ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ασφάλεια υποδομών, με έμφαση στην αντιμετώπιση φυσικών, διαδικτυακών και μικτών απειλών

Ιωάννης Καραγιάννης
2022201502009

Επιβλέπων:

Νικόλαος Τσελίκας
Αναπληρωτής Καθηγητής

Τρίπολη, Ιούλιος, 2021

Πρόλογος

Η παρούσα Διπλωματική Εργασία εκπονήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος σπουδών με τίτλο «Επιστήμη των Υπολογιστών» του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Πελοποννήσου, υπό την επίβλεψη του κ. Νικόλαου Τσελίκα.

Αποτελεί την έγγραφη συλλογή γνώσεων μέσω των σπουδών μου στο μεταπτυχιακό πρόγραμμα και πιο συγκεκριμένα το αντικείμενο της επιστήμης των Υπολογιστών.

Μέσω του Πανεπιστημίου Πελοποννήσου, μου δόθηκε η δυνατότητα να γνωρίσω ένα μεγάλο εύρος τεχνολογιών και να ανοίξω τους ορίζοντές μου πάνω στον τομέα της Πληροφορικής.

Με βάση τις γνώσεις που απέκτησα αλλά και τη μελέτη ερευνητικών αλλά και εμπορικών εφαρμογών, έκανα μια εκτίμηση των πρόσφατων τεχνολογικών προσεγγίσεων για την ασφάλεια υποδομών. Έπειτα, με τη βοήθεια του τεχνολογικού υπόβαθρου που μου προσφέρθηκε από το μεταπτυχιακό πρόγραμμα, έφτασα σε μία σειρά συμπερασμάτων, προτάσεων αλλά και υλοποίησης επεκτάσεων για την πληρέστερη κάλυψη των ήδη υπαρχουσών λύσεων.

Παρά το γεγονός ότι υπάρχουν διαδεδομένες λύσεις στην αγορά, η παρούσα εργασία δίνει μια άλλη διάσταση, κυρίως αρχιτεκτονικά αλλά και τεχνολογικά η οποία δεν απαντάται σε κάποια από τις διαθέσιμες λύσεις. Η εργασία βασίστηκε στο έργο FINSEC (Financial Security), ένα έργο το οποίο χρηματοδοτήθηκε από την Ευρωπαϊκή Ένωση στα πλαίσια του προγράμματος Horizon H2020, αλλά και το επιστημονικό κείμενο που προέκυψε μέσω του έργου και παρουσιάστηκε στο συνέδριο ESORICS (Σεπτέμβριος 2019 Λουξεμβούργο).

Σκοπός της εργασίας είναι αρχικά να δώσει στον αναγνώστη την απαραίτητη τεχνική γνώση των τεχνολογιών που χρησιμοποιήθηκαν ώστε, το έγγραφο να μπορεί να χρησιμοποιηθεί αυτόνομα και σαν ένας οδηγός για την υποστήριξη αντίστοιχων συστημάτων. Επίσης παρουσιάζονται οι διαθέσιμες λύσεις αλλά και το τεχνολογικό τους υπόβαθρο προβάλλοντας παράλληλα τις διαφορές με την υλοποίηση της παρούσας εργασίας. Οι τροποποιήσεις που έγιναν στην υλοποίηση αλλά και στην αρχιτεκτονική παρουσιάζονται μετά την ανάλυση των υφιστάμενων έργων.

Η μελέτη στηρίχθηκε σε επιστημονικά κείμενα, επίσημους οδηγούς αλλά και πρακτική εξάσκηση πάνω στην κάθε εργαλείο/τεχνολογία με σκοπό την επιβεβαίωση ότι μπορεί να δώσει λύση σε συγκεκριμένα προβλήματα. Ο αναγνώστης μπορεί να ανατρέξει στη βιβλιογραφία που παρατίθεται για επιπλέον εξοικείωση αλλά και κατανόηση των εννοιών που περιγράφονται.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Τσελίκα καθώς επίσης και τα μέλη της οικογένειάς μου Γεώργιο, Γεωργία και Χρήστο.

Συντομογραφίες

API	Application Programming Interface
CRUD	Create-Read-Update-Delete
CVSS	Common Vulnerability Scoring System
JSON	Business Continuity Management System
JWT	Business Continuity Plan
LESS	Leaner Style Sheets
REST	Representational state transfer
SASS/SCSS	Syntactically Awesome Stylesheets
SECaaS	Security-as-a- Service
SME	Small and Medium-Sized Enterprises
STIX	Structured Threat Information eXpression
TAXII	Trusted Automated eXchange of Indicator Information
TI	Threat Intelligence
TTP	Tactics Techniques Procedures
XML	Competent Authorities
YAML	Committee of European Banking Supervisors

Περίληψη

Διαχρονικά ένα πολύ μεγάλο κομμάτι του προϋπολογισμού κρατών, τραπεζών, εργοστασίων, ιδιωτικών οργανισμών αλλά και μικρών ιδιωτικών επιχειρήσεων διοχετεύεται στην κατάλληλη φύλαξη υποδομών είτε κτηριακών ή υλικοτεχνικών (servers, ATMs, καλωδίωση κλπ.).

Ήδη, η τεχνολογία έχει βελτιώσει κατά πολύ το επίπεδο ασφάλειας των υποδομών αυτών είτε πρόκειται για φυσικές είτε για απειλές κυβερνοχώρου. Συστήματα ασφαλείας δικτύων, αισθητήρες για ανίχνευση οποιασδήποτε ανωμαλίας καθώς επίσης και έμψυχο δυναμικό βρίσκονται καθημερινά στη μάχη διατήρησης της λειτουργίας των υποδομών, την αποτροπή ζημιών αλλά και τη διαφύλαξη σημαντικών και απόρρητων πληροφοριών και εγγράφων. Οι υποδομές αυτές είναι ευάλωτες σε φυσικές καταστροφές, τρομοκρατικά χτυπήματα βανδαλισμούς, λεηλασίες, υφαρπαγής χρημάτων αλλά και άλλων πόρων. Το περιβάλλον είναι πολύπλοκο, με πολλές πτυχές και όλα εξαρτώνται από τη δομή του οργανισμού, τις ανάγκες, τα περιουσιακά στοιχεία (assets) καθώς επίσης και τη φύση τους και την κρισιμότητα φύλαξης τους.

Στην παρούσα εργασία γίνεται μία προσπάθεια ενοποίησης των φυσικών (physical) απειλών με τις απειλές διαδικτύου (cyber) καθώς επίσης και ενός συστήματος επικοινωνίας μεταξύ οργανισμών για την έγκαιρη προειδοποίηση και αποφυγή επιθέσεων. Η ιδέα βασίστηκε στο έργο FINSEC H2020, ωστόσο τροποποιήθηκε τεχνολογικά αλλά και αρχιτεκτονικά για να καλύψει ένα μεγαλύτερο εύρος τομέων (domains).

Η επικοινωνία μεταξύ των components του συστήματος πραγματοποιείται με τη βοήθεια είτε Restful APIs είτε του Apache Kafka με χρήση προκαθορισμένων καναλιών επικοινωνίας εντός ενός οργανισμού αλλά και μεταξύ οργανισμών. Η διεπαφή χρήστη είναι μία επέκταση του FINSEC Dashboard ενός προγράμματος ανοιχτού κώδικα για την υποστήριξη των αναγκών του FINSEC. Για να επεκταθεί το συγκεκριμένο εργαλείο, χρειάστηκαν γνώσεις Angular, αλλά και του Laravel PHP framework. Για τις ανάγκες της εργασίας δημιουργήθηκε μία mobile εφαρμογή, η οποία αναπτύχθηκε σε Ionic framework. Τέλος το κάθε component της τελικής λύσης αποτελεί μία αυτόνομη οντότητα, η οποία αναπτύχθηκε ως microservice με χρήση Docker containers.

Λέξεις-Κλειδιά

Υποδομές, Ασφάλεια, Φυσικές απειλές, Διαδικτυακές απειλές, STIX, Kafka, GraphQL, Angular, Laravel.

Abstract

Over time a respectful budget part, is utilized to support the security of critical infrastructures. This fact is true for public sector, government, financial and private sector organizations either big or small.

Technology has already provided a vital support and a plethora of solutions in this direction. With the sensor integration physical threats are handled. Additionally, mature software solutions, provide full support for cyber threats. However, there is always room for improvement especially, when it comes to secure vital and private information, expensive or even dangerous equipment. Assets are susceptible to natural disaster, terrorism, vandalism, looting or heist. The complicated environment as well as the variety of possible scenarios, make the security of each organization even harder. A careful analysis of its assets, the criticality of them as well the possible threats is critical.

In the current document we try to provide a solution to unify physical and cyber threats and provide a common platform for collaboration between organizations of the system. The idea is based on FINSEC H2020 project, however it is adjusted both technologically and architecturally to support a broader range of domains (other than financial).

The communication between system components is achieved using Restful APIs and Apache Kafka. Communication channels have been setup for inter-organization communication. The user interface is an extension of the FINSEC Dashboard. An open-source component developed to cover the FINSEC needs. To successfully extend it, both Angular and Laravel Framework familiarity was necessary. For the scope of this dissertation, an Ionic mobile app was developed to support the same needs detected for the web application. Each component is an autonomous entity and was developed as a microservice using Docker containers.

Keywords

Infrastructure, Security, Physical Threats, Cyber Threats, STIX, Kafka, GraphQL, Angular, Laravel.

Περιεχόμενα

Ευρετήριο εικόνων

Ευρετήριο πινάκων

Κεφάλαιο 1: Εισαγωγή

Στο κεφάλαιο αυτό θα εξετάσουμε τις ήδη υπάρχουσες εμπορικές, αλλά και ερευνητικές λύσεις, οι οποίες προσφέρουν ένα πλαίσιο προστασίας υποδομών. Θα δούμε επίσης και γνωστά εργαλεία, τα οποία προσφέρουν οπτικοποίηση των υποδομών ενός οργανισμού, αλλά και την επίβλεψη της ορθής λειτουργίας των συστημάτων/προγραμμάτων του οργανισμού.

Θα ξεκινήσουμε με τις επιλογές σε ασφάλεια υποδομών και έπειτα θα εξετάσουμε τα εργαλεία οπτικοποίησης δεδομένων, τα οποία στοχεύον στην επίβλεψη των συστημάτων και της εύρυθμης λειτουργίας τους εντός του οργανισμού.

Επίσης θα παρουσιάσουμε το πρόβλημα το οποίο προσπαθούμε να λύσουμε και θα παρουσιαστεί και η δομή του κειμένου.

1.1 Παρουσίαση Προβλήματος

Η ασφάλεια υποδομών αποτελεί ένα από τα κύρια προβλήματα οποιοδήποτε οργανισμού. Κτηριακές δομές, υλικοτεχνική υποδομή αλλά και αρχεία, συναλλαγές μπορούν όλα να θεωρηθούν υποδομές για τις οποίες απαιτείται μία μελέτη απειλών και ένας μηχανισμός αντιμετώπισής τους.

Σκοπός της εργασίας είναι η μελέτη, η πρόταση αλλά και η υλοποίηση λύσεων για μια πιο αποδοτική προστασία υποδομών.

Η λύση θα πρέπει να λαμβάνει υπόψιν όλες τις πιθανές απειλές, να είναι επεκτάσιμη και να παρέχει μία κατανοητή διεπαφή χρήστη που θα βοηθήσει έμπειρους χρήστες να αντιμετωπίσουν τις απειλές αυτές.

Επίσης, το έγγραφο θα πρέπει να είναι αυτόνομο, κάτι που σημαίνει ότι ο αναγνώστης έχει όλη τη βασική πληροφορία για να κατανοήσει τις τεχνολογίες, την αρχιτεκτονική αλλά και τον τρόπο υλοποίησης και πώς αυτός έλυσε κάποιες από τις αρχικές απαιτήσεις.

Τέλος, στα πλαίσια της εργασίας και για την ικανοποίηση των αναγκών που προέκυψαν, θα πρέπει το παρόν έγγραφο να κάνει σαφές τις προσαρμογές που έγιναν σε γνωστά πρωτόκολλα.

1.2 Δομή Εργασίας

Στο παρόν κεφάλαιο δίνονται πληροφορίες για το κύριο πρόβλημα, αλλά και τις ήδη υπάρχουσες λύσεις.

Στο Κεφάλαιο 2: Τεχνολογίες, δίνονται λεπτομέρειες για τις τεχνολογίες που χρησιμοποιήθηκαν ή είναι απαραίτητες για την κατανόηση προτάσεων που έγιναν στα πλαίσια της εργασίας αυτής.

Στο Κεφάλαιο 3: Ανάλυση Απαιτήσεων, γίνεται αναφορά στις απαιτήσεις χρήστη αλλά και συστήματος. Οι απαιτήσεις αυτές βοηθούν στην κατανόηση των λεπτομερειών υλοποίησης αλλά και τις ανάγκες πρότασης νέας προσέγγισης για μελλοντική εξέλιξη της πλατφόρμας.

Στο Κεφάλαιο 4: Υλοποίηση, παρουσιάζονται οι προσαρμογές στην αρχιτεκτονική αλλά και η τελική υλοποίηση, ενώ στο Κεφάλαιο 5: Συμπεράσματα, παρουσιάζονται τα συμπεράσματα αλλά και τα επόμενα βήματα που θα οδηγούσαν σε περαιτέρω ολοκλήρωση της προσέγγισης της εργασίας αυτής.

1.3 MITIGATE

Το MITIGATE αποτέλεσε ένα έργο ερευνητικού χαρακτήρα το οποίο χρηματοδοτήθηκε από την Ευρωπαϊκή ένωση στα πλαίσια του προγράμματος Horizon H2020. Η διάρκεια του ήταν τρία χρόνια από το 2015 έως το 2018.

Το έργο αναλύει την ασφάλεια υποδομών και τον υπολογισμό του ρίσκου σε ναυτιλιακές υποδομές. Η βασική του ιδέα είναι ο ορισμός αλυσίδων (supply chain) οι οποίες συνθέτουν μία ναυτιλιακή εγκατάσταση. Οι αλυσίδες αυτές είναι η ένωση υποδομών τα οποία ονομάζονται assets. Μία καινοτόμα προσέγγιση του MITIGATE ήταν η αναγνώριση κόμβων οι οποίοι θα μπορούσαν να αποτελούν ολόκληρους οργανισμούς, επιμέρους συστήματα ή λειτουργίες. Οι κόμβοι αυτοί, στα πλαίσια του έργου, συνεργάζονται (collaborate) προκειμένου να πετύχουν έναν πιο αξιόπιστο υπολογισμό ρίσκου, καθώς επίσης και να αντιμετωπίσουν σε συνεργασία απειλές στα πρώτα στάδια ανίχνευσής τους.

Το MITIGATE χρησιμοποίησε τη μαθηματικό μοντέλο g-MRSA το οποίο ουσιαστικά είναι μία μεθοδολογία για τον καθορισμό απειλών και τον υπολογισμό ρίσκου σε μία αλυσίδα υποδομών.

Για τον υπολογισμό του ρίσκου λαμβάνονται υπ' όψη ιστορικά στοιχεία όπως επιθέσεις, social media posts, ο τύπος του κάθε asset (και πώς μπορεί κάποιος να αποκτήσει πρόσβαση) καθώς επίσης και οι γνωστές ευπάθειες (vulnerabilities) ενός asset. Τέλος ένα ακόμα στοιχείο που εκτιμάται είναι η επίπτωση (impact) μίας επίθεσης και το πόσο κρίσιμη είναι.

