. το σύστημα monitoring βασίζεται στην συλλογή διάφορων δεδομένων, την επεξεργασία τους και την παρουσίασή τους μέσω dashboards. Επομένως, τα ερωτήματα που προκύπτει από την πλευρά της ασφάλειας είναι ποια είναι τα δεδομένα που αφορούν την ασφάλεια και πώς αυτά θα συλλεχθούν. Τα δεδομένα αυτά μπορούν να διακριθούν σε δεδομένα που αφορούν το τι συμβαίνει – logging και events, μετρικές – metrics και δεδομένα που αφορούν την κίνηση στο δίκτυο – trace [18]. Παρακάτω

παρατίθενται προτάσεις για κάθε περίπτωση [18]:

- **Logging data:** Δεδομένα που αφορούν το τι συμβαίνει, όπως διάφορα γεγονότα, με σκοπό να ανιχνευθούν διάφορες επιθέσεις και δυσλειτουργίες που μπορεί να υπονοούν κάποιες επιθέσεις. Μπορούν να διακριθούν σε τρεις περιπτώσεις.
 - **Συμβάντα – Events:** το κομμάτι των γεγονότων – events αφορά στην καταγραφή διάφορων σφαλμάτων, όπως σφάλματα επικύρωσης (validation errors), σφάλματα στην εισαγωγή παραμέτρων, εγκατάλειψη πυρήνα και καταστροφή κάποιας διεργασίας. Από την στιγμή που αυτά τα γεγονότα καταγραφούν και παρουσιαστούν, μπορεί να ανιχνευθεί κάποια προσπάθεια επίθεσης, όπως επαναχρησιμοποίηση κάποιου συνθηματικού πρόσβασης ή απόπειρα για injection attack.
 - **Αιτήματα – Requests:** Τα microservices επικοινωνούν μεταξύ τους μέσω HTTP αιτημάτων (GET, POST). Η καταγραφή αυτών των αιτημάτων και των αποκρίσεων σε αυτά, μπορούν να οδηγήσουν σε ανίχνευση αιτημάτων τα οποία δεν θα έπρεπε να έχουν γίνει, και πιθανότατα έχουν προέλθει από κάποια εξωτερική πηγή ή από κάποια μη ομαλή λειτουργία εσωτερικά. Η παρακολούθηση των επιτυχών αιτημάτων από μόνη της δεν έχει κάποιο ενδιαφέρον, ωστόσο η καταγραφή των αποτυχιών μπορεί να οδηγήσει στην ανίχνευση κάποιας επίθεσης (π.χ. αν η απόκριση σε ένα αίτημα είναι ο HTTP κωδικός 403, δηλαδή “ΑΠΑΓΟΡΕΥΕΤΑΙ” σημαίνει ότι κάποιος server έχει αρνηθεί να εκπληρώσει κάποιο αίτημα και το απορρίπτει γιατί απαγορεύεται η πρόσβαση στην συγκεκριμένη διεύθυνση. Πιθανότατα αν συμβαίνει κάτι τέτοιο να έχει γίνει απόπειρα επίθεσης.).
 - **Περιεχόμενο Μηνυμάτων – Message Content:** Η καταγραφή των μηνυμάτων αφορά κυρίως στην αποθήκευση κάποιων βασικών μεταδεδομένων, όπως η ημερομηνία και η ώρα αποστολής, η πηγή και ο προορισμός του μηνύματος (ποιο microservice ή ποιος τύπος microservice το έχει στείλει και ποιο το έχει λάβει), τι τύπου αίτημα περιέχει, και άλλες πληροφορίες. Αυτή η

καταγραφή μπορεί να βοηθήσει στην ανίχνευση της πηγής μιας επίθεσης (π.χ. εξωτερική ip ή κάποιο microservice που έχει κλωνοποιηθεί-μολυνθεί). Ωστόσο, και αυτό ισχύει σε όλες τις περιπτώσεις, χρειάζεται ιδιαίτερη προσοχή στο να μην αποθηκεύονται δεδομένα που μπορούν να χρησιμοποιηθούν δυνητικά σε κάποια επίθεση, όπως συνθηματικά πρόσβασης – tokens, και κωδικοί – passwords.

- **Μετρικές – Metrics:** Οι μετρικές αφορούν αριθμητικά μετρήσιμες μεταβλητές, οι οποίες δίνουν πληροφορίες για την λειτουργία του συστήματος. Αποτυπώνουν κομμάτι της συμπεριφοράς και της λειτουργικότητας της εφαρμογής. Κάποιες από τις βασικές μετρικές που προτείνεται να παρακολουθούνται για λόγους ασφαλείας είναι το πλήθος των αιτημάτων που πραγματοποιούνται προς κάποια χρονική μονάδα-διάρκεια, ο αριθμός των αιτημάτων που απέτυχαν λόγω σφάλματος του κώδικα, η μέση καθυστέρηση-διάρκεια εκπόνησης διεργασιών σε επίπεδο microservice αλλά και η μέση καθυστέρηση-διάρκεια για την εκπλήρωση ενός αιτήματος. Μέσα από αυτές τις μετρικές μπορεί να αποτυπωθεί μία ομαλή λειτουργία, και βάσει αυτής να εντοπίζεται η μη ομαλή συμπεριφορά. Αν η επεξεργασία ενός αιτήματος αργεί υπερβολικά ή ο αριθμός των αιτημάτων που αποτυγχάνουν είναι μεγαλύτερος από ότι στην ομαλή λειτουργία, υπάρχει ένδειξη πιθανής επίθεσης. Η παρουσίαση αυτών των μετρικών σε χρονοσειρές μπορεί να αποτυπώσει γρήγορα την κατάσταση του συστήματος γενικά και την παρούσα στιγμή ή περίοδο.
- **Παρακολούθηση Δικτύου – Tracing:** Η παρακολούθηση του δικτύου, δηλαδή των μηνυμάτων-αιτημάτων που ανταλλάσσουν τα microservices και το πλέγμα που αυτά δημιουργούν έχει μεγάλη σημασία για την ασφάλεια. Η προώθηση των στοιχείων που αφορούν την αποτύπωση αυτού του πλέγματος (δηλαδή οι κεφαλίδες-headers των μηνυμάτων που έχουν την πληροφορία του αποστολέα, του παραλήπτη, του είδους μηνύματος, και του όγκου) πρέπει να συμπεριλαμβάνεται στον σχεδιασμό της εφαρμογής. Από την πλευρά της ασφάλειας, η ομαλή λειτουργία αφορά σε επικοινωνία μεταξύ συγκεκριμένων

τύπων microservices και το φορτίο επικοινωνίας στους διάφορους διαύλους. Αν αυτή η συμπεριφορά διαταραχτεί, παραδείγματος χάρη υπάρχουν απόπειρες σύνδεσης οι οποίες απαγορεύονται ή κάποια ροή παύει ή υπερφορτώνεται, υπάρχουν υπόνοιες για ύπαρξη κάποιου ζητήματος ασφαλείας.

Σημειώνεται ότι τα παραπάνω αποτελούν τον ελάχιστο στόχο ενός συστήματος monitoring για την ασφάλεια του συστήματος αναφορικά με τα είδη δεδομένων που πρέπει να παρακολουθούνται. Επιπλέον, προκύπτει ότι η συλλογή αυτών των δεδομένων – τουλάχιστον των απαραίτητων που σημειώνονται – μπορεί να υλοποιηθεί σχετικά εύκολα [18], καθώς αυτά τα περισσότερα από αυτά τα δεδομένα είναι διαθέσιμα σε κάθε εφαρμογή ή μπορούν να υπολογιστούν χωρίς κόπο, δεδομένου ότι όλες οι σύγχρονες εφαρμογές τηρούν ένα υποτυπώδες σύστημα καταγραφής συμβάντων-κατάστασης (logging system).

5.3 Εφαρμογές συστημάτων monitoring σε microservices υπό το πρίσμα της ασφαλείας

Έχοντας παρουσιάσει ένα βασικό θεωρητικό πλαίσιο, σε αυτό το σημείο εξετάζονται ορισμένα παραδείγματα εφαρμογής συστήματος monitoring σε εφαρμογές microservices και επισημαίνονται διάφορα ζητήματα ασφαλείας: αν έχουν σχεδιαστεί ώστε να εξυπηρετούν αυτόν τον σκοπό και αν όχι γιατί, πώς συνεισφέρουν στο κομμάτι της ασφαλείας, αν κάποια μέρη που παρακολουθούνται για λόγους επίδοσης του συστήματος εμπεριέχουν πληροφορίες σχετικά με την ασφάλεια, και ποιες είναι οι ελλείψεις που παρουσιάζουν. Η παράθεση της έρευνας ακολουθεί την περιγραφή κάθε περίπτωσης με τους αντίστοιχους σχολιασμούς. Ορισμένα παραδείγματα παρουσιάζονται συνδυαστικά σε περιπτώσεις που ακολουθούν πανομοιότυπη

μεθοδολογία ή εμφανίζουν πολλά κοινά σημεία.

Επειδή τα εφαρμοσμένα παραδείγματα στην βιβλιογραφία δεν ήταν επαρκή, συμπεριλήφθηκαν τελικά και συνολικές μελέτες αναφορικά με την βιβλιογραφία τα τελευταία χρόνια. Από εκεί αντλούνται συμπεράσματα και τάσεις, όπως έχουν διατυπωθεί, αλλά και προτάσεις για αρχιτεκτονική του συστήματος secure monitoring. Στον Πίνακα 5.1. αναγράφονται τα άρθρα που μελετήθηκαν με χρονολογική σειρά, η χρονιά δημοσίευσής τους, καθώς και η αναζήτηση στην οποία εμφανίστηκαν. Η σειρά επίσης ακολουθείται στην συνέχεια στην εξέταση των άρθρων.

Δημοσίευση	Χρονιά	Αναζήτηση με τον όρο security	Αναζήτηση χωρίς τον όρο security
REAL TIME MONITORING OF SERVERS WITH PROMETHEUS AND GRAFANA FOR HIGH AVAILABILITY [10]	2019		X
CAPODAZ: A containerised authorisation and policy-driven architecture using microservices [22]	2020	X	
Securing Microservices and Microservice Architectures: A Systematic Mapping Study [19]	2020	X	
A Black-box Monitoring Approach to Measure Microservices Runtime Performance [13]	2020	X	
Conceptual Model of a Dashboard for Monitoring Microservices [11]	2020		X
Security in Microservice-Based Systems: A Multivocal Literature Review [23]	2021	X	
Detecting Cyber Security Attacks against a Microservices Application using Distributed Tracing [24]	2021	X	
Microservices in IoT Security: Current Solutions, Research Challenges, and Future Directions [20]	2021	X	
Secure microservices by monitoring behavior [25]	2022	X	
SoK: Run-time security for cloud microservices. Are we there yet? [26]	2023	X	

Πίνακας 5.1.: Επιλεγμένα άρθρα.

Στο παράδειγμα εφαρμογής των λογισμικών Prometheus και Grafana που παρουσιάστηκε και στο Κεφάλαιο 4, η δημοσίευση του 2019, *REAL TIME MONITORING OF SERVERS WITH PROMETHEUS AND GRAFANA FOR HIGH AVAILABILITY* [10], ενώ παρουσιάζονται κάποια βήματα για την αρχικοποίηση λειτουργίας του Prometheus και κάποια διαγράμματα που παράγονται με το εργαλείο Grafana, δεν δίνονται περαιτέρω πληροφορίες υλοποίησης. Βασικό στόχο αποτελεί η καλύτερη παρακολούθηση της εφαρμογής σε πραγματικό χρόνο από πλευρά επίδοσης-performance. Σαν σχόλιο για την συγκεκριμένη περίπτωση, ο συνδυασμός των συγκεκριμένων τεχνολογιών μπορεί να χρησιμοποιηθεί για λόγους ασφαλείας, και ήδη κάποιες από τις μετρικές που παρακολουθούνται μπορούν να χρησιμοποιηθούν για την παρακολούθηση της ομαλής λειτουργίας, η οποία αν διαταραχτεί μπορεί να υποδεικνύει κάποια απειλή.