Τελικά το ρίσκο για κάθε asset προκύπτει από το γινόμενο του threat level με το vulnerability αλλά και το threat level.

$$RL = TL * VL * IL$$

Ειδικά για το επίπεδο ευπάθειας το MITIGATE έχει βασιστεί στο πρότυπο CVSS [6]. Τα επίπεδα (scores) που προκύπτουν για κάθε μια από τις ευπάθειες έχουν μεταφραστεί σε κλίμακες του MITIGATE.

Πέραν του ρίσκου μίας συγκεκριμένης υποδομής ή συστήματος, υπάρχει και το ρίσκο της αλυσίδας που έχει καθοριστεί και απαρτίζεται από τα επιμέρους assets.

Τα λογικά βήματα για τον υπολογισμό ρίσκου στο MITIGATE είναι:

- Βήμα 0: Ανάλυση του σκοπού μίας αλυσίδας ναυτιλιακής υποδομής
- Βήμα 1: Ανάλυση της ίδιας της αλυσίδας και ορισμός επιμέρους υποδομών ή συστημάτων που την απαρτίζουν.
- Βήμα 2: Καθορισμός Απειλής
- Βήμα 3: Υπολογισμός πιθανότητας εκπλήρωσης της απειλής
- Βήμα 4: Καθορισμός Επιπτώσεων
- Βήμα 5: Υπολογισμός ρίσκου κάθε επιμέρους συστήματος/asset
- Βήμα 6: Υπολογισμός ρίσκου όλης της αλυσίδας

Το MITIGATE παρέχει επίσης μία διεπαφή χρήστη για την ανίχνευση ρίσκων στις ναυτιλιακές υποδομές, την πυροδότηση υπολογισμών αλλά και την οπτικοποίηση της υποδομής αλλά και των κινδύνων.

Περισσότερες πληροφορίες για το MITIGATE μπορείτε να βρείτε (cross reference προς mitigate link).

1.4 FINSEC

Το έργο FINSEC είχε διάρκεια τρία έτη από το 2018 έως το 2021. Πρόκειται για ένα καινοτόμο έργο το οποίο ενοποιεί φυσικές αλλά και διαδικτυακές απειλές στον τομέα των χρηματοπιστωτικών ιδρυμάτων. Με την τεχνολογία blockchain και πιο συγκεκριμένα το Hyperledger Fabric, επιτρέπει την επικοινωνία μεταξύ οργανισμών με γρήγορο και ασφαλή τρόπο.

Η αρχιτεκτονική του FINSEC φαίνεται στην Εικόνα 1. Μπορούμε να διακρίνουμε 3 επίπεδα στην αρχιτεκτονική αυτή.

Το Edge tier όπου γίνεται συλλογή δεδομένων από αισθητήρες, log αρχεία, το εσωτερικό δίκτυο, συστήματα πληρωμών ή και άλλα τραπεζικά συστήματα.

Στο δεύτερο επίπεδο βρίσκεται το Data Tier. Εκεί αποθηκεύονται δεδομένα για την υποδομή του οργανισμού, δεδομένα από τις συσκευές κατώτερου επιπέδου, αλλά και άλλα ιστορικά δεδομένα χρήσιμα για την επεξεργασία και την εξαγωγή συμπερασμάτων.

Στο τρίτο επίπεδο έχουμε τα services. Εκεί γίνεται η αναγνώριση ανωμαλιών, η πρόβλεψη απειλών, ο υπολογισμός ρίσκων, η αναγνώριση παραβάσεων κανονισμών αλλά και ο εντοπισμός επιθέσεων.

Έξω από τον πυρήνα του FINSEC βρίσκονται δύο components. Το ένα είναι το FINSEC Dashboard, το οποίο αποτελεί την οπτική πρόσβαση στην πλατφόρμα. Για τις ανάγκες της τρέχουσας εργασίας, θα επεκτείνουμε το component αυτό ώστε να φτάσουμε στην ολοκλήρωση των απαιτήσεων που θα δούμε και στο Κεφάλαιο 3: Ανάλυση Απαιτήσεων.

Το άλλο component εκτός του πυρήνα FINSEC είναι το Collaborative DLT το οποίο αποτελεί το κομμάτι το οποίο είναι υπεύθυνο για την επικοινωνία μεταξύ οργανισμών [1].

Παράλληλα σε όλα τα επίπεδα του FINSEC υποστηρίζεται η καταγραφή (logging) των λειτουργιών, ενώ τα επιμέρους τμήματα αλληλοεπιδρούν ως αυτόνομες οντότητες με τη βοήθεια των τεχνολογιών Docker και Kubernetes.

Για τις ανάγκες του έργου αλλά και του χρηματοπιστωτικού τομέα αναπτύχθηκε το μοντέλο μεταφοράς και αποθήκευσης δεδομένων FINSTIX (cross reference here).

Πρόκειται για μία επέκταση του STIX με τις απαιτούμενες προσαρμογές για όλα τα services αλλά και τα δεδομένα από τους αισθητήρες στο κατώτερο επίπεδο.

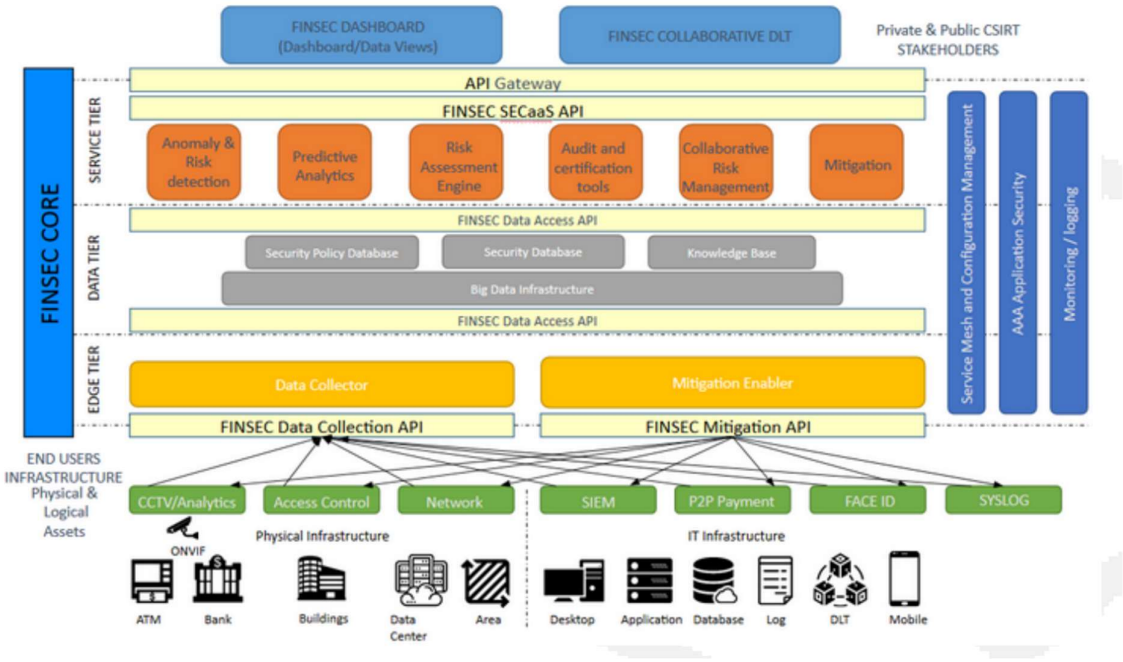
Επίσης στο FINSTIX βασίστηκαν η δημιουργία κανόνων για την ανίχνευση ανωμαλιών, τη δημιουργία events αλλά και την πρόβλεψη επιθέσεων.

Διάφορα εργαλεία χρησιμοποιήθηκαν για να φτιαχτεί καθένα από τα blocks/services. Ενδεικτικά για το Collaborative Risk Assessment, έγινε επέκταση του MITIGATE και για τις λειτουργίες αλλά και για τον τύπο δεδομένων ώστε να είναι συμβατά με το FINSTIX μοντέλο δεδομένων.

Για την επικοινωνία με την πλατφόρμα, στήθηκε ένας AAA μηχανισμός, με το Istio Service mesh. Τα HTTP requests περνάνε μέσα από επαλήθευση με τη βοήθεια JSON Web Tokens (JWT).

Τέλος για την ασφάλεια του συστήματος και πάλι με τη βοήθεια της τεχνολογίας Kubernetes, στήθηκε ένας μηχανισμός αποσφαλμάτωσης και εντοπισμού ευπαθειών πηγαίου κώδικα. Τα εργαλεία που χρησιμοποιήθηκαν για το σκοπό αυτό ήταν το Sonarqube (cross reference here), το Dependency check αλλά και το Anchore.

Για περισσότερες πληροφορίες, webinars, tutorials και άλλο υλικό μπορείτε να επισκεφθείτε τη σελίδα του FINSEC.



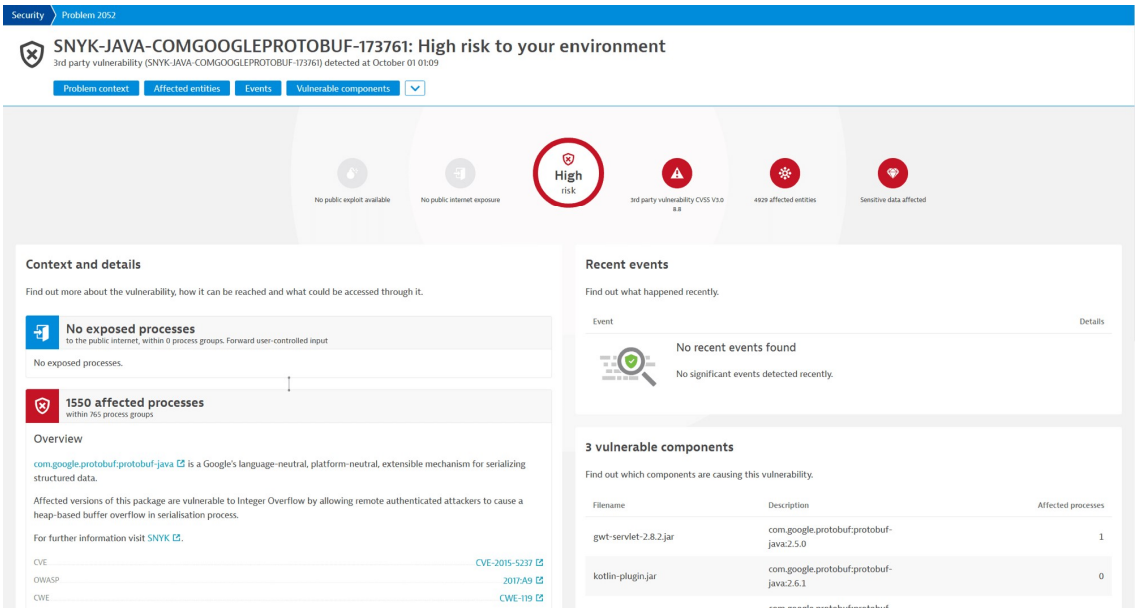
Εικόνα 1: Αρχιτεκτονική FINSEC

1.5 Dynatrace

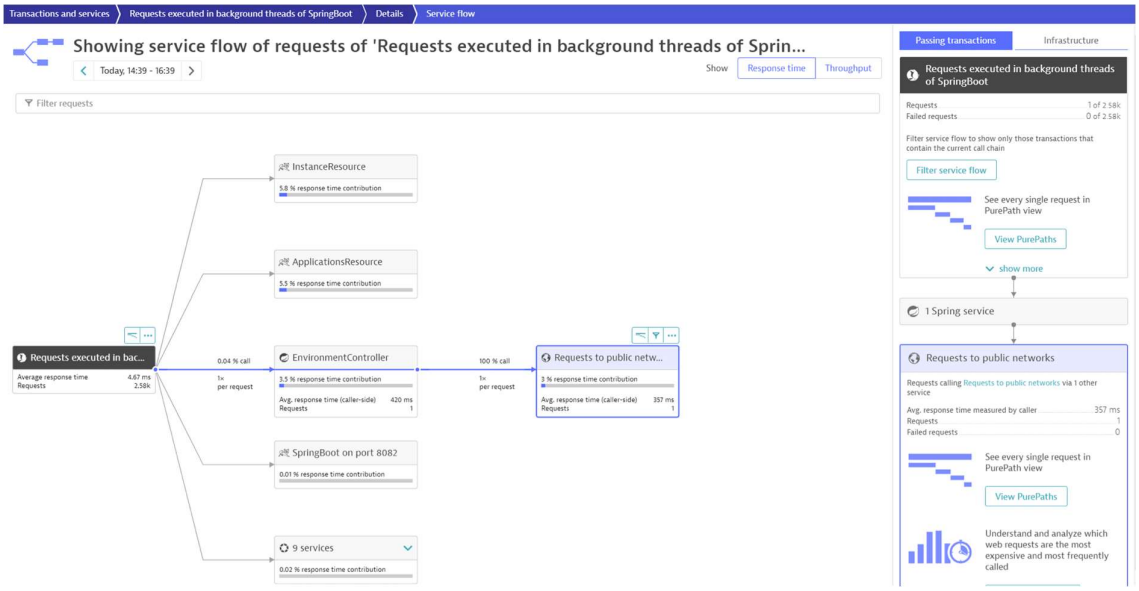
Το Dynatrace αποτελεί ένα ολοκληρωμένο λογισμικό το οποίο μεταξύ άλλων υποστηρίζει τον έλεγχο για κενά ασφαλείας, τον έλεγχο της υποδομής με αυτόματη ανίχνευση των συσκευών αλλά και των συστημάτων που λειτουργούν μέσα στον οργανισμό, την αυτοματοποίηση διαδικασιών ανάπτυξης λογισμικού, αλλά και την ανίχνευση σφαλμάτων πηγαίου κώδικα.

Πρόκειται για μία λύση ωστόσο η οποία υποστηρίζει μόνο διαδικτυακές υποδομές και συστήματα, ενώ δεν είναι ανοιχτού κώδικα για περαιτέρω ανάπτυξη αλλά και προσαρμογή σε νέες απαιτήσεις.

Στην Εικόνα 2 και Εικόνα 3 μπορούμε να δούμε τα αποτελέσματα όπως φαίνονται στη διεπαφή τελικού χρήστη για την ανίχνευση μίας ευπάθειας, αλλά και τον εντοπισμό ενεργειών που οδήγησαν σε ένα μήνυμα προειδοποίησης (alert).