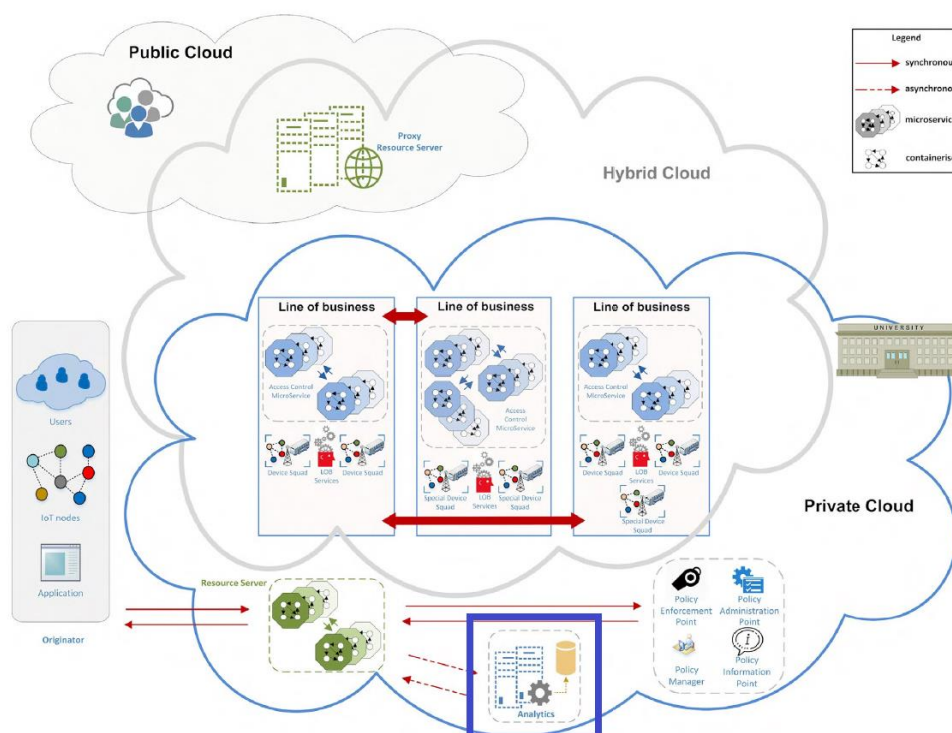


Fig. 1. The proposed CAPODAZ.

Εικόνα 5.2.: Η προτεινόμενη αρχιτεκτονική CAPODAZ. [22]

Η σημασία του monitoring στο κομμάτι της ασφάλειας επισημαίνεται σχεδόν σε όλα τα άρθρα. Ιδιαίτερα όταν αυτά εστιάζουν στην ασφάλεια και όχι στην παρακολούθηση, η αξία του monitoring υπογραμμίζεται. Σε δύο άρθρα του 2020, που κυρίως προτείνουν ένα σύστημα ασφαλείας στις συγκεκριμένες αρχιτεκτονικές, *CAPODAZ: A containerised authorisation and policy-driven architecture using microservices* [22], και παρουσιάζουν μία συνοπτική εικόνα της έρευνας που πραγματοποιείται *Securing Microservices and Microservice Architectures: A Systematic Mapping Study* [19]. Στην Εικόνα 5.2. φαίνεται η προτεινόμενη αρχιτεκτονική CAPODAZ [22], δηλαδή μια αρχιτεκτονική η οποία έχει ως βάση σχεδιασμού τις επιχειρησιακές αποφάσεις και λογική, καθοδηγούμενη από υπηρεσίες, υλοποιημένη σε containers, με την λογική εφαρμογής σε IoT εφαρμογές στο Cloud. Επισημαίνεται στο άρθρο η σημασία της ασφάλειας, και η ανάγκη ύπαρξης λειτουργίας monitoring σε πραγματικό χρόνο, αλλά και στη δομή της εικόνας, η περιοχή Analytics υπονοεί την ύπαρξη ενός συστήματος monitoring. Από την άλλη, η ανάγκη του διαρκούς-continuous monitoring – ιδιαίτερα των εσωτερικών ροών – με δεδομένα πραγματικού χρόνου, αναφέρεται ως μία από τις βασικές τακτικές ασφαλείας που εντοπίζεται στα περισσότερα σύγχρονα άρθρα στο δεύτερο άρθρο [19].

Στο άρθρο *A Black-box Monitoring Approach to Measure Microservices Runtime Performance* [13] του 2020, το κεντρικό ενδιαφέρον εστιάζεται στον υπολογισμό μετρικών και στον τρόπο με τον οποίο θα παρακολουθούνται διάφορες μεταβλητές, διαχωρίζοντάς τες σε τρία επίπεδα, όπως έχει αναλυθεί και στο Κεφάλαιο 4. Ενώ το κομμάτι της ασφάλειας δεν είναι το επίκεντρο, γίνεται αναφορά σε μετρικές ασφαλείας. Επιπλέον, η ανάλυση που πραγματοποιείται, τόσο για παρακολούθηση του επιπέδου δικτύου (TCP πρωτόκολλο), όσο του λειτουργικού συστήματος και των δεδομένων του χρήστη, μπορεί κάλλιστα να χρησιμοποιηθεί και για λόγους ασφαλείας. Η παρακολούθηση της πορείας του αιτήματος στο δίκτυο (παρακολούθηση RTT), οι κύκλοι ενέργειας του λειτουργικού συστήματος, και η αποθήκευση δεδομένων που προκύπτουν στα διάφορα APIs λόγω ενεργειών του χρήστη, εμπίπτουν στην περίπτωση των δεδομένων που προτείνεται να συλλέγονται, να παρακολουθούνται και να ελέγχονται για λόγους ασφαλείας. Επισημαίνονται ξανά τα δεδομένα παρακολούθησης

δικτύων, δεδομένα μετρικών και τα δεδομένα συμβάντων, προς αντιστοιχία των παραπάνω. Η συγκεκριμένη δημοσίευση μπορεί να χρησιμεύσει για λόγους ασφαλείας και σχετικού σχεδιασμού σε επίπεδο ανάλυσης και καθορισμού σημείων εξόρυξης δεδομένων και μετρικών, αλλά και στην μαθηματική τους επεξεργασία.

Αντίστοιχη περίπτωση αποτελεί και το τελευταίο παράδειγμα που αναλύθηκε στο Κεφάλαιο 4, την δημοσίευση του 2020 *Conceptual Model of a Dashboard for Monitoring Microservices* [11], όπου η ανάλυση μπορεί να χρησιμοποιηθεί για λόγους ασφαλείας, παρόλο που επισημαίνεται ότι δεν εστιάζει εκεί το ενδιαφέρον και ότι το εγχείρημα του security monitoring αποτελεί μία πολύπλοκη διαδικασία για τα microservices. Το προτεινόμενο πλάνο παρακολούθησης των διάφορων σημείων διεπαφής-endpoints αποτελεί και αυτό δεδομένο που μπορεί πέρα από το κομμάτι της επίδοσης να αξιοποιηθεί για την παρακολούθηση της κατάστασης ασφάλειας: πόσα αιτήματα πραγματοποιούνται και ποια από αυτά είναι αποτυχημένα. Ποια σύνδεση συγκεντρώνει τις περισσότερες αποτυχίες, και άρα είναι ύποπτη για απειλή και ενδεχομένως χρειάζεται έρευνα από πλευράς ασφαλείας.

Σε μία συστηματική μελέτη του 2021, το σύστημα monitoring εξετάζεται μαζί με το σύστημα αποθήκευσης γεγονότων (logging) ως ένας μηχανισμός ασφαλείας, άρθρο *Security in Microservice-Based Systems: A Multivocal Literature Review* [23]. Αυτός ο μηχανισμός ασφαλείας είναι υπεύθυνος για την αποθήκευση των γεγονότων και της λειτουργίας της εφαρμογής, και έπειτα χρησιμοποιώντας αυτά τα δεδομένα μπορεί να πραγματοποιηθεί ανίχνευση μη ομαλής συμπεριφοράς, όταν η εφαρμογή είναι πια διαθέσιμη και ενεργή. Το σημαντικό εδώ είναι ότι σημειώνεται τόσο η αναγκαιότητα ύπαρξης ενός συστήματος secure monitoring, όσο και η ανάγκη ύπαρξης παραπάνω βιβλιογραφίας και μελέτης πάνω στο συγκεκριμένο θέμα στον ακαδημαϊκό χώρο, καθώς ελάχιστες είναι οι αναφορές και ακόμα λιγότερα τα παραδείγματα εφαρμογής. Κάτι τέτοιο αποτυπώνει και ένα από τα προβλήματα που αντιμετώπισε η έρευνα της παρούσας διπλωματικής, ψάχνοντας σε διαφορετικό χρονικό πλαίσιο. Συγκρίνοντας και με άλλους μηχανισμούς ασφαλείας, η εφαρμογή συστήματος monitoring φαίνεται να μην κατέχει αντίστοιχο μερίδιο στην βιβλιογραφία, όπως αποτυπώνεται και στην Εικόνα 5.3..

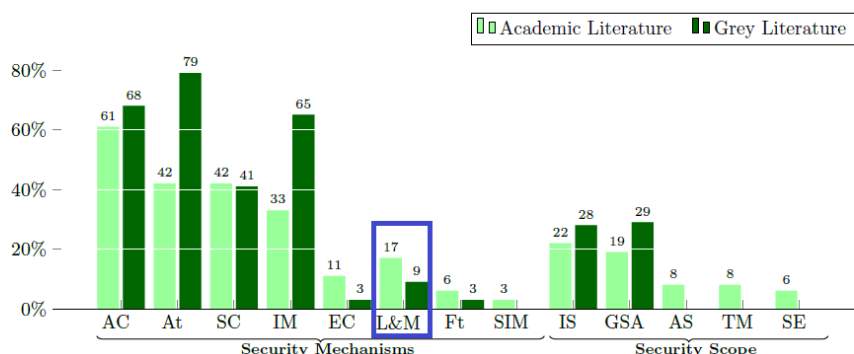


Figure 13: Security mechanisms and security scope identified in studies of academic and grey literature on microservice-based systems. The initials used were: **At** (Authorization); **IM** (Identity Management/Authentication); **AC** (Access Control), **SC** (Secure Communication); **Ft** (Filtering); **L&M** (Logging & Monitoring); **EC** (Execution Control); **SIM** (Security Information Management); **IS** (Implementation Security); **SE** (Security Evaluation); **TM** (Threat Modeling); **GSA** (General Security Architecture); **AS** (Application Security).

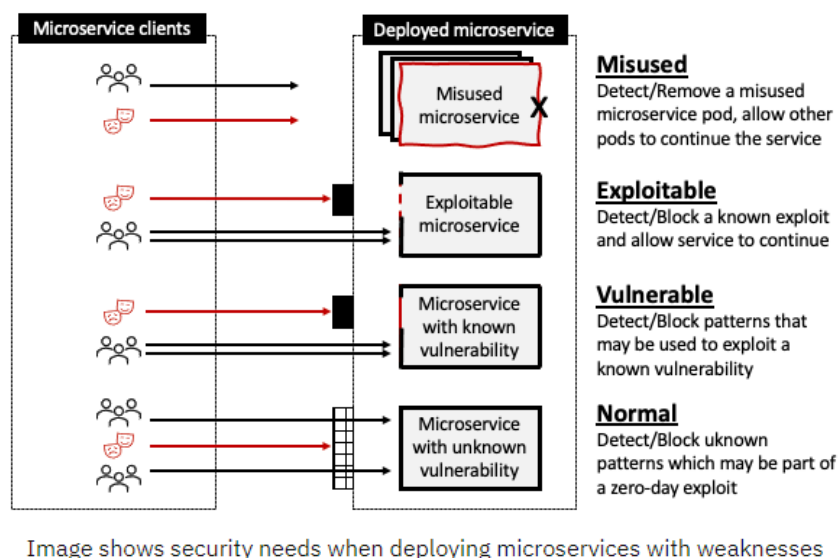
Εικόνα 5.3.: Αποτύπωση μηχανισμών και πεδίων ασφάλειας στα microservices όπως ανιχνεύονται στη βιβλιογραφία στο διάστημα 2015-2019. [23]

Στο άρθρο *Detecting Cyber Security Attacks against a Microservices Application using Distributed Tracing* [24] δημοσιευμένο το 2021 παρουσιάζεται ένα σύστημα ανίχνευσης επιθέσεων (απόπειρες για password guessing και SQL injection) σε μία εφαρμογή microservices. Η ροή δεδομένων που παρακολουθήθηκε για την διεξαγωγή των πειραμάτων ήταν αυτή της παρακολούθησης δικτύου (tracing data). Ορίζονται εντός και η έννοια της ανίχνευσης μη ομαλής συμπεριφοράς (anomaly detection) καθώς και οι τύποι επιθέσεων. Η όλη διαδικασία της ανίχνευσης εμπίπτει στο σύστημα monitoring καθώς και η ροή δεδομένων που παρακολουθείται είναι μία από τις τρεις κύριες πηγές δεδομένων για τον σχεδιασμό ενός συστήματος monitoring που αποσκοπεί στην διατήρηση της ασφάλειας. Επισημαίνεται εδώ ότι το σύστημα monitoring δεν αποτελείται μόνο από οθόνες και dashboards που σκοπό έχουν την γραφική αναπαράσταση της κατάστασης του συστήματος και διάφορων μετρικών. Συμπεριλαμβάνει επίσης και διεξαγωγή διαρκών ελέγχων καθώς και την αποστολή αυτόματων προειδοποιήσεων-alerts. Το συγκεκριμένο άρθρο αποτελεί μια στοχευμένη προσπάθεια αντιμετώπισης ενός προβλήματος ασφαλείας με τεχνικές monitoring.