Εικόνα 2: Dynatrace - Ανίχνευση ευπάθειας

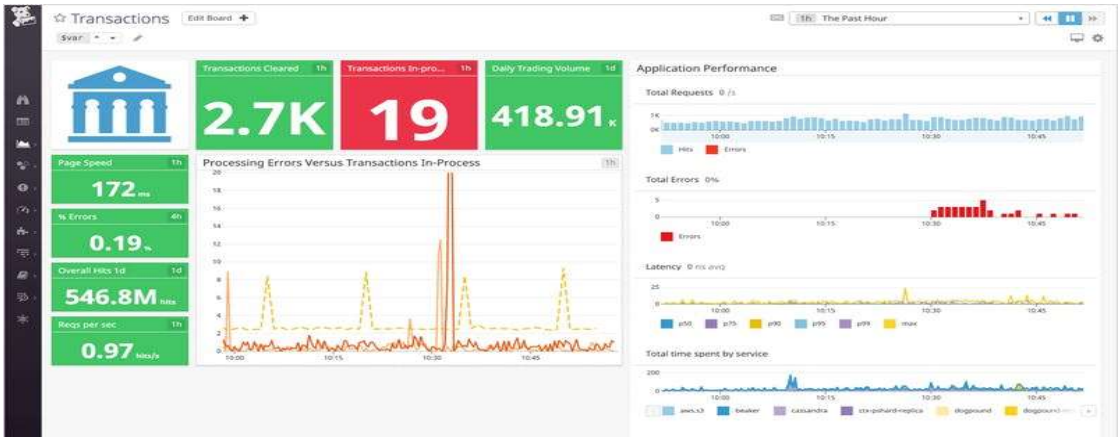


Εικόνα 3: Dynatrace - Κύκλος δράσεων που κατέληξε σε παραγωγή μηνύματος προειδοποίησης

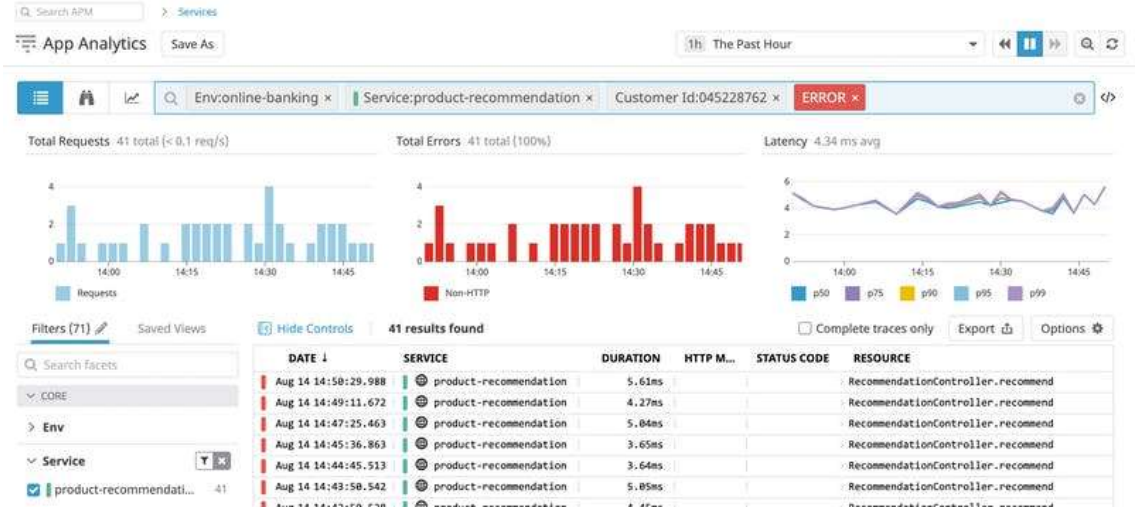
1.6 Datadog

Το Datadog είναι ένα άλλο πρόγραμμα αντίστοιχο με το Dynatrace. Υποστηρίζει και αυτό ένα μεγάλο εύρος τομέων όπως ο χρηματοπιστωτικός, ο δημόσιος τομέας, ο τεχνολογικός τομέας και άλλοι. Όπως και το Dynatrace, υποστηρίζει τον έλεγχο υποδομής, αρχείων καταγραφής έλεγχο κατάστασης δικτύων και ασφαλείας.

Στην Εικόνα 4 και Εικόνα 5, παρουσιάζονται μερικές από τις δυνατότητες οι οποίες παρέχονται στον τελικό χρήστη μέσω της αντίστοιχης διαδικτυακής εφαρμογής.



Εικόνα 4: Datadog - Αρχική Σελίδα



Εικόνα 5: Datadog - Application/source analysis

Κεφάλαιο 2: Τεχνολογίες

Στο κεφάλαιο αυτό παρατίθενται οι απαραίτητες τεχνολογίες που χρησιμοποιήθηκαν κατά την παρούσα εργασία και είναι απαραίτητες στην κατανόηση του κειμένου, αλλά και της τελικής λύσης που προτείνεται.

Σε περίπτωση που ο αναγνώστης είναι εξοικειωμένος με κάποια από τις τεχνολογικές έννοιες μπορεί με ασφάλεια να προχωρήσει στην επόμενη.

Λεπτομέρειες για το πώς χρησιμοποιήθηκε η κάθε τεχνολογία παρατίθενται στο (cross reference κεφάλαιο υλοποίηση).

2.1 JSON

Η λέξη JSON είναι ακρωνύμιο των λέξεων Javascript Object Notation. Προτάθηκε από τον Douglas Crockford το 2001. Πρόκειται για ένα πρότυπο για ανταλλαγή δεδομένων. Τα αρχεία τα οποία είναι γραμμένα σε JSON, έχουν κατάληξη .json και ο αντίστοιχος τύπος δεδομένων (media type) είναι της μορφής application/json. Τα πλεονεκτήματά του συγκεκριμένου πρότυπου είναι πολλά. Τα πιο βασικά περιλαμβάνουν

- Παράγει αρχεία εύκολα προς ανάγνωση. Ακόμα και ένας άπειρος προγραμματιστής αν ανοίξει ένα αρχείο τύπου JSON, εύκολα μπορεί να καταλάβει το είδος της πληροφορίας που μεταφέρει.
- Οι μηχανές αναλύουν εύκολα το περιεχόμενο των αντίστοιχων αρχείων JSON (γίνεται εύκολα parsing δεδομένων από αρχείο JSON).
- Οι γλώσσες προγραμματισμού μπορούν να παράγουν εύκολα αρχεία της μορφής JSON.

Σχεδόν όλες οι γλώσσες προγραμματισμού υποστηρίζουν το JSON καθώς αποτελεί ένα πρότυπο μεταφοράς δεδομένων ανεξάρτητο από γλώσσες προγραμματισμού. Είναι εναλλακτική προσέγγιση ανταλλαγής δεδομένων σε σχέση με την XML

Οι τύποι δεδομένων που υποστηρίζονται από το πρότυπο JSON είναι

- Αλφαριθμητικά (Strings).
- Αριθμοί. ακέραιοι ή δεκαδικοί.
- Λογικές τιμές (Boolean). True ή False.
- Πίνακες. Μία αριθμημένη λίστα η οποία περιέχει κλειδιά και τις αντίστοιχες τιμές οι οποίες μπορούν να είναι οποιουδήποτε τύπου.
- Αντικείμενα(Objects). Μη αριθμημένη συλλογή η οποία περιέχει ζεύγη ονομάτων-τιμών, στα οποία οι τιμές ορίζονται ως αλφαριθμητικά.
- null. Τιμή η οποία ουσιαστικά υποδεικνύει τη μη ύπαρξη αντίστοιχης τιμής.

```
{
      "type": "x-event",
      "id": "x-event--8c6af861-7b20-41ef-9b59-6344fd872a8f",
      "name": "Apache tomcat asset",
      "course_of_action_refs": [
        "course-of-action--8c6af861-7b20-41ef-9b59-6344fd872a8f"
      ]
    }
}
```

Εικόνα 6: Παράδειγμα JSON αρχείου

2.1.1 JSON Schema

Τα τελευταία χρόνια όλο και περισσότερο χρησιμοποιείται το πρότυπο JSON Schema το οποίο συμπληρώνει το πρότυπο JSON και δίνει τη δυνατότητα για ενδελεχή έλεγχο σε δεδομένα που ανταλλάσσονται ή αποθηκεύονται. Το πλεονέκτημα της προσέγγισης αυτής είναι ότι δεν αποθηκεύονται ή μεταφέρονται δεδομένα λάθος μορφής, ενώ με τη χρήση βιβλιοθηκών σε διάφορες γλώσσες προγραμματισμού, το έλεγχος των δεδομένων (validation) γίνεται ακόμα πιο γρήγορος, με λιγότερο αριθμό σφαλμάτων και με μηδενική σχεδόν ανάγκη συντήρησης/αλλαγής του πηγαίου κώδικα. Το παράδειγμα της παρακάτω εικόνας (Εικόνα 7) θα μπορούσε να αποτελεί ένα σχήμα με βάση το οποίο γίνεται ο έλεγχος (validation) του αρχείου του παραπάνω παραδείγματος (Εικόνα 6).

```
{
  "id": "../sdos/x-event.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "x-event",
  "description": "Identities an event.",
  "type": "object",
  "allOf": [
    {
      "$ref": "../../../schemas/schemas/common/core.json"
    },
    {
      "properties": {
        "type": {
          "type": "string",
          "description": "The type of this object, which MUST be the literal `x-
event`.",
          "enum": [
            "x-event"
          ]
        },
        "id": {
          "title": "id",
          "pattern": "^x-event--[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-4[0-9a-fA-F]{3}-
[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}$"
        },
        "name": {
          "type": "string",
          "description": "The name of this event."
        },
        "course_of_action_refs": {
          "type": "array",
          "description": "Specifies the Course of action that produces this
Event.",
          "items": {
            "pattern": "^course-of-action--[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-4[0-9a-
fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}$"
          }
        }
      }
    }
]
```

Εικόνα 7: Παράδειγμα JSON schema αρχείου

Στο JSON schema φαίνεται η δήλωση του type ως μία και μόνο τιμή (x-event), η δήλωση ότι το πεδίο name θα πρέπει να είναι αλφαριθμητικό, η δήλωση ότι το course_of_action_refs θα πρέπει να είναι

πίνακας με στοιχεία αλφαριθμητικά που ακολουθούν ένα συγκεκριμένο πρότυπο και τέλος η μορφή του πεδίου Id. Υπάρχουν επίσης αναφορές σε εξωτερικά αρχεία (με το keyword $ref) στα οποία μπορεί να δηλωθεί η μορφή κάποιων επιπλέον πεδίων.

Η προσέγγιση αυτή προσφέρει επεκτασιμότητα καθώς μπορούν να διαχωριστούν τα πεδία σε ξεχωριστά αρχεία, να ομαδοποιηθούν και να χρησιμοποιηθούν σε περισσότερα σημεία ενός προγράμματος.

2.2 STIX

Το πρότυπο STIX [3] βρίσκεται στην έκδοση 2.1 και αποτελεί ένα διαδεδομένο πρότυπο γραμμένο σε JSON το οποίο ορίζει οντότητες για συστήματα ασφαλείας και πιο συγκεκριμένα για ευφυία απειλών κυβερνοχώρου (cyber threat intelligence - CTI). Μπορεί να χρησιμοποιηθεί είτε μαζί με το πρωτόκολλο TAXII είτε αυτόνομα για μεταφορά και αποθήκευση δεδομένων.

Έχει αναπτυχθεί από την OASIS Cyber Threat Intelligence αλλά και μια σειρά από contributors.

Μερικά από τα πλεονεκτήματα που έχει έναντι άλλων παρόμοιων πρωτοκόλλων είναι:

- Πλούσιο documentation – Με σελίδα για issues στο Github αλλά και μια σειρά από έγγραφα τα οποία περιγράφουν τις οντότητες αλλά και τις μεταξύ τους διασυνδέσεις. Ακόμα παρέχεται μία σειρά από παραδείγματα για την καλύτερη κατανόηση του προτύπου.

- JSON – Πρότυπο το οποίο είναι εύκολα διαχειρίσιμο από προγράμματα και γλώσσες προγραμματισμού

- Μεγάλη Κοινότητα – Το project STIX έχει υποστήριξη και μάλιστα πρόσφατα έγινε ενημέρωση στην έκδοση 2.1.

- Open source – Δίνει τη δυνατότητα χρήσης αλλά και συνεισφοράς σε μία μεγάλη βάση προγραμματιστών. Κάτι το οποίο διασφαλίζει τη μελλοντική του πορεία ως project αλλά και την αποσφαλμάτωση σε περίπτωση που καταστεί αναγκαίο.

- Πληθώρα συμπληρωματικών εργαλείων – Πέρα από τα JSON Schemas που ορίζουν τις οντότητες, ο οργανισμός OASIS παραθέτει μία σειρά από συμπληρωματικά εργαλεία τα οποία μπορούν να χρησιμοποιηθούν για έλεγχο (validation), οπτικοποίηση (visualization), δημιουργία test δεδομένων (dummy data generation) αλλά και διαφανή αλληλεπίδραση με το πρότυπο μεταφοράς TAXII [5].

Οι βασικές οντότητες που υποστηρίζει το πρότυπο STIX 2.1 είναι:

- Attack Pattern – TTP το οποίο ορίζει τρόπο με τον οποίο οι επιτιθέμενοι προσπαθούν να χτυπήσουν έναν στόχο
- Campaign – Ομάδα επιθέσεων ή περιγραφή τακτικών σε ένα χρονικό διάστημα εναντίον στόχων
- Course of Action – Δράση για απάντηση σε μία απειλή
- Grouping – Ομαδοποίηση STIX αντικειμένων
- Identity – Άτομο, οργανισμός ή ομάδα. Μπορεί να αφορά ακόμα και σύστημα. Εξαρτάται από τη μοντελοποίηση κάθε ξεχωριστού συστήματος.
- Indicator – Μοντέλο το οποίο αναδεικνύει την ύπαρξη κακόβουλης ενέργειας
- Infrastructure – Σύστημα/Πρόγραμμα ή κάποιο asset που υποστηρίζει μία λειτουργία
- Intrusion Set – Ομάδα κακόβουλων ενεργειών με κοινό πρότυπο που πιστεύεται ότι έχουν πυροδοτηθεί από έναν συγκεκριμένο οργανισμό

- Location – Γεωγραφική Τοποθεσία
- Malware – Κακόβουλο λογισμικό
- Malware Analysis – Μετα-δεδομένα τα οποία αποτελούν στατική η δυναμική ανάλυση του κακόβουλου λογισμικού ή μίας οικογένειας κακόβουλου λογισμικού.
- Note – Επιπλέον πληροφορίες για κάποιο STIX αντικείμενο.
- Observed Data – Πληροφορία για κάποιο σύστημα, αρχείο ή δίκτυο. Μπορεί να χρησιμοποιηθεί και για δεδομένα stream από αισθητήρες
- Opinion - Μια ανάλυση για ένα TTP από ένα άτομο
- Report – Συλλογή πληροφοριών για μία σειρά από άλλα αντικείμενα τα οποία ομαδοποιημένα μπορεί να δίνουν μία γενική ιδέα σε κάποιον ειδικό ασφαλείας
- Threat Actor – Κακόβουλα άτομα, ομάδες ή οργανισμοί.
- Tool – Λογισμικό το οποίο ενώ από τη φύση του δεν είναι κακόβουλο, έχει χρησιμοποιηθεί από κακόβουλους χρήστες για την επίτευξη του στόχου τους.
- Vulnerability – Ευπάθεια συστήματος.

Τα αντικείμενα που παρουσιάστηκαν μπορούν να συνδυαστούν μεταξύ τους με τη χρήση των αντικειμένων

- Relationship – Ορίζει μία σχέση πηγής και προορισμού για δύο αντικείμενα STIX.
- Sighting – Ορίζει την ανίχνευση κάποιου TTP

Τα πεδία, καθώς και οι κανόνες που αφορούν ποια αντικείμενα είναι συμβατά μεταξύ τους περιγράφονται στο (https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.html).

Επίσης στο έγγραφο αυτό ορίζονται σαφώς οι κανόνες για την επέκταση του προτύπου STIX και της προσαρμογής του στις ανάγκες της εκάστοτε εφαρμογής.

2.3 REST

Πρόκειται για το πλέον διαδεδομένο πρότυπο για τη δημιουργία APIs αλλά αποτελεί και τη δομή της υλοποίησης σύγχρονων web services.

Τρέχει πάνω από το πρωτόκολλο HTTP και η μορφή της πληροφορίας μπορεί να είναι JSON, TXT, HTML, XML ή και άλλες.

Ένα Restful API συνήθως υποστηρίζει διαδικασίες CRUD (Create, READ, Update και DELETE) ενώ η τεκμηρίωση ενός τέτοιου API υποστηρίζεται από πολλές βιβλιοθήκες και γλώσσες προγραμματισμού.

Το πιο διαδεδομένο πρότυπο τεκμηρίωσης είναι το Open API (παλαιότερα Swagger).

Μέσω annotations ή YAML αρχείων μέσα στον πηγαίο κώδικα, δίνεται η δυνατότητα για δημιουργία ενός API το οποίο παρέχει στον τελικό χρήστη τη διεπαφή που είναι απαραίτητη για την εξοικείωση αλλά και τη δοκιμή του.

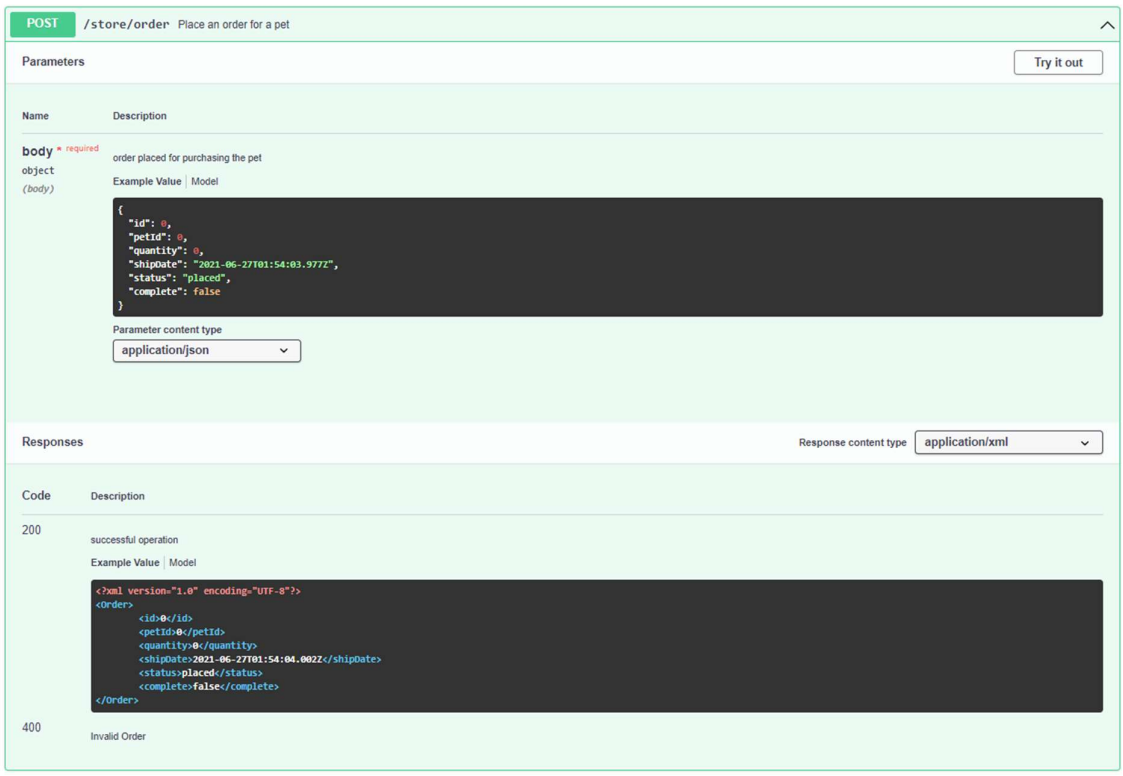
Το Open API δίνει πληροφορίες για τους πόρους (resources) που είναι διαθέσιμοι σε ένα API, για τη δομή ενός σώματος αίτησης (request) αλλά και τη δομή πιθανών απαντήσεων.

Επίσης δημιουργεί ένα model το οποίο ουσιαστικά υποδεικνύει τους κανόνες των πεδίων αίτησης και περιγράφει τους κωδικούς HTTP, σε περίπτωση σωστής ή λάθος απόκρισης.

Τα πλεονεκτήματα του Open API για τη δημιουργία της διεπαφής UI για το API είναι:

- Δημοφιλία – Το Open API είναι πλέον ώριμο και χρησιμοποιείται ολοένα και περισσότερο σε νέες εφαρμογές είτε είναι εμπορικές είτε ερευνητικές
- Ευκολία Χρήσης – Είναι ένα εργαλείο το οποίο παρέχει ένα πολύ κατανοητό περιβάλλον και διευκολύνει την πιο γρήγορη εκμάθηση συστημάτων από νέους προγραμματιστές
- Δυνατότητες – Παρέχει μεγάλο εύρος δυνατοτήτων δίνοντας ακριβείς πληροφορίες για το είδος της πληροφορίας που “περιμένει” ή “επιστρέφει” μία εφαρμογή.
- Testing – Με τη βοήθεια του Open API, μπορεί να παρακαμφθεί η χρήση επιπλέον εργαλείων όπως το Postman ή το Insomnia για τον έλεγχο των λειτουργιών του API.

Στην Εικόνα 8 βλέπουμε ένα παράδειγμα POST request. Στο συγκεκριμένο παράδειγμα, τα αναμενόμενα αποτελέσματα έγκυρης απόκρισης είναι σε μορφή XML.



Εικόνα 8: Παράδειγμα Open API UI για ένα POST HTTP request

2.4 Docker

Το Docker είναι μία τεχνολογία/πλατφόρμα η οποία επιτρέπει την ανάπτυξη εφαρμογών με χρήση τεχνολογίας virtualization σε επίπεδο Λειτουργικού Συστήματος. Για κάθε εφαρμογή δημιουργείται ένα αυτόνομο περιβάλλον. Αυτό το περιβάλλον, διαθέτει μόνο τα δομικά στοιχεία που επιλέγει ο προγραμματιστής και σε αντίθεση με παλαιότερες τεχνολογίες virtualization, δεν απαιτεί τη δημιουργία ολόκληρου εικονικού λειτουργικού συστήματος.

Τα υπο-περιβάλλοντα που δημιουργούνται για την εκτέλεση των εφαρμογών ονομάζονται containers ενώ το σύνολο των τεχνολογιών και εργαλείων που επιλέγει ο προγραμματιστής να χρησιμοποιήσει ονομάζονται image. Τα images είναι read-only πρότυπα τα οποία χρησιμοποιούνται για να χτιστούν (build) τα Docker containers.

Από τη στιγμή που το Docker επιτρέπει τη χρήση επιλεγμένων βιβλιοθηκών και εργαλείων χωρίς να είναι απαραίτητη η ύπαρξη νέου λειτουργικού συστήματος μπορούμε να διακρίνουμε τα εξής πλεονεκτήματα:

- Εξοικονόμηση πόρων
- Διαχωρισμός λειτουργιών
- Ευκολότερη ενημέρωση εφαρμογών
- Ευκολότερος τρόπος αναπαραγωγής της εφαρμογής σε πολλαπλά περιβάλλοντα (Development/Staging/Testing και άλλα)

Ενώ οι ενσωματωμένες λειτουργίες του Docker, επιτρέπουν

- Εύκολη διαχείριση κίνησης (Load Balancing)
- Αξιόπιστες εφαρμογές με ελάχιστο downtime (με τον πολλαπλασιασμό των containers σε λειτουργία)
- Επεκτασιμότητα ανάλογα με την κίνηση (traffic)
- Δυναμικό κόστος λειτουργίας εφαρμογής (πάλι με βάση την κίνηση – traffic)

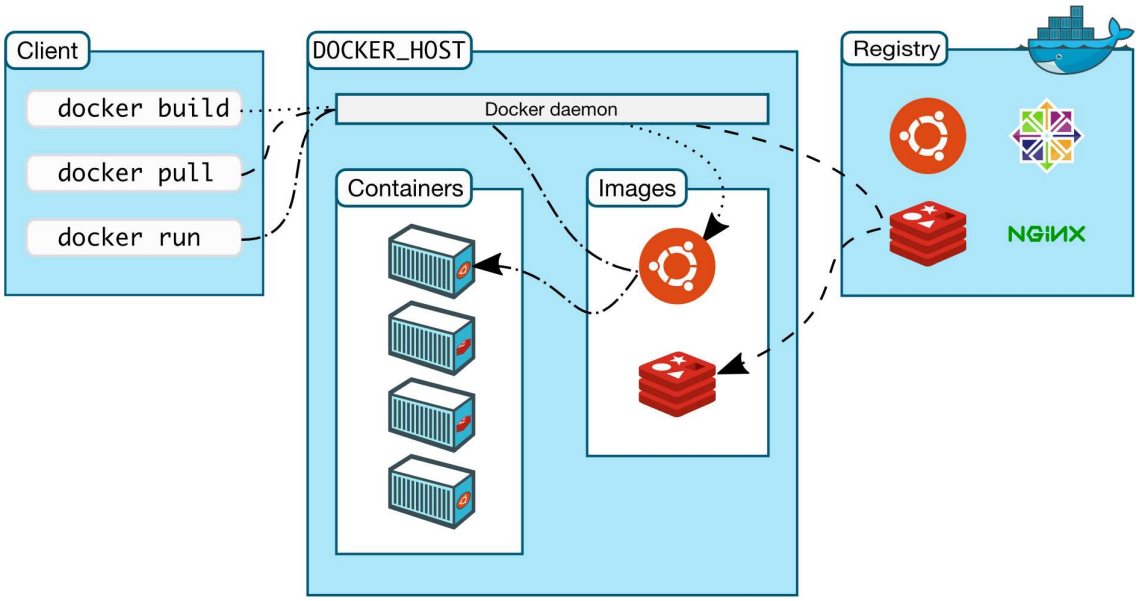
Ο επίσημος χώρος ανίχνευσης images είναι το Docker Hub. Από εκεί ο κάθε προγραμματιστής μπορεί να πάρει τα κομμάτια που επιθυμεί για να χτίσει το τελικό container.

Με ρυθμίσεις που γίνονται σε αρχείο Dockerfile, ο προγραμματιστής μπορεί να θέσει μεταβλητές περιβάλλοντος, να δημιουργήσει πολλαπλά και κλιμακωτά containers, να θέσει ονόματα σε αυτά και να τα εντάξει σε εικονικό δίκτυο ώστε να μπορούν να επικοινωνήσουν.

Ένας άλλος τρόπος διαχείρισης πολλαπλών containers είναι η χρήση του Docker compose εργαλείου με την παραμετροποίηση αρχείων ρυθμίσεων Docker-compose.yaml.

Το Docker compose ορίζει και τον τρόπο εκτέλεσης Docker εφαρμογών. Ορίζει ουσιαστικά Docker services τα οποία συνιστούν την εφαρμογή.

Στην Εικόνα 9 μπορούμε να δούμε την αρχιτεκτονική της πλατφόρμας Docker. Σε αυτή διακρίνουμε τον client αλλά και το server της αρχιτεκτονικής. Ο server είναι υπεύθυνος για τη λειτουργία των εφαρμογών με τη βοήθεια και του Dockerd (deamon). Ένας Docker client μπορεί να είναι το Docker compose ενώ η registry συνήθως είναι το Docker hub χωρίς να αποκλείονται άλλες πηγές (συνήθως private) .



Εικόνα 9: Docker - High level Αρχιτεκτονική

Για περισσότερες πληροφορίες σχετικά με το Docker αλλά και το πώς μπορείτε να τρέξετε τις δικές σας εφαρμογές, ανατρέξτε στο επίσημο documentation.

2.5 Angular

Πρόκειται για ένα από τα πιο διαδομένα πλαίσια (frameworks) γραμμένο σε γλώσσα Typescript, ένα υπερ-σύνολο της γλώσσας Javascript. Τη στιγμή που γράφεται η εργασία βρίσκεται στην LTS έκδοση 12.0.0. Έχει αναπτυχθεί από την Google με πρώτη έκδοση το 2016.

Συγκρίσιμα frameworks αποτελούν το React το οποίο έχει αναπτυχθεί από το Facebook και η πρώτη διάθεσή του ήταν το 2013. Μία ακόμα δημοφιλής επιλογή είναι και το VueJS το οποίο αναπτύχθηκε από έναν πρώην υπάλληλο της Google τον Evan You και πρώτη διάθεση το 2014.

Και τα τρία αυτά δημοφιλή frameworks, έχουν μεγάλη κοινότητα ενώ και η αγορά εργασίας φαίνεται να τα προτιμά τα τελευταία χρόνια.

Το Angular αποτελεί την πιο ολοκληρωμένη λύση καθώς είναι ένα πλήρες MVC framework.

Τα τελευταία χρόνια και λόγω της γρήγορης προσαρμογής και ανάπτυξης της γλώσσας Typescript, τα frameworks αυτά έχουν πολύ καλή υποστήριξη streaming με την υποστήριξη ασύγχρονων προγραμμάτων βασισμένα σε events.

Ειδικά για το Angular, μία πολύ δημοφιλής επιλογή βιβλιοθήκης για τις παραπάνω λειτουργίες αποτελεί η RxJS.

Δομικοί λίθοι του Angular αποτελούν

- Components – Directives που δηλώνουν και ένα template. Ορίζουν μία κλάση Typescript για τη συμπεριφορά, ένα html template, προαιρετικά styling με CSS ή SCSS
- Templates – HTML για το οπτικό κομμάτι ενός component
- Modules – Μεγαλύτερα τμήματα τα οποία μπορούν να περιέχουν services, components ή άλλα αρχεία κώδικα. Βοηθούν στην επεκτασιμότητα angular εφαρμογών
- Directives – Προσθέτουν λειτουργικότητα και μπορούν να αλλάξουν είτε τη συμπεριφορά ενός element/component/directive ή να αλλάξουν το DOM

Πλεονεκτήματα του Angular Framework αποτελούν

- Μεγάλη κοινότητα
- Ολοκληρωμένο MVC framework
- Ανάπτυξη από Google
- Εύκολη υποστήριξη επεκτάσεων CSS (SASS, LESS)
- Εύκολο testing (jasmine/karma)
- Καλό documentation
- Δυνατό εργαλείο κονσόλας (Angular CLI) για πιο γρήγορη και αποδοτική ανάπτυξη εφαρμογών
- Εύκολος διαχωρισμός κώδικα (components, services)
- Διαφανές dependency Injection
- Typescript

Για περισσότερες πληροφορίες μπορείτε να ανατρέξετε στο επίσημο site του Angular.