Σε ακόμα μία δημοσίευση του 2021, εξετάζεται μία περίπτωση εφαρμογής των

microservices σε εφαρμογή IoT: *Microservices in IoT Security: Current Solutions, Research Challenges, and Future Directions* [20]. Το άρθρο αυτό συμπεριλαμβάνεται διότι παρουσιάζει μια προοπτική για μία ακόμα εφαρμογή των microservices, καθώς από την φύση του ο κλάδος του Internet of Things (IoT) έχει πολλά κοινά σημεία με την έννοια των κατακεταμμένων συστημάτων, και πλέον οι εφαρμογές IoT υλοποιούνται σε Cloud αρχιτεκτονικές. Στο άρθρο υπογραμμίζεται η σημασία της ασφάλειας, ωστόσο δεν δίνεται κάποια περαιτέρω λεπτομέρεια, καθώς αφορά άρθρο σύνοψης ερευνών και όχι μία πρόταση.

Σε δημοσιογραφικό άρθρο του 2022 της εταιρείας IBM με τίτλο *Secure microservices by monitoring behavior* [25], αναφέρονται πολλές προοπτικές του monitoring για χρήση στην υπηρεσία της ασφάλειας. Τονίζεται ότι κάθε microservice είναι ευάλωτο σε επιθέσεις, και επομένως η διαρκής παρακολούθηση είναι πολύ μεγάλης σημασίας, ώστε να μπορούν να ανιχνευθούν τα microservices τα οποία χρησιμοποιούνται κακόβουλα ή έχουν δεχθεί επιθέσεις. Στην Εικόνα 5.4. φαίνονται τα είδη επιθέσεων και καταστάσεις όπου η ασφάλεια των microservices έχει υποστεί κάποια ζημιά, καθώς και τις ανάγκες από πλευράς ασφάλειας σε κάθε περίπτωση και τις ενέργειες που πρέπει να πραγματοποιηθούν. Σημαντικό είναι ότι σε κάθε περίπτωση σημειώνεται η σημασία της ανίχνευσης της κατάστασης, υπονοώντας μία λειτουργία monitoring. Στο άρθρο, πέρα από λεπτομέρειες για τους τύπους απειλών, παρουσιάζεται και η χρήση ενός προϊόντος ανοιχτού κώδικα, το πρότζεκτ *Guard*, το οποίο παρακολουθεί το κάθε microservice όσον αφορά τα θέματα ασφάλειας.



Εικόνα 5.4.: Είδη επιθέσεων/απειλών στα microservices, όπως ορίζονται από την IBM. [25]

Στην πιο πρόσφατη και τελευταία μελέτη του 2023, εξετάζεται γενικότερα το κατά πόσο έχει υπάρξει πρόοδος στην ασφάλεια των microservices σε πραγματικό χρόνο. Το άρθρο *SoK: Run-time security for cloud microservices. Are we there yet?* [26] εξετάζει γενικά τους μηχανισμούς και τις στρατηγικές ασφαλείας που εμφανίζονται στην βιβλιογραφία. Όσον αφορά το ζήτημα του monitoring, η αναζήτηση που πραγματοποιήθηκε δεν σχετιζόταν ευθέως με τον όρο security, αλλά η σχέση του monitoring με το κατά πόσο είναι ευάλωτο το σύστημα. Επομένως, υπάρχουν πολλές αναφορές για παρακολούθηση του δικτύου, αλλά λίγες για security monitoring. Εδώ ανακύπτει ένα θέμα που θα συζητηθεί στο κεφάλαιο των συμπερασμάτων και έχει γίνει νύξη νωρίτερα, δηλαδή ότι η διαχωριστική γραμμή ανάμεσα στην επίδοση και στην ασφάλεια ορισμένες φορές είναι θολή, καθώς παρακολούθηση της υγείας του συστήματος είναι από μόνη της μια γενική έννοια.

6

Συμπεράσματα και Συζήτηση

Η παρούσα διπλωματική είχε ως στόχο και σκοπό την διερεύνηση του προβλήματος του security monitoring στην αρχιτεκτονική των microservices και την καταγραφή διάφορων λύσεων. Στο κεφάλαιο αυτό, έχοντας παρουσιάσει και το θεωρητικό πλαίσιο και την έρευνα που διεξήχθη, παρουσιάζονται κάποια συμπεράσματα.

Αρχικά, σαν πρώτο συμπέρασμα, σημειώνεται όπως και σε πολλές απολογιστικές συνολικές έρευνες που μελετήθηκαν ότι υπάρχει σαφής έλλειψη στην βιβλιογραφία. Ιδιαίτερα στο κομμάτι του security monitoring οι αναφορές είναι ελάχιστες και ακόμα λιγότερες είναι οι δημοσιευμένες προτάσεις για εφαρμογή κάποιας συγκεκριμένης λύσης. Οι περισσότερες αναφορές αποτελούσαν γενικές στρατηγικές μεθοδολογίες και προτάσεις για την οπτική του σχεδιασμού ενός συστήματος security monitoring. Η θεωρία ως προς το σκεπτικό αν και περιορισμένη υπάρχει, αλλά συγκεκριμένα παραδείγματα εφαρμογών είναι ελάχιστα.

Σαν δεύτερο συμπέρασμα, τονίζεται πολλαπλώς η αναγκαιότητα ύπαρξης ενός καλοσχεδιασμένου και προνοητικού συστήματος security monitoring καθώς οι διαρκείς έλεγχοι και η παρακολούθηση σε πραγματικό χρόνο είναι καίρια κομμάτια του συστήματος ασφαλείας μιας εφαρμογής. Σημειώνεται επίσης ότι καθώς αποτελεί ένα

πολύ σύγχρονο θέμα, αναφέρεται ότι υπάρχουν ανοιχτοί δρόμοι για έρευνα, και ότι η καταγραφή λύσεων και έρευνα προς την συγκεκριμένη κατεύθυνση αποτελεί ένα γενικό αίτημα και ανάγκη σήμερα.