2.6 Ionic

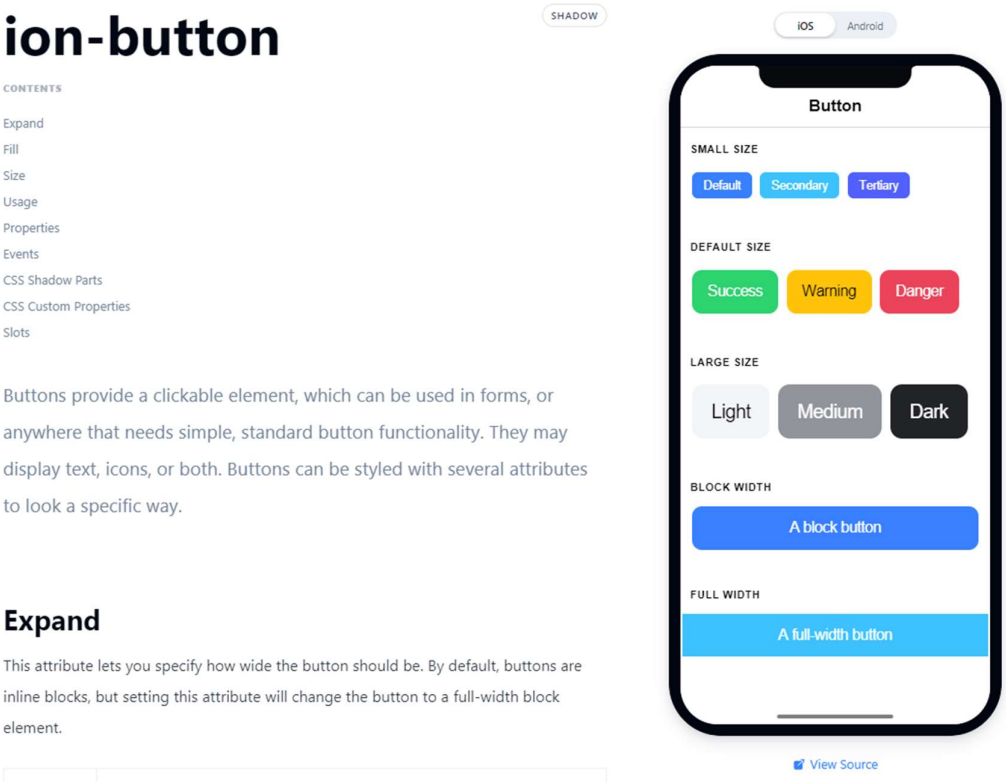
Το Ionic αποτελεί ένα σετ εργαλείων ανοιχτού κώδικα για την ανάπτυξη native mobile αλλά και web εφαρμογών. Έχει δημιουργηθεί από τον Max Lynch, το Ben Sperry και τον Adam Bradley εργαζόμενους της εταιρίας Drifty το 2013. Η πρώτη έκδοση γράφτηκε με AngularJS, ωστόσο οι νέες εκδόσεις υποστηρίζουν μία πληθώρα από επιλογές όπως το Angular 2+, το React αλλά και το VueJS.

Τη στιγμή της εκπόνησης της εργασίας το Ionic βρίσκεται στην LTS έκδοση 5.3.4.

Οι εφαρμογές γράφονται με τη βοήθεια web τεχνολογιών ενώ για την εγκατάσταση και την εκτέλεσή τους χρησιμοποιείται το Apache Cordova ή το Capacitor, εργαλεία που επιτρέπουν τη διασύνδεση με υλικό σε mobile συσκευές (κάμερα, μικρόφωνο, GPS κ.α.) και δίνει τη δυνατότητα να μη χρησιμοποιηθούν αμιγώς platform specific APIs όπως το Android ή το Windows Phone.

Το Ionic έχει τα εξής πλεονεκτήματα

- Πλατφόρμα ανοιχτού κώδικα
- Ανεξάρτητο από το framework υλοποίησης
- Αρκετά ώριμο και δοκιμασμένο από μεγάλους οργανισμούς
- Πληθώρα έτοιμων components που διευκολύνουν την ανάπτυξη μίας εφαρμογής. Δείτε για παράδειγμα τα κουμπιά και την υλοποίησή τους στην Εικόνα 10 και Εικόνα 11.



Εικόνα 10: Ionic Buttons

```
<!-- Default -->
<ion-button>Default</ion-button>

<!-- Anchor -->
<ion-button href="#">Anchor</ion-button>

<!-- Colors -->
<ion-button color="primary">Primary</ion-button>
<ion-button color="secondary">Secondary</ion-button>
<ion-button color="tertiary">Tertiary</ion-button>
<ion-button color="success">Success</ion-button>
<ion-button color="warning">Warning</ion-button>
<ion-button color="danger">Danger</ion-button>
<ion-button color="light">Light</ion-button>
<ion-button color="medium">Medium</ion-button>
<ion-button color="dark">Dark</ion-button>

<!-- Expand -->
<ion-button expand="full">Full Button</ion-button>
<ion-button expand="block">Block Button</ion-button>

<!-- Round -->
<ion-button shape="round">Round Button</ion-button>

<!-- Fill -->
<ion-button expand="full" fill="outline">Outline + Full</ion-button>
<ion-button expand="block" fill="outline">Outline + Block</ion-button>
<ion-button shape="round" fill="outline">Outline + Round</ion-button>

<!-- Icons -->
<ion-button>
 <ion-icon slot="start" name="star"></ion-icon>
 Left Icon
</ion-button>

<ion-button>
 Right Icon
 <ion-icon slot="end" name="star"></ion-icon>
</ion-button>

<ion-button>
 <ion-icon slot="icon-only" name="star"></ion-icon>
</ion-button>

<!-- Sizes -->
<ion-button size="large">Large</ion-button>
<ion-button>Default</ion-button>
<ion-button size="small">Small</ion-button>
```

Εικόνα 11: UI buttons code

Για περισσότερες πληροφορίες μπορείτε να ανατρέξετε στην επίσημη ιστοσελίδα του Ionic.

2.7 Laravel

Το Laravel είναι ένα framework ανοιχτού κώδικα και αποτελεί ένα σύνολο αρχείων και πακέτων το οποίο διευκολύνει την ανάπτυξη διαδικτυακών εφαρμογών. Είναι γραμμένο σε γλώσσα PHP και αναπτύχθηκε από τον Taylor Otwell. Η πρώτη προσπάθεια έγινε τον Ιούνιο του 2011 με την επίσημη έκδοση Laravel 1, η οποία στόχο είχε να υποστηρίξει τους προγραμματιστές γλώσσας PHP, οι οποίοι αντιμετώπιζαν εκείνη την περίοδο σημαντικά προβλήματα με το πιο δημοφιλές τότε PHP framework, το CodeIgniter.

Άλλα δημοφιλή Frameworks είναι το Symfony (πάνω στο οποίο βασίστηκε το Laravel), το Zend και το Yii.

Το Laravel αναπτύχθηκε σε γλώσσα PHP. Οποιαδήποτε εφαρμογή Laravel, γράφεται με γλώσσα PHP και την έκδοση που υποστηρίζεται από το framework. Το Laravel τη στιγμή της εκπόνησης της εργασίας βρίσκεται στην LTS έκδοση 9 ενώ η PHP στην έκδοση 8.

Πάνω από τις μισές διαδικτυακές εφαρμογές είναι γραμμένες σε PHP. Έχει απίστευτα μεγάλη κοινότητα, η οποία είναι εκεί για να λύσει κάθε σας πρόβλημα, αναπτύσσεται συνεχώς και έχει εύκολη σύνταξη. Λόγω της δημοφιλίας της, σαν γλώσσα παρέχει και πολλές θέσεις εργασίας. Ακόμα, η PHP από την έκδοση 5 υποστηρίζει αντικειμενοστρεφή προγραμματισμό καθώς και πολλά άλλα νέα στοιχεία τα οποία την καθιστούν την κατάλληλη επιλογή.

Πέραν της γλώσσας PHP άλλα πλεονεκτήματα του Laravel Framework αποτελούν

- Δημοφιλία – Πρόκειται για το πιο δημοφιλές PHP Framework
- Κοινότητα – Documentation και αρκετά websites με ενημερωμένο υλικό (π.χ. laracasts)
- Συντακτικό – Με τη βοήθεια του Eloquent, ερωτήματα προς τη βάση γίνονται γρήγορα και σε πολύ λίγες γραμμές κώδικα
- Dependency Injection – Με τη βοήθεια του Service container, το dependency injection είναι εξαιρετικά διαφανές.
- Οργάνωση Κώδικα – Με χρήση namespaces αλλά και τη γενικότερη δομή φακέλων το framework διευκολύνει την οργάνωση κώδικα
- Object Oriented – Όλες οι βιβλιοθήκες αλλά και ο core κώδικας του project είναι γραμμένος σε αντικειμενοστρεφή PHP
- Ανοιχτού Κώδικα – Με συχνές ενημερώσεις και LTS εκδόσεις
- Ασφάλεια – Με υποστήριξη λειτουργιών για την εύκολη αντιμετώπιση ζητημάτων OWASP και άλλων
- Κόστος Συντήρησης – Λόγω της γλώσσας PHP το κόστος συντήρησης ενός Laravel application είναι χαμηλό

Το οικοσύστημα του Laravel αποτελείται μεταξύ άλλων από

- Mix – Μεταγλώττιση webpack assets
- Socialite – Αυθεντικοποίηση OAuth μέσω πλατφορμών όπως το Google, το Facebook, το Github και άλλα
- Lumen – Micro framework, ιδανικό για την ανάπτυξη απλών APIs
- Nova – Διαχειριστικό περιβάλλον
- Scout – Αναζήτηση κειμένου σε eloquent models
- Vapor – Serverless πλατφόρμα ανάπτυξης
- Horizon -Έλεγχος ουρών μηνυμάτων

2.8 GraphQL

Η GraphQL είναι μία γλώσσα ερωτημάτων η οποία κάνει την ανάπτυξη APIs μία απλούστερη διαδικασία.

Αντί να οριστεί και στην εφαρμογή πελάτη αλλά και στον εξυπηρετητή το κάθε endpoint και πώς είναι προσπελάσιμο, αρκεί να οριστεί στην εφαρμογή του εξυπηρετητή ένα και μόνο endpoint και από την εφαρμογή του πελάτη να γίνονται πλέον στοχευμένα αιτήματα για ανάκτηση πληροφορίας.

Επίσης αυτό που το καθιστά ακόμα πιο ευέλικτο είναι η ικανότητα του προγράμματος πελάτη να ζητάει ακριβώς την πληροφορία που χρειάζεται χωρίς να γίνει κάποια αλλαγή στο πρόγραμμα του εξυπηρετητή. Για παράδειγμα αν χρειαστεί να ανακτήσουμε ένα επιπλέον πεδίο αρκεί να το ζητήσουμε από την εφαρμογή του πελάτη.

Η προσέγγιση αυτή κάνει την ανάπτυξη της εφαρμογής εξυπηρετητή πολύ πιο απλή και εύκολα διαχειρίσιμη. Κάνει επίσης το testing μία ακόμα απλούστερη διαδικασία. Ενώ ο προγραμματιστής δεν είναι απαραίτητο να γνωρίζει μία σειρά από endpoints με inputs και outputs αλλά μόνο ένα.

Η υποστήριξη της GraphQL είναι αρκετά μεγάλη και υπάρχουν SDKs και βιβλιοθήκες για όλες τις διαδεδομένες γλώσσες προγραμματισμού.

Η GraphQL δεν είναι εξαρτημένη από κάποια βάση δεδομένων. Βασίζεται πάνω σε queries και mutation όπου το query κάνει αναζήτηση δεδομένων ενώ το mutation, είναι υπεύθυνο για την τροποποίηση ή αποθήκευση δεδομένων. Σε μία REST αναλογικά εφαρμογή τα mutations θα ήταν τα http ερωτήματα POST, PUT, PATCH και DELETE ενώ τα GET ερωτήματα αναλογούν σε ένα GraphQL query. Σημαντικό επίσης είναι ότι και στα queries αλλά και στα mutations μπορούν να χρησιμοποιηθούν arguments αλλά και operators.

Στην Εικόνα 12 φαίνεται ένα query το οποίο είναι υπεύθυνο για τη συλλογή δεδομένων ενώ το αποτέλεσμα φαίνεται στην Εικόνα 13.

```
query {
    articles(where: {rating: {_gte: 4}})
    {
        id
        title
        rating
    }
}
```

Εικόνα 12: Παράδειγμα GraphQL ερωτήματος

```
{
    "data": {
        "articles": [
            {
                "id": 3,
                "title": "Test Article #1",
                "rating": 4
            },
            {
                "id": 7,
                "title": "Test Article #2",
                "rating": 4
            },
            {
                "id": 17,
                "title": "Test Article #3",
                "rating": 5
            }
        ]
    }
}
```

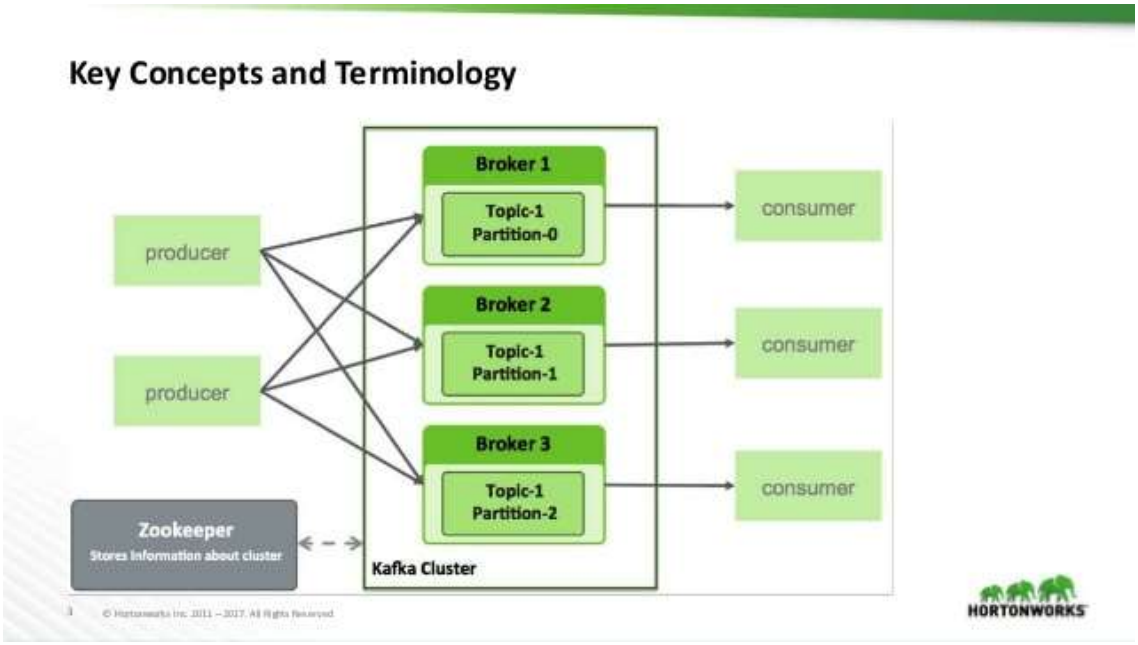

Εικόνα 13: Παράδειγμα GraphQL απόκρισης

2.9 Kafka

Το Apache Kafka αρχικά αναπτύχθηκε από το Linkedin και πλέον (2011 και έπειτα) είναι ένα σύστημα ανοιχτού κώδικα για διαχείριση ροών δεδομένων. Είναι ευρέως διαδεδομένο ενώ παρέχει εύκολη διασύνδεση με άλλα Apache εργαλεία όπως το Hadoop για κατανεμημένη αποθήκευση και επεξεργασία δεδομένων, την Cassandra DB, Kafka Connect και το Apache Spark.

Προσφέρει υψηλή απόδοση, είναι αξιόπιστο ενώ, λόγω αρχιτεκτονικής, είναι σπάνιες οι περιπτώσεις downtime. Χρησιμοποιεί την τοπολογία publish/subscribe και κάνει χρήση topics. Το μήνυμα διατηρείται έως ότου λήξει ένας προκαθορισμένος χρόνος (retention time).

Τα events στο Kafka είναι της μορφής key-value. Ωστόσο το key δεν είναι μοναδικό αναγνωριστικό. Producers (που μπορούν να είναι εφαρμογές/αισθητήρες κλπ), δημιουργούν events τα οποία Consumers μπορούν να τα ανακτήσουν από Topics μέσα στα οποία αποθηκεύονται. Μια αρχιτεκτονική υψηλού επιπέδου φαίνεται στην Εικόνα 14. Το Apache Zookeeper είναι υπεύθυνο για την επίβλεψη των cluster nodes (brokers) των topics και των partitions. Ενώ οι Brokers επιτρέπουν στους καταναλωτές να ανακτήσουν μηνύματα με βάση το topic, το partition και το offset. Πολλοί brokers μπορούν να σχηματίσουν ένα Kafka Cluster το οποίο διευκολύνει τη μεταξύ τους επικοινωνία.



Εικόνα 14: Αρχιτεκτονική Υψηλού Επιπέδου Kafka

Η αυθεντικοποίηση αλλά και ο καθορισμός ρόλων του χρήστη γίνεται χρησιμοποιώντας Kafka Access Control Lists (ACLs) και ο κάθε Kafka χρήστης μπορεί να έχει ένα ή περισσότερα από τα ακόλουθα δικαιώματα

- Read
- Write
- Create

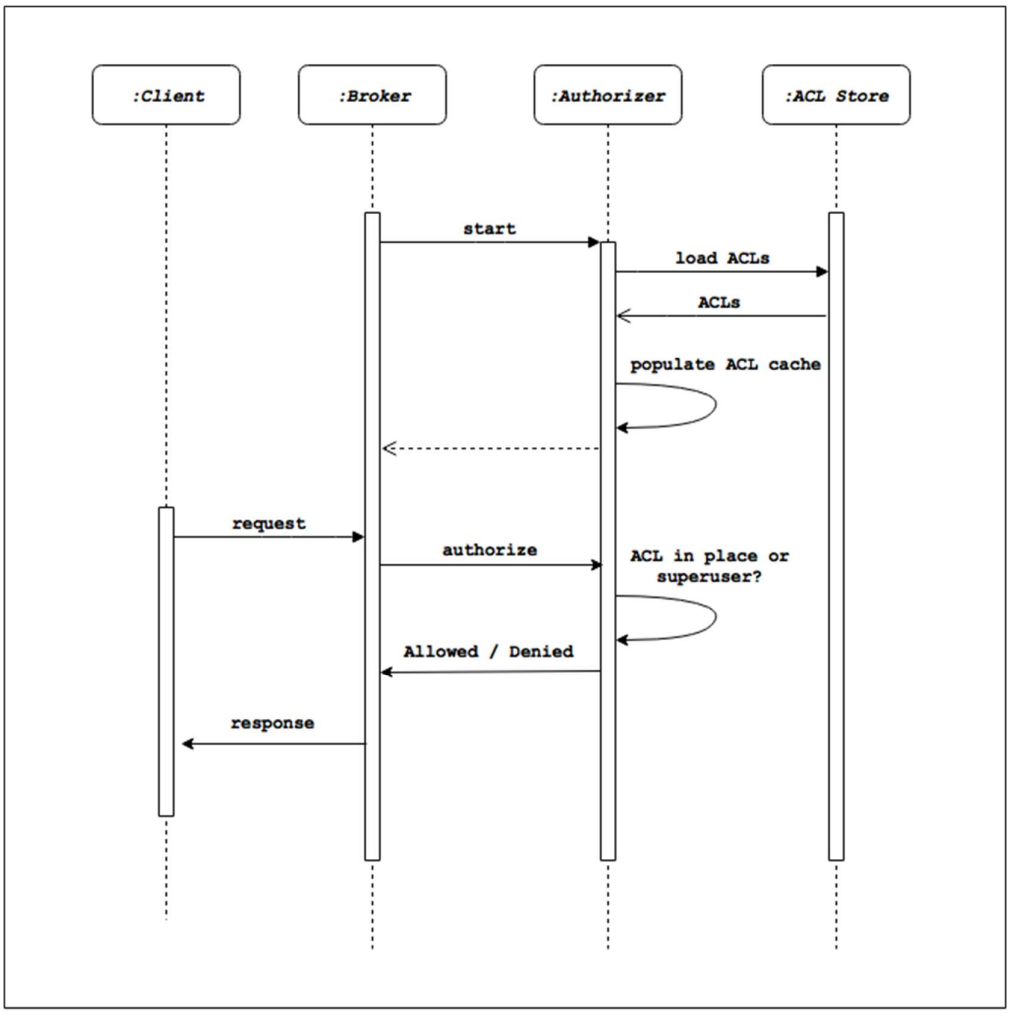
- Describe
- Alter
- Delete
- DescribeConfigs
- AlterConfigs
- ClusterAction
- IdempotentWrite
- All

Ειδικά για την περίπτωση του Topic resource που θα μας απασχολήσει, ο κάθε Kafka χρήστης μπορεί να έχει ένα ή περισσότερα από τα δικαιώματα

- Read
- Write
- Describe
- Delete
- DescribeConfigs
- AlterConfigs
- All

Ειδικά για την αυθεντικοποίηση χρησιμοποιείται το SASL (Simple Authentication and Security Layer) πρότυπο και έχουν υλοποιηθεί οι μέθοδοι PLAIN, SCRAM, OAUTH και GSSAPI. Ένα flow αυθεντικοποίησης φαίνεται στην Εικόνα 15.

Προσφέροντας πολύ υψηλότερη απόδοση (~1 εκατ. Μηνύματα το δευτερόλεπτο) από αντίστοιχα εργαλεία (πχ RabbitMQ) με ασφάλεια αλλά και επεκτασιμότητα αλλά και ένα εύρος εργαλείων που μπορούν να αξιοποιηθούν σε συνεργασία με το αυτό, αποτελεί μία πολύ καλή επιλογή για τη διαχείριση ροών.



Εικόνα 15: Authentication Flow (Kafka)

Κεφάλαιο 3: Ανάλυση Απαιτήσεων

Στο κεφάλαιο αυτό, θα αναλύσουμε τις λειτουργικές, αλλά και τις μη λειτουργικές απαιτήσεις που θα πρέπει να καλύπτει η τελική εφαρμογή μας.

Πολλές από αυτές τις απαιτήσεις έχουν καλυφθεί από τις ήδη υπάρχουσες λύσεις. Ωστόσο, προκειμένου ο αναγνώστης να έχει μία ολοκληρωμένη εικόνα των λειτουργιών, παραθέτουμε και αυτές τις απαιτήσεις.

Όσον αφορά τις μη λειτουργικές απαιτήσεις, δεν κρίνεται απαραίτητο να δώσουμε ακριβή στόχο, αλλά μία γενική περιγραφή κάλυψης αναγκών που δεν αφορούν ακριβώς τη λειτουργία του συστήματος αλλά κυρίως, το πώς θα επιτευχθούν κάποιες μετρικές (π.χ. ταχύτητα, ευκολία χρήσης, επεκτασιμότητα και άλλες)

3.1 Κύριες Απαιτήσεις Συστήματος

Ο Πίνακας 1, παρουσιάζει τις κύριες απαιτήσεις συστήματος. Χωρίζονται σε απαιτήσεις σε επίπεδο εφαρμογής (APP-REQ), αποθήκευσης Δεδομένων (DB-REQ), προγραμματισμού εργασιών (SCH-REQ), επικοινωνίας (COM-REQ) και σε επίπεδο πλατφόρμας (PLAT-REQ).

Οι απαιτήσεις εφαρμογής (APP-REQ) αναφέρονται στο τελικό περιβάλλον χρήστη, καθώς επίσης και σε άλλα components απαραίτητα για τη λειτουργία της τελικής λύσης. Τα components αυτά μπορεί να προέρχονται είτε από το FINSEC ή από κάποιο άλλο αντίστοιχο εργαλείο.

Οι απαιτήσεις πλατφόρμας, αναφέρονται στην εγκατάσταση του συστήματος καθώς επίσης και σε αρχιτεκτονικές απαιτήσεις για την τροποποίηση και πρόταση νέων ιδεών.

Οι απαιτήσεις επικοινωνίας, αναφέρονται στον τρόπο επικοινωνίας μεταξύ των οντοτήτων που συμμετέχουν στη διαμορφωμένη λύση ενώ, οι απαιτήσεις προγραμματισμού εργασιών και αποθήκευσης δεδομένων αναφέρονται στις εργασίες που μπορεί να τρέχουν στο παρασκήνιο σε προκαθορισμένα διαστήματα και τις επιλογές για την αποθήκευση και διαχείριση των δεδομένων αντίστοιχα.

Πίνακας 1: Απαιτήσεις Συστήματος

Απαιτήσεις Συστήματος				
Απαίτηση	Τίτλος	Περιγραφή	Πώς	Type
APP-REQ-000	Ανάπτυξη εφαρμογής για mobile συσκευές	Εφαρμογή η οποία θα είναι διαθέσιμη και σε mobile εφαρμογές. Αυτή η προσέγγιση θα ήταν χρήσιμη σε μικρότερους οργανισμούς που ενδιαφέρονται για την ασφάλεια των υποδομών τους	Ionic	functional

APP-REQ-001	Υποστήριξη data model πολλαπλών domains	Θα πρέπει η τελική λύση να δίνει τη δυνατότητα πολλαπλών τομέων (domains)	Προσαρμογή του FINSEC dashboard στις νέες απαιτήσεις	functional
APP-REQ-002	Υποστήριξη φυσικών και διαδικτυακών απειλών	Ολοκληρωμένη λύση για φυσικές, διαδικτυακές αλλά και μικτές	Λόγω του μοντέλου δεδομένων υποστηρίζεται ήδη	Functional
APP-REQ-003	Έλεγχος δεδομένων πριν την αποθήκευση ή την αποστολή τους σε παραλήπτες	Validation στο FINSEC Dashboard για CRUD operations	Υποστηρίζεται ήδη μέσω JSON schema forms	functional
APP-REQ-004	Real time δεδομένα μέσω topics	Υποστήριξη άμεσων προειδοποιήσεων προς τον τελικό χρήστη	Τροποποίηση του Dashboard ώστε να υποστηρίξει Kafka (subscribe)	functional
APP-REQ-005	Υποστήριξη αυθεντικοποίησης και συστήματος διαχείρισης ρόλων	Ένα σύστημα διαχωρισμού ρόλων χρήστη αλλά και ελέγχου πρόσβασης σε δεδομένα της πλατφόρμας	Υποστηρίζεται ήδη με χρήση JWT tokens από το FINSEC Dashboard	functional
APP-REQ-006	Δεδομένα πλατφόρμας διαθέσιμα στον τελικό χρήστη	Όλα τα αποτελέσματα από τα κατώτερα στρώματα θα πρέπει να είναι διαθέσιμα στον τελικό χρήστη	Υποστηρίζεται ήδη από το FINSEC Dashboard API endpoints	functional
APP-REQ-007	Οπτικοποίηση Δεδομένων	Πίνακες, Πίτες, διαγράμματα και γράφοι αλλά και απλοί πίνακες βοηθούν τους τελικούς χρήστες να κατανοήσουν καλύτερα τα δεδομένα τους	Υποστηρίζεται ήδη από το FINSEC Dashboard Apache echarts	functional

APP-REQ-008	Χρηστικότητα	Απλή διεπαφή χρήστη, εύκολα κατανοητή από τον τελικό χρήστη	Υποστηρίζεται ήδη από το FINSEC Dashboard	non-functional
APP-REQ-009	Υποστήριξη ανίχνευσης ανωμαλιών/επιθέσεων	Με βάση κανόνες και events από τα συστήματα παρακολούθησης θα πρέπει να γίνεται εντοπισμός ανωμαλιών	FINSEC anomaly detection component / Οποιοδήποτε νέο θα πρέπει να λαμβάνει δεδομένα του νέου data model από τα συστήματα παρακολούθησης και να παράγει με βάση κανόνες έξοδο για το Dashboard	functional
APP-REQ-010	Υποστήριξη πρόβλεψης επιθέσεων	Με βάση ιστορικά δεδομένα αλλά και τη χρήση αλγορίθμων γίνεται εκπαίδευση του συστήματος ώστε να παράγει προβλέψεις για επιθέσεις	FINSEC predictive analytics component / Οποιοδήποτε νέο θα πρέπει να λαμβάνει δεδομένα του νέου data model από τα συστήματα παρακολούθησης και να παράγει με βάση κανόνες έξοδο για το Dashboard	functional
APP-REQ-011	Υποστήριξη υπολογισμού ρίσκου	Με βάση συγκεκριμένες μετρικές, ευπάθειες συστήματος, ιστορικά δεδομένα, αλλά και σχεδιασμό ειδικών ασφαλείας, θα πρέπει να εκτελείται υπολογισμός ρίσκου είτε για μία υποδομή ή για μία αλυσίδα	FINSEC risk assessment engine και collaborative risk assessment component / Οποιοδήποτε νέο θα πρέπει να λαμβάνει δεδομένα του νέου data model από τα συστήματα παρακολούθησης και να παράγει με βάση κανόνες έξοδο για το	functional

			Dashboard	
APP-REQ-012	Υποστήριξη ανίχνευσης παραβάσεων κανονισμών	Βιβλιοθήκες και έλεγχος της υποδομής του οργανισμού για παραβάσεις σε κανόνες και διατάξεις	FINSEC Audit and Certification component / Οποιοδήποτε νέο θα πρέπει να λαμβάνει δεδομένα του νέου data model από τα συστήματα παρακολούθησης και να παράγει με βάση κανόνες έξοδο για το Dashboard	functional
APP-REQ-013	Υποστήριξη επικοινωνίας μεταξύ οργανισμών	Φόρμα αποστολής αλλά και μηχανισμός λήψης δεδομένων προς και από άλλους οργανισμούς.	Τροποποίηση του FINSEC Dashboard ώστε να υποστηρίζει publish και subscribe σε Kafka topics. Ήδη διαθέσιμες φόρμες και γραφική αναπαράσταση	functional
PLAT-REQ-000	Ανταλλαγή μηνυμάτων μεταξύ οργανισμών	Έγκαιρη προειδοποίηση για απειλές	Τροποποίηση ή εισαγωγή νέων συστημάτων που θα είναι συμβατά με Apache Kafka. Τροποποίηση του Collaborative DLT	functional
PLAT-REQ-001	Επεκτασιμότητα	Εύκολη εισαγωγή νέων component	Με χρήση Docker και δήλωση των νέων APIs	non-functional