Τελευταία σημείωση που μπορεί να συνυπολογιστεί στα συμπεράσματα, αφορά στην πολυπλοκότητα του σχεδιασμού ενός συστήματος security, ενός συστήματος monitoring και ενός συστήματος security monitoring. Η περίπλοκη αρχιτεκτονική των microservices, οι διαφορετικές τεχνολογίες που χρησιμοποιούνται, το μεγάλο πλήθος διεπαφών και επικοινωνίας, το στοίχημα του συντονισμού και της ενορχήστρωσης προσθέτουν μεταβλητές οι οποίες δεν υπήρχαν στην περίπτωση των μονολιθικών εφαρμογών. Αυτό μπορεί να βοηθήσει στην εξήγηση του γιατί δεν υπάρχουν διαθέσιμες ανοιχτές εφαρμογές τέτοιων συστημάτων.

Συνεχίζοντας, μια αισιόδοξη παρατήρηση αποτελεί ότι στα διαθέσιμα παραδείγματα που επιλέγουν να παρουσιάσουν ένα σύστημα security monitoring φαίνεται ότι οι συγκεκριμένες καθοδηγητικές γραμμές ακολουθούνται. Όμως δεν βρέθηκαν παραδείγματα που να παρουσιάζουν μια συνολική λύση σχεδιασμού ενός συστήματος security monitoring για την αρχιτεκτονική των microservices, αλλά πολύ εύστοχα ορισμένα κομμάτια. Οι περισσότερες μελέτες για συστήματα monitoring επιλέγουν να στραφούν προς την υλοποίηση ενός συστήματος που να στοχεύει στην εποπτεία της λειτουργίας και της επίδοση του συστήματος – performance monitoring, λόγω της πολυπλοκότητας του συνδυαστικού προβλήματος της παρακολούθησης της κατάστασης της ασφάλειας. Τα σημεία που υπερέχουν είναι η παρακολούθηση του δικτύου και των σημείων επικοινωνίας, και όχι η εσωτερική κατάσταση των microservices και άλλες μετρικές, όπως μη συμβατές τιμές μεταβλητών. Αυτό είναι λογικό από μία οπτική, καθώς είναι σημαντικός ο εντοπισμός της επίθεσης από κάποιον εξωτερικό παράγοντα, κάτι το οποίο θα επέλθει μέσω επικοινωνίας με κάποιο microservice, η μέσω της χειραγώγησής του, κάτι το οποίο αποτυπώνεται στο δίκτυο. Η ύπαρξη ενός κοινού θεωρητικού υπόβαθρου και στρατηγικών είναι κάτι ενθαρρυντικό για την συνέχεια της έρευνας και της ανάπτυξης.

Μία επόμενη παρατήρηση που προέκυψε τόσο από την ανάλυση της θεωρίας, όσο και

από την μελέτη των διάφορων παραδειγμάτων είναι ότι ο διαχωρισμός των εννοιών performance και security δεν είναι πάντα εύκολος και ευδιάκριτος. Σημεία παρακολούθησης για την επίδοση και τη λειτουργία ενός συστήματος μπορούν να χρησιμοποιηθούν και για ένδειξη της κατάστασης ασφαλείας. Ένα παράδειγμα εδώ αποτελεί η παρακολούθηση της καθυστέρησης-latency των microservices και της εφαρμογής: μεγάλη καθυστέρηση σημαίνει ότι υπάρχει κάποιο πρόβλημα, είτε αυτό έχει προκύψει εσωτερικά είτε από κάποια εξωτερική επίθεση. Επίσης, άλλα σημεία που παρακολουθούνται για λόγους performance μπορούν με μικρές επεκτάσεις να χρησιμοποιηθούν τελικά για λόγους ασφαλείας. Σαν παράδειγμα, αναφέρεται η παρακολούθηση της ροής του δικτύου για εντοπισμούς bottle-necks. Με μία επέκταση, το σύστημα θα μπορούσε να καταγράφει και τον αριθμό επιτυχών και αποτυχημένων αιτημάτων, και να οριστούν κατώφλια για ειδοποίηση ασφαλείας. Τόσο το θέμα του performance όσο και του security αποτελούν μέρη της αποστολής του συστήματος monitoring, και με τις ίδιες ροές δεδομένων και παρόμοιες ή ίδιες οθόνες-διαγράμματα μπορεί να ενισχυθεί τόσο το τμήμα της ασφαλείας όσο και το τμήμα των μηχανικών λογισμικού και συντήρησης.

Περνώντας στα σχόλια, το γεγονός ότι πολλές λύσεις είναι “κλειστές”, μη διαθέσιμες, μπορεί να αποτελεί γεγονός λόγω της ίδιας της ασφαλείας του συστήματος, αλλά και του γεγονότος ότι επειδή αποτελεί ένα περίπλοκο δύσκολο σύγχρονο πρόβλημα, στο οποίο πραγματοποιείται διαρκώς έρευνα και ανάπτυξη ενδοεταίρως, τα αποτελέσματα για λόγους κέρδους και ανταγωνισμού δεν είναι διαθέσιμα. Κάθε εταιρεία που αναπτύσσει μία λύση για security monitoring για την αρχιτεκτονική της, πρέπει να προφυλάξει και την λύση και τα ευαίσθητα σημεία της, αλλά και το ανταγωνιστικό πλεονέκτημα που αυτό της προσφέρει.

Όσον αφορά στον περιορισμένο αριθμό λύσεων στην ακαδημαϊκή βιβλιογραφία, μία απάντηση θα μπορούσε να είναι η δυσκολία αναπαραγωγής συστημάτων τόσο μεγάλου όγκου. Παραδείγματος χάρη, η εταιρεία Netflix σημειώνεται ότι έχει εμπεριέχει πάνω από 7000 microservices στην αρχιτεκτονική της, και καθημερινά εξυπηρετεί πάνω από

2,5 εκατομμύρια στιγμιότυπα και containers¹⁷. Η αναπαραγωγή και συντήρηση ενός τέτοιου συστήματος για ικανό χρονικό διάστημα, και η χρήση του με πραγματικές παραμέτρους, σε ακαδημαϊκά πλαίσια αποτελεί κάτι σχεδόν αδύνατο. Μεγάλο μέρος της έρευνας προκύπτει από πραγματικά προβλήματα και ανάλυση, κάτι το οποίο στον τομέα των υπολογιστών αποτελεί κανονικότητα από την αρχή της ύπαρξής τους. Επομένως, είναι δύσκολο, πέρα από το θεωρητικό κομμάτι και την διερεύνηση και αναπαραγωγή συγκεκριμένων προβλημάτων η ακαδημαϊκή βιβλιογραφία να μπορέσει να παρέχει έρευνα και προτάσεις για πραγματικές εφαρμογές ή έστω για πολύπλοκες εφαρμογές πραγματικής κλίμακας.