PLAT-REQ-002	Ανάθεση πόρων	Αυτόματη ανάθεση επιπλέον ή λιγότερων πόρων	Docker / Kubernetes	non-functional
PLAT-REQ-003	Logging	Ενιαίος μηχανισμός logging	Kafka	functional
PLAT-REQ-004	Καλή απόδοση	Ακόμα και σε συνθήκες συμφόρησης.	Load balancing	non-functional
PLAT-REQ-005	Καλοί χρόνοι απόκρισης εφαρμογής	Με χρήση caching όπου χρειάζεται / backend pagination	Redis/Memcached/AJAX	functional
PLAT-REQ-006	Γρήγορη ανάκτηση δεδομένων	Realtime μεταφορά δεδομένων και αποτελέσματα προς άλλα components ή το Dashboard	Τροποποίηση των services ώστε να υποστηρίζουν Kafka publish/subscribe	non-functional
PLAT-REQ-007	Σαφώς ορισμένος τρόπος ανάκτησης για όλα τα components	Υποστήριξη καναλιών επικοινωνίας και ανάκτησης πληροφορίας	Topics μέσω Kafka	functional

PLAT-REQ-008	Διαχωρισμός της εφαρμογής	Ώστε ο πυρήνας core να διαθέτει τα απολύτως απαραίτητα.	Πρόκειται για αρχιτεκτονικό διαχωρισμό αλλά και επιλογή διάθεσης της λύσης. Τα extra components θα πρέπει να παρέχονται ξεχωριστά on demand και ανάλογα με τις απαιτήσεις του εκάστοτε χρήστη ή domain	non-functional
PLAT-REQ-009	Εύκολη Εγκατάσταση	Τα components της εφαρμογής θα πρέπει να μπορούν να εγκατασταθούν γρήγορα σε πολλαπλά περιβάλλοντα	Docker	non-functional
PLAT-REQ-010	Confidentiality	Το σύστημα θα πρέπει να ελέγχει την πρόσβαση σε πόρους συστήματος ανάλογα με τα δικαιώματα του κάθε χρήστη	JWT υποστήριξη από όλα τα components και το Dashboard	non-functional
PLAT-REQ-011	Νέο μοντέλο δεδομένων	Υποστήριξη του μοντέλου δεδομένων από τα επιμέρους services	Τροποποίηση ή νέα components με συμμόρφωση στο νέο μοντέλο δεδομένων	non-functional

PLAT-REQ-012	Υπολογισμοί στο παρασκήνιο	Τυχόν υπολογισμοί δε θα πρέπει να επεμβαίνουν στην εμπειρία του τελικού χρήστη	Cron jobs	non-functional
SCH-REQ-000	Αξιοποίηση Δεδομένων από κοινωνικά Δίκτυα	Twitter	Νέο component για crawling σε κοινωνικά δίκτυα	functional
SCH-REQ-001	Χρονικοί Περιορισμοί	Περιορισμός δεδομένων ανά χρόνο	Υποστήριξη από όλα τα components	functional
SCH-REQ-002	Παραλληλισμός	Παράλληλη εκτέλεση εργασιών	Queues	non-functional
COM-REQ-000	Συγχρονισμός	Συγχρονισμός components για επικοινωνία αλλά και εσωτερικά του Apache Kafka	Kafka / Zookeeper	functional
COM-REQ-001	Διαθεσιμότητα	Τα components θα πρέπει να είναι προσβάσιμα	Restart πολιτικές, κατάλληλα resources	non-functional
COM-REQ-003	Real time	Υποστήριξη real-time επικοινωνίας μεταξύ components	Kafka	non-functional

COM-REQ-004	Open API	Τα components θα πρέπει να παρέχουν endpoints σε μορφή OpenAPI	OpenAPI / GraphQL	functional
DB-REQ-000	Αποθήκευση	Τα δεδομένα θα πρέπει να αποθηκεύονται	MongoDB / MySQL	functional
DB-REQ-001	Έλεγχος παλαιών δεδομένων ή διπλο-εγγραφών	Διαγραφή παλιών δεδομένων, έλεγχοι κατά την εισαγωγή για διπλότυπα	Cron Jobs / validators / MongoDB middleware	functional

3.2 Απαιτήσεις Χρηστών

Ο Πίνακας 2 περιγράφει τις απαιτήσεις που θα πρέπει να ικανοποιηθούν για τους χρήστες της τελικής εφαρμογής. Οι απαιτήσεις αυτές έχουν ήδη υλοποιηθεί από το FINSEC Dashboard, ωστόσο τις περιλαμβάνουμε προκειμένου ο αναγνώστης να έχει ένα πιο ολοκληρωμένο έγγραφο και μια καλύτερη κατανόηση των λειτουργιών που υποστηρίζονται.

Για το σύστημά μας έχουμε δύο τύπους χρήστη ονομαστικά:

- Admin
- Security Officer

Ο πίνακας χωρίζει τις απαιτήσεις ανάλογα με το ρόλο του κάθε χρήστη.

Πίνακας 2: Απαιτήσεις Χρηστών

Full User Requirements			
Requirement	Title	Description	User
ADMIN-REQ-001	CRUD λειτουργίες για τον security officer	Ο Admin μπορεί να διαχειριστεί άλλους χρήστες με λειτουργίες CRUD	admin
ADMIN-REQ-002	Πρόσβαση σε σύστημα καταγραφής στατιστικών ανά χρήστη	Ο Admin μπορεί να επιβλέψει τη δραστηριότητα άλλων χρηστών	admin
ADMIN-REQ-003	Επικοινωνία με άλλους οργανισμούς	Ο Admin μπορεί μέσω μίας φόρμας να επικοινωνήσει με admins άλλων οργανισμών	admin
ADMIN-REQ-004	Εκ νέου ανάθεση πιθανού περιστατικού	Ο Admin μπορεί να αναθέσει κάποιο περιστατικό σε έναν officer	admin

ADMIN-REQ-005	Οπτικοποίηση περιστατικού	Ο admin μπορεί να δει όλες τις πληροφορίες για το εκάστοτε περιστατικό και το πως προέκυψε	admin
USER-REQ-000	Εικόνα οργανισμού	Οι χρήστες μπορούν να βλέπουν τη σφαιρική εικόνα του οργανισμού με πληροφορίες όπως πρόσφατες επιθέσεις, ευπάθειες, πληροφορίες για τις εγκαταστάσεις οργανισμού και άλλα	admin, officer
USER-REQ-001	Επικοινωνία με άλλους ειδικούς ασφαλείας	Μπορεί να υπάρξει επικοινωνία μεταξύ των χρηστών του ίδιου οργανισμού	admin, officer
USER-REQ-002	CRUD λειτουργίες για Assets	Σε όλους τους χρήστες επιτρέπεται η διαχείριση assets μέσω CRUD λειτουργιών	admin, officer
USER-REQ-003	CRUD λειτουργίες για Services	Σε όλους τους χρήστες επιτρέπεται η διαχείριση services μέσω CRUD λειτουργιών	admin, officer
USER-REQ-004	CRUD λειτουργίες για Regulations	Σε όλους τους χρήστες επιτρέπεται η διαχείριση regulations μέσω CRUD λειτουργιών	admin, officer

USER-REQ-005	CRUD λειτουργίες για Threats	Σε όλους τους χρήστες επιτρέπεται η διαχείριση απειλών μέσω CRUD λειτουργιών	admin, officer
USER-REQ-006	CRUD λειτουργίες για Vulnerabilities	Σε όλους τους χρήστες επιτρέπεται η διαχείριση ευπαθειών μέσω CRUD λειτουργιών	admin, officer
USER-REQ-007	Οπτικοποίηση ευπαθειών, απειλών και ρίσκων για μία αλυσίδα (service)	Όλοι οι χρήστες μπορούν να δουν πληροφορίες για τα ρίσκα που υπολογίστηκαν, τις ευπάθειες αλλά και τις απειλές που έχουν οριστεί για ένα service	admin, officer
USER-REQ-008	Οπτικοποίηση ευπαθειών, απειλών και ρίσκων για ένα asset	Όλοι οι χρήστες μπορούν να δουν πληροφορίες για τα ρίσκα που υπολογίστηκαν, τις ευπάθειες αλλά και τις απειλές που έχουν οριστεί για ένα asset	admin, officer
USER-REQ-009	Δυνατότητα αποδοχής ή απόρριψης διαμοιρασμού πληροφορίας προς άλλους οργανισμούς	Οι χρήστες έχουν τη δυνατότητα να μοιραστούν πληροφορία μέσω φορμών. Ωστόσο αν κρίνουν ότι περιέχονται απόρρητα δεδομένα μπορούν να παρακάμψουν αυτό το βήμα	admin, officer

USER-REQ-010	Πυροδότηση υπολογισμού ρίσκου	Οι χρήστες μπορούν να εκκινήσουν τη διαδικασία υπολογισμού ρίσκου χειροκίνητα(manually)	admin, officer
USER-REQ-011	Οπτικοποίηση αποτελεσμάτων ρίσκου	Οι χρήστες μπορούν να δουν οπτικοποιημένα τα αποτελέσματα για τους υπολογισμούς ρίσκου	admin, officer
USER-REQ-012	Ανίχνευση νέου TTP	Οι χρήστες μπορούν να δουν οπτικοποιημένα τα αποτελέσματα ανίχνευσης TTPs από εργαλεία scanning	admin, officer
USER-REQ-013	Ανίχνευση νέου κακόβουλου χρήστη	Οι χρήστες μπορούν να δουν οπτικοποιημένα τα αποτελέσματα ανίχνευσης κακόβουλων χρηστών κυρίως από δεδομένα που έρχονται από άλλους οργανισμούς	admin, officer
USER-REQ-014	Διαμοιρασμός Πληροφορίας	Οι χρήστες μπορούν να εγκρίνουν το διαμοιρασμό πληροφορίας. Η πληροφορία συνήθως αφορά TTPs, ρίσκα, επιθέσεις, ευπάθειες και δράσεις που εκτελέστηκαν για την αντιμετώπιση κινδύνων	admin, officer

Κεφάλαιο 4: Υλοποίηση

Στο παρόν κεφάλαιο, θα αναλυθεί η προσέγγιση για την επικοινωνία μεταξύ των microservices της τελικής εφαρμογής, αλλά και μεταξύ οργανισμών. Στη συνέχεια, θα περιγράφει ο τρόπος με τον οποίο έγινε η επέκταση του STIX μοντέλου για την κάλυψη ενός μεγαλύτερου εύρους τομέων. Τέλος, θα περιγράφει το FINSEC Dashboard, αλλά και η διαδικτυακή εφαρμογή που αναπτύχθηκε.

Η υλοποίηση αφορά τροποποιήσεις στη διαδικτυακή εφαρμογή. Οι προτάσεις όμως που θα παρατεθούν αφορούν στο σύνολο των components που λειτουργούν στην τελική λύση. Θα ήταν αδύνατο να υλοποιηθεί ένα τόσο μεγάλο έργο από την αρχή μέχρι το τέλος σε μία μόνο εργασία. Επομένως, κάποιες από τις προτάσεις αφορούν microservices που θα μπορούσαν να τροποποιηθούν για την υποστήριξη του όλου εγχειρήματος.

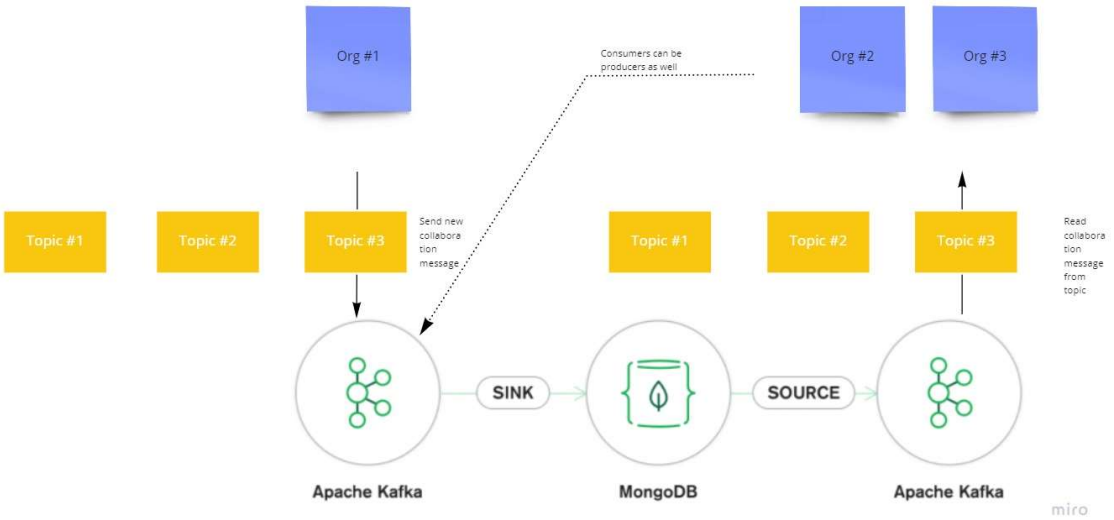
4.1 Επικοινωνία/Αρχιτεκτονική

Όσον αφορά στην επικοινωνία μεταξύ οργανισμών, το στήσιμο μίας εξωτερικής κοινής πλατφόρμας αποθήκευσης κρίνεται απαραίτητο.. Στην πλατφόρμα αυτή, η αποθήκευση δεδομένων θα γίνεται σε MongoDB, ενώ η πρόσβαση στα δεδομένα, επιτυγχάνεται με τον ίδιο μηχανισμό Keycloak (και JWT) ο οποίος παρέχεται και εντός του οργανισμού. Πρακτικά αυτό σημαίνει ότι με ένα single sign-in, ένας χρήστης, μπορεί να αποθηκεύσει δεδομένα στην εξωτερική κοινή υποδομή, αλλά προφανώς έχει και πρόσβαση στα δεδομένα του οργανισμού στον οποίο ανήκει. Η προσέγγιση αυτή έρχεται σε αντίθεση με τη χρήση blockchain μηχανισμού που έγινε στο FINSEC project. Κάποιοι από τους λόγους είναι:

- Λιγότερο απαιτητική συντήρηση του Collaborative DLT καθώς, πρόκειται απλά για τον ίδιο τρόπο ανάκτησης και αποθήκευσης πληροφοριών (με το data tier της αρχιτεκτονικής)
- Δυνατότητα επεξεργασίας και διαγραφής δεδομένων
- Υποστήριξη μεγαλύτερης ταχύτητας ειδοποιήσεων και μηνυμάτων (Kafka)
- Ίδιος μηχανισμός αυθεντικοποίησης με αυτόν που βρίσκεται εντός της πλατφόρμας

Η υποστήριξη ειδοποιήσεων και μηνυμάτων πραγματικού χρόνου γίνεται πολύ εύκολη με τη χρήση Kafka topics. Έτσι, κατά την είσοδο στη διαδικτυακή εφαρμογή, το dashboard κάνει subscribe σε συγκεκριμένο κανάλι και περιμένει μηνύματα από άλλους οργανισμούς. Η σύνδεση της εξωτερικής υποδομής με το στιγμιότυπο του Kafka μπορεί να γίνει με τη βοήθεια του Apache Kafka Connect και, πιο συγκεκριμένα. (από τη στιγμή που θα χρησιμοποιήσουμε MongoDB) μέσω του MongoDB Kafka Connector, μίας εφαρμογής διασύνδεσης, που υποστηρίζεται από την ίδια τη MongoDB.