Ολοκληρώνοντας αυτήν την εργασία, πρέπει να σημειωθεί ότι η παρούσα έρευνα είναι αντίστοιχου εύρους με τους στόχους που τέθηκαν. Σκοπός δεν ήταν να αποτελέσει μία εξαντλητική έρευνα πάνω στο συγκεκριμένο πρόβλημα, αλλά να καταγραφεί και να οριστεί το πρόβλημα και να αποτυπωθούν η τάση και οι ελλείψεις στην σύγχρονη εποχή.

Οι ελλείψεις που σημειώθηκαν αποτελούν έναυσμα για έρευνα, καθώς η αρχιτεκτονική των microservices ανοίγει δρόμους για εξερεύνηση νέων προβλημάτων, επαναορισμό και εξέταση παλαιότερων προβλημάτων και εξυπηρέτηση απαιτήσεων που πριν κάποια χρόνια φάνταζαν αδύνατες.

¹⁷ Δημοσιογραφικό Άρθρο, 30 Ιανουαρίου 2023, link: <https://www.uptycs.com/blog/netflix-cyber-security-team-osquery-aws-cloud>

Βιβλιογραφία - Αναφορές

- [1] Johannes Thönes, **SOFTWARE ENGINEERING: Microservices**. (2015), PUBLISHED BY THE IEEE COMPUTER SOCIETY (*Συνέντευξη του James Lewis*)
- [2] Keith D. Foote, **A Brief History of Microservices**, (2021) (*Δημοσιογραφικό Άρθρο*, *Link: <https://www.dataversity.net/a-brief-history-of-microservices/>*)
- [3] T. Yarygina and A. H. Bagge, **Overcoming Security Challenges in Microservice Architectures**, (2018) *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 2018, pp. 11-20, doi: 10.1109/SOSE.2018.00011.
- [4] Pahl, C., Jamshidi, P. & Zimmermann, O., **Microservices and Containers**. (2020) In: Felderer, M., Hasselbring, W., Rabiser, R. & Jung, R. (Hrsg.), **Software Engineering 2020**. Bonn: Gesellschaft für Informatik e.V.. (S. 115–116). DOI: 10.18420/SE2020_34
- [5] Adrian Cockcroft, **The Evolution of Microservices**, (2016) (*Παρουσίαση για την εταιρία Battery Ventures - BV*)
- [6] Kennedy A. Torkura, Muhammad I.H. Sukmana, and Christoph Meinel. **Integrating Continuous Security Assessments in Microservices and Cloud Native Applications**. (2017). In *Proceedings of the 10th International Conference on Utility and Cloud Computing (UCC '17)*. Association for Computing Machinery, New York, NY, USA, 171–180. <https://doi.org/10.1145/3147213.3147229>
- [7] A. Pereira-Vale. G. Márquez. H. Astudillo and E. B. Fernandez. **Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping**. (2019). *XLV Latin American Computing Conference (CLEI)*, Panama, Panama, 2019, pp. 01-10. doi: 10.1109/CLEI47609.2019.235060.
- [8] N. Mateus-Coelho, M. Cruz-Cunha, L.G. Ferreira. **Security in Microservices**

Architectures. (2021). *Procedia Comput. Sci.*, 181, pp. 1225-1236. doi: 10.1016/J.PROCS.2021.01.320

[9] Turnbull, J. (2016). **The Art of Monitoring**

[10] Arun Kumar K., Vinutha B. S., Vinayaditya B. V. (2019). **REAL TIME MONITORING OF SERVERS WITH PROMETHEUS AND GRAFANA FOR HIGH AVAILABILITY.** In *International Research Journal of Engineering and Technology (IRJET)* (Vol. 6, Issue, 4, pp. 5093-5096)

[11] Utomo, P., & Falahah, F. (2020). **Conceptual Model of a Dashboard for Monitoring Microservices.** In *EAI Endorsed Transactions on Cloud Systems* (Vol. 6, Issue 18, p. 166285). European Alliance for Innovation n.o. <https://doi.org/10.4108/eai.7-9-2020.166285>

[12] WHITE PAPER. (2016). **The Five Principles of Monitoring Microservices.** (Από τα άρθρα της εταιρίας Sysdig)

[13] Brondolin, R., & Santambrogio, M. D. (2020). **A Black-box Monitoring Approach to Measure Microservices Runtime Performance.** In *ACM Transactions on Architecture and Code Optimization* (Vol. 17, Issue 4, pp. 1–26). Association for Computing Machinery (ACM). <https://doi.org/10.1145/3418899>

[14] Turnbull, J. (2019). **Monitoring With Prometheus.**

[15] Chawla, H., & Kathuria, H. (2019). **Building Microservices Applications on Microsoft Azure.** Apress. <https://doi.org/10.1007/978-1-4842-4828-7>

[16] Scott Sweetland. (2017) **Paper SAS472-2017: Using SAS® Viya™ Microservices Logging for Performance Analysis of SAS® Visual Analytics 8.1 Reports** (Από τα άρθρα της εταιρίας SAS)

- [17] Chandramouli, R. (2019). **Security strategies for microservices-based application systems.** *National Institute of Standards and Technology*. <https://doi.org/10.6028/nist.sp.800-204>
- [18] Chandramouli, R., & Butcher, Z. (2020). **Building secure microservices-based applications using service-mesh architecture.** *National Institute of Standards and Technology*. <https://doi.org/10.6028/nist.sp.800-204a>
- [19] Hannousse, Abdelhakim & Yahiouche, Salima. (2020). **Securing Microservices and Microservice Architectures: A Systematic Mapping Study** [abs/2003.07262].
- [20] Driss, M., Hasan, D., Boulila, W., & Ahmad, J. (2021). **Microservices in IoT Security: Current Solutions, Research Challenges, and Future Directions.** *In Procedia Computer Science* (Vol. 192, pp. 2385–2395). Elsevier BV. <https://doi.org/10.1016/j.procs.2021.09.007>
- [21] Barakat, Saman. (2017). **Monitoring and Analysis of Microservices Performance.** *Journal of Computer Science and Control Systems*. 10. 19-22.
- [22] Kallergis, D., Garofalaki, Z., Katsikogiannis, G., & Douligeris, C. (2020). **CAPODAZ: A containerised authorisation and policy-driven architecture using microservices.** *In Ad Hoc Networks* (Vol. 104, p. 102153). Elsevier BV. <https://doi.org/10.1016/j.adhoc.2020.102153>
- [23] Pereira-Vale, A., Fernandez, E. B., Monge, R., Astudillo, H., & Márquez, G. (2021). **Security in microservice-based systems: A Multivocal literature review.** *In Computers & Security* (Vol. 103, p. 102200). Elsevier BV. <https://doi.org/10.1016/j.cose.2021.102200>
- [24] Jacob, S., Qiao, Y., & Lee, B. (2021). **Detecting Cyber Security Attacks against a Microservices Application using Distributed Tracing.** *In Proceedings of the 7th International Conference on Information Systems Security and Privacy*. 7th

International Conference on Information Systems Security and Privacy. SCITEPRESS - Science and Technology Publications. <https://doi.org/10.5220/0010308905880595>

[25] Hadas, D., Maximilien, M., Brown, M., & Schweigert, P. (2022) **Secure microservices by monitoring behavior**. (Δημοσιογραφικό άρθρο της εταιρίας IBM, link: <https://developer.ibm.com/articles/secure-microservices-by-monitoring-behavior/>)

[26] Minna, F., & Massacci, F. (2023). **SoK: Run-time security for cloud microservices. Are we there yet?** In *Computers & Security* (Vol. 127, p. 103119). Elsevier BV. <https://doi.org/10.1016/j.cose.2023.103119>

Εικόνες

Κεφάλαιο 2

Εικόνα 2.1.: Τεχνολογία microservices ως υποσύνολο σχεδιαστικών λύσεων. [3]

Εικόνα 2.2.: Η μονολιθική προσέγγιση ενός συστήματος online πληρωμών. [5]

Εικόνα 2.3.: Η προσέγγιση της αρχιτεκτονικής microservices του αντίστοιχου με την Εικόνα 1.2. συστήματος online πληρωμών. [5]

Εικόνα 2.4.: Μονολιθική προσέγγιση σχεδιασμού ψηφιακού συστήματος PetClinic. [6]

Εικόνα 2.5.: Αντίστοιχη προσέγγιση με microservices προσέγγιση για σχεδιασμό ψηφιακού συστήματος PetClinic της Εικόνας 1.4. [6]

Κεφάλαιο 3

Εικόνα 3.1.: Η αλλαγή στην προσέγγιση της αρχιτεκτονικής από μονολιθική σε κατανεμημένη, επιφέρει αλλαγές στα σημεία της εφαρμογής που εγείρουν ζητήματα ασφάλειας. [3]

Εικόνα 3.2.: Γενική απεικόνιση πρωτοκόλλου MTLS (συμπεριλαμβάνεται και η αρχική λεζάντα για την επεξήγηση του πρωτοκόλλου). [3]

Εικόνα 3.3.: Θεματική των άρθρων που μελετήθηκαν από τους Pereira-Vale et al. σε σχέση με την προοπτική μιας ενδεχόμενης επίθεσης. [7]

Εικόνα 3.4.: Πλήθος άρθρων ανά χρονιά, όπως αυτά επιλέχθηκαν και μελετήθηκαν από τους Pereira-Vale et al. το 2019. [7]

Εικόνα 3.5.: Περιοχή ενδιαφέροντος των άρθρων που μελετήθηκαν από τους Pereira-Vale et al. σε σχέση με το είδος μιας ενδεχόμενης επίθεσης. [7]

Κεφάλαιο 4

Εικόνα 4.1.: Πλαίσιο τρόπου λειτουργίας ενός συστήματος monitoring. [9]

Εικόνα 4.2.: Η θέση των containers συγκριτικά με το project και την υλοποίηση των microservices (λεζάντα όπως στο πρωτότυπο στην εικόνα). [11]

Εικόνα 4.3.: Οθόνη monitoring για την παρακολούθηση μετρικών του server της εφαρμογής με χρήση τεχνολογιών Prometheus και οπτικοποίηση μέσω εργαλείου Grafana. [10]

Εικόνα 4.4.: Βασικό σχεδιάγραμμα ροής προτεινόμενου συστήματος monitoring. [11]

Εικόνα 4.5.: Dashboards καταστάσεων: (α) προτεινόμενο dashboard για κύρια οθόνη monitoring της συνολικής κατάστασης των microservices, (β) προτεινόμενο dashboard για κύρια οθόνη κατάστασης των επιμέρους endpoints. [11]

Εικόνα 4.6.: Μέρος του dashboard για την αναλυτική παρουσίαση των τεσσάρων διαφορετικών παραμέτρων-μετρικών που συλλέγονται για ένα επιλεγμένο endpoint (failed requests, server requests, server responses time, availability). Η παρουσίαση γίνεται σε χρονοσειρές, με διαφορετικούς τύπους γραφημάτων ανά περίπτωση παραμέτρου (δεν συμπεριλαμβάνονται οι λεπτομέρειες που παρέχονται σε μορφή κειμένου, βλ. Πρωτότυπο). [11]

Εικόνα 4.7.: Ροή δεδομένων (συμπεριλαμβάνετε η λεζάντα του πρωτότυπου) [13]

Εικόνα 4.8.: Η αρχιτεκτονική δομή του συστήματος Prometheus και των διάφορων μερών του.¹⁸

Κεφάλαιο 5

Εικόνα 5.1.: Διάγραμμα ροής μεθοδολογίας.

Εικόνα 5.2.: Η προτεινόμενη αρχιτεκτονική CAPODAZ. [22]

Εικόνα 5.3.: Αποτύπωση μηχανισμών και πεδίων ασφάλειας στα microservices όπως ανιχνεύονται στη βιβλιογραφία στο διάστημα 2015-2019. [23]

Εικόνα 5.4.: Είδη επιθέσεων/απειλών στα microservices, όπως ορίζονται από την IBM. [25]

¹⁸Επίσημη σελίδα του λογισμικού Prometheus: <https://prometheus.io/>

Πίνακες

Κεφάλαιο 5

Πίνακας 5.1.: Επιλεγμένα άρθρα.