Στην Εικόνα 16 φαίνεται η νέα αρχιτεκτονική προσέγγιση για την επικοινωνία μεταξύ οργανισμών. Στην καρδιά της νέας προσέγγισης βρίσκεται η δημιουργία Sink και Source. Το sink είναι ο χώρος όπου τα δεδομένα από ένα topic αποθηκεύονται στην εσωτερική δομή του Kafka. Ουσιαστικά γίνεται μία μετάφραση των events σε δεδομένα ενός MongoDB collection. Μετά από το βήμα αυτό, τα δεδομένα μέσω των MongoDB change streams (μηχανισμός υποστήριξης ροών για τη MongoDB) στέλνονται στο source. Από εκεί γίνεται publish σε topics από όπου μπορούν να ακούσουν άλλοι οργανισμοί.



Εικόνα 16: Επικοινωνία μεταξύ οργανισμών

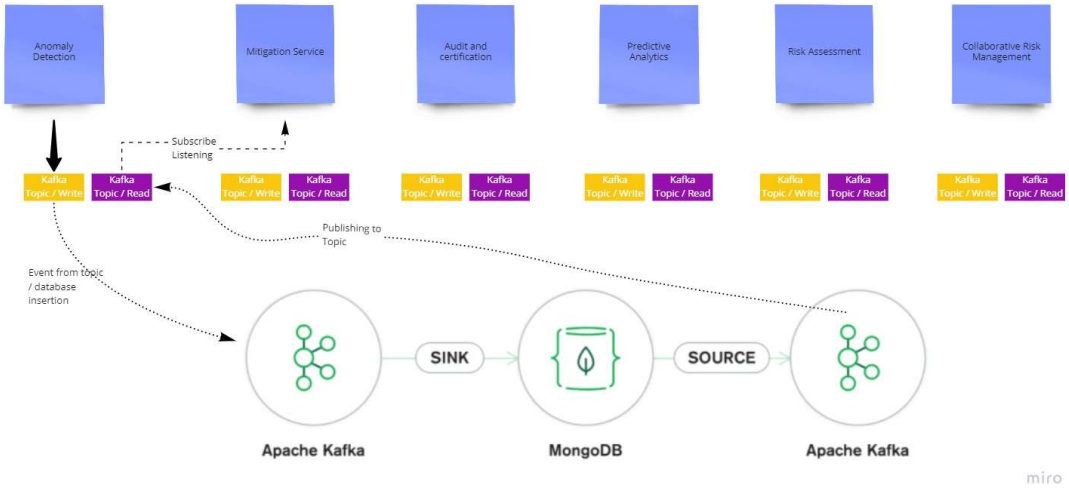
Για την επικοινωνία μεταξύ των components εντός του οργανισμού, προτείνεται η αντικατάσταση των RESTful endpoints με δυο GraphQL endpoints. Το ένα θα αφορά την ανάκτηση δεδομένων (query) και το άλλο την επεξεργασία τους (mutation).

Η υλοποίηση αυτή μπορεί να βασιστεί στο MongoDB Realm GraphQL API, το οποίο επιτρέπει την ανάκτηση δεδομένων από ένα συνδεδεμένο MongoDB cluster με τη χρήση ενός GraphQL client. Το realm για κάθε collection δημιουργεί Graphql types για κάθε collection.

Η χρήση GraphQL θα μπορεί να γίνει ένα ξεχωριστό component με το οποίο θα επικοινωνούν όλα τα υπόλοιπα μέσω Kafka topics. Το κάθε topic θα αφορά ένα component/service. Κάθε component θα κάνει subscribe στα topics των άλλων components. Έτσι, θα έχει άμεση πρόσβαση σε νέα δεδομένα (αν χρειαστεί) αλλά και αυτό με τη σειρά του θα μπορεί να δημιουργήσει δεδομένα και να τα κάνει publish στο δικό του topic.

Με την προσέγγιση αυτή, γίνεται πιο εύκολη και γρήγορη η επικοινωνία μεταξύ microservices, αλλά και το κάθε component είναι υπεύθυνο μόνο για την πραγματική λειτουργικότητα για την οποία προορίζεται. Η συντήρηση των εφαρμογών γίνεται επίσης πιο εύκολη καθώς δε χρειάζεται πλέον η δημιουργία endpoints για κάθε πιθανή δυνατότητα του εκάστοτε προγράμματος. Η μεταφορά μηνυμάτων πλέον γίνεται ταχύτατα, ενώ μπορεί να υπάρξει και ένας κοινός μηχανισμός logging για όλα τα services. Όπως και στην περίπτωση της επικοινωνίας μεταξύ των οργανισμών, θα χρειαστεί και μια βάση δεδομένων αποθήκευσης για τα δεδομένα του κάθε service. Η βάση δεδομένων θα είναι πάλι ένα MongoDB instance, διαφορετικό από αυτό του data tier και θα αφορά δεδομένα τα οποία παράγονται από το εκάστοτε service.

Στην Εικόνα 17 βλέπουμε το μηχανισμό επικοινωνίας μεταξύ των components. Σε ένα κανάλι το κάθε service, στέλνει τα νέα δεδομένα τα οποία αποθηκεύονται. Παράλληλα, γίνεται publish σε ένα δεύτερο κανάλι, το οποίο και πάλι αφορά το ίδιο το service. Όποιο από τα υπόλοιπα services χρειάζεται πληροφορία, κάνει subscribe στο κανάλι Read της εικόνας.



Εικόνα 17: Επικοινωνία μεταξύ components

4.2 Μοντέλο Δεδομένων

Για την ανάπτυξη του έργου FINSEC και την επιτυχημένη υποστήριξη του χρηματοπιστωτικού τομέα, δημιουργήθηκε το FINSTIX [2]. Ένα μοντέλο δεδομένων το οποίο επεκτείνει το STIX2.0. Παρόμοια με αυτήν την προσέγγιση, το νέο μοντέλο δεδομένων επεκτείνει την επόμενη έκδοση του STIX, την έκδοση 2.1.

Για την επέκταση του STIX θα πρέπει να ικανοποιούνται οι παρακάτω συνθήκες

- Ένα STIX αντικείμενο μπορεί να έχει οποιοδήποτε αριθμό επιπλέον πεδίων ορισμένων από τον χρήστη
- Ένα όνομα επιπλέον πεδίου πρέπει να είναι σε ASCII μορφή και πρέπει να περιέχει λατινικούς χαρακτήρες a–z (πεζά ASCII), 0–9, και κάτω παύλα (_)
- Τα ονόματα επιπλέον πεδίων θα πρέπει να ξεκινούν με "x_" ακολουθούμενα από ένα μοναδικό αναγνωριστικό (π.χ. UUID)
- Τα ονόματα επιπλέον πεδίων πρέπει να αποτελούνται από τουλάχιστον τρεις χαρακτήρες ASCII
- Τα ονόματα επιπλέον πεδίων δεν πρέπει να αποτελούνται από περισσότερους από 250 ASCII
- Σε περίπτωση που θέλουμε να είμαστε συμβατοί με νέες εκδόσεις του πρωτοκόλλου, θα πρέπει τα ονόματα επιπλέον πεδίων να ξεκινούν με "x_"
- Τα επιπλέον πεδία θα πρέπει να χρησιμοποιούνται μόνο όταν δεν υπάρχουν άλλα κατάλληλα πεδία ή οντότητες οι οποίες να υποστηρίζουν το σκοπό της δημιουργίας τους

Σχετικά με τις επιπλέον οντότητες STIX:

- Πρέπει να τηρούνται οι κανόνες των STIX πεδίων (όχι των επιπλέων του STIX που περιεγράφηκαν νωρίτερα)
- Το πεδίο type πρέπει να είναι σε μορφή ASCII και πρέπει να περιλαμβάνει μόνο τους λατινικούς χαρακτήρες a–z (πεζά ASCII), 0–9, και την παύλα (-)
- Το πεδίο type δεν πρέπει να περιλαμβάνει αλληλουχία από δύο παύλες (--)
- Τα ονόματα των επιπλέον οντοτήτων θα πρέπει να περιλαμβάνουν τουλάχιστον 3 ASCII χαρακτήρες
- Τα ονόματα των επιπλέον οντοτήτων θα πρέπει να περιορίζονται σε 250 ASCII χαρακτήρες
- Η τιμή του type πεδίου για μία επιπλέον οντότητα θα πρέπει να ξεκινάει με "x-" ακολουθούμενη από ένα μοναδικό αναγνωριστικό (π.χ. x-asset)
- Ειδικά ο προηγούμενος κανόνας μετατρέπεται σε αναγκαστικός για τη σίγουρη συμβατότητα με επόμενες εκδόσεις
- Το πεδίο Id θα πρέπει να ξεκινάει με το "x-" ακολουθούμενο από τον τύπο του αντικειμένου, μία παύλα (-) και τελικά ένα μοναδικό αναγνωριστικό σε μορφή UUID4
- Οι επιπλέον οντότητες θα πρέπει να χρησιμοποιούνται όταν δεν υπάρχει άλλη οντότητα στο πρωτόκολλο STIX που να επιτυγχάνει τον ίδιο σκοπό.

Οι επιπλέον οντότητες οι οποίες ορίστηκαν για το μοντέλο του FINSTIX φαίνονται στην Εικόνα 18.

Name	Description
Organization	Financial organization.
Asset	Organizations' valuable infrastructure. PCs, server rooms, ATMs, applications, and everything inside an organization that is crucial.
Area of Interest	Logical/physical area, for example, an indoor area (server room).
Service	A collection of assets forming a publicly exposed service, for example, a web application.
Probe	Object used to support monitoring infrastructure. A Probe usually monitors one or more areas of interest.
Probe Configuration	Data sent to a probe to configure details such as the area under monitoring or the bit rate of the monitoring process.
Event	Information on something happened/happening.
Collected data	A group of observed data collected by the network probe.
Agent	Person involved in the events created by the probes.
Risk	The calculated risk for a specific asset or service. The upper levels of FINSEC calculate it in real time.
Risk Configuration	Parameter specification to optimize the risk assessment process. It defines the triggers and other useful options.
Regulation	An object used to depict a regulation violation. FINSEC must deal with this kind of issue, even if different from attacks. The regulation violation information will be sent to regulatory authorities and other organizations.
Vulnerability score	Rating used to provide a score to a vulnerability.
Cyber-Physical Threat Intelligence	Data set fed and enriched by threat information as soon as they are gathered from the probes and processed by the Predictive Analytics module. One or more CPTI objects are used to generate the output of the intelligence process, which is a report about ongoing or possible future attacks on one or more assets belonging to the infrastructure.

Εικόνα 18: FINSTIX - επιπλέον οντότητες

Οι διαφοροποιήσεις του νέου μοντέλου που υλοποιήσαμε σε σχέση με το FINSTIX περιλαμβάνουν:

- Χρήση της νέας οντότητας Infrastructure, αντί για το Asset που ορίζεται στο FINSTIX
- Ένταξη του αντικειμένου Grouping στο data model και χρήσης του στις περιπτώσεις που γνωρίζουμε ότι η πληροφορία θα είναι πάντα απαραίτητο να είναι εμφωλευμένη
- Χρήση του Location αντί για το Area, το οποίο ορίζεται στο FINSTIX
- Τροποποίηση του Organization object, ώστε να αντικατασταθεί με την οντότητα Identity
- Χρήση namespaces. Στο core namespace οι οντότητες θα χρησιμοποιούνται από όλους τους τομείς (domains) ενώ ανάλογα με το domain θα δημιουργείται ένα νέο namespace.
- Αφαίρεση της οντότητας collected data (μπορεί να υποστηριχθεί από τα observed data)
- Χρήση της οντότητας Infrastructure αντί για Probe
- Μετατροπή του probe configuration και του risk configuration σε ένα νέο object, το οποίο θα ονομάζεται Configuration
- Χρήση της οντότητας Identity αντί για Agent που ορίζεται στο FINSTIX
- Ένταξη όλων των αντικειμένων από το STIX 2.1, αλλά και αυτών που έμειναν από το FINSTIX, σε ένα Core namespace
- Δημιουργία νέου namespace για κάθε νέο domain και με την προϋπόθεση ότι δεν καλύπτονται οι ανάγκες μας από το υπάρχον μοντέλο

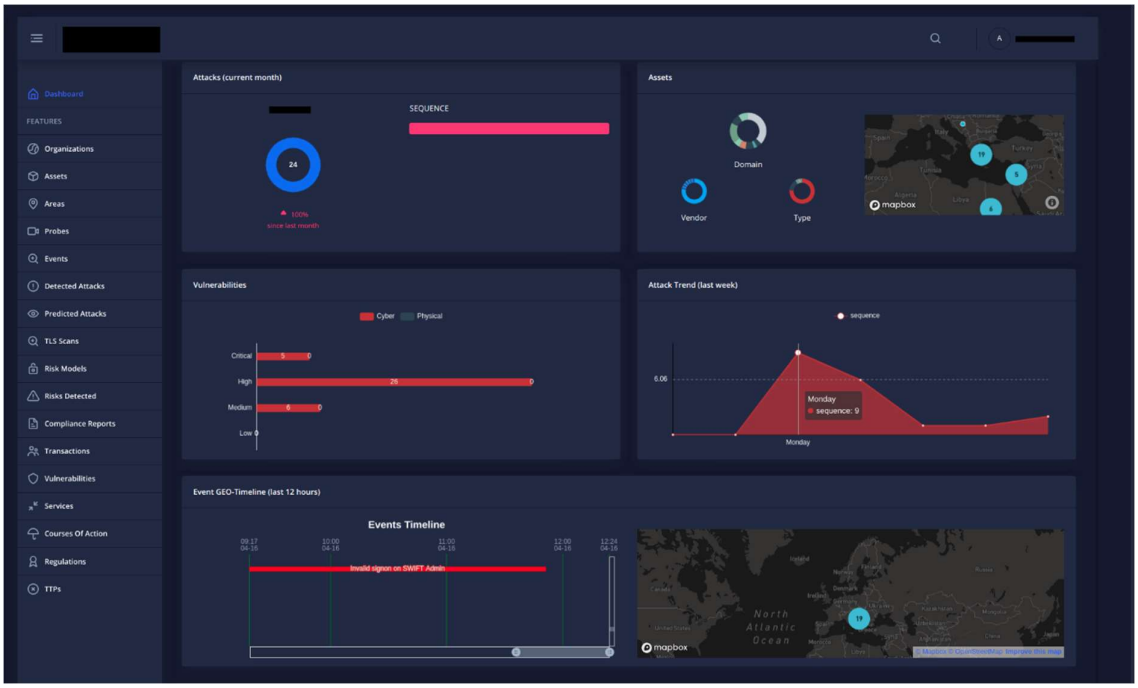
4.3 Υλοποίηση

Όπως αναφέρθηκε και σε προηγούμενη ενότητα, η παρούσα εφαρμογή βασίστηκε στο FINSEC Dashboard. Μία εφαρμογή ανοιχτού κώδικα, η οποία επεκτάθηκε για την κάλυψη των επιπλέον αναγκών που περιεγράφηκαν και στο προηγούμενο κεφάλαιο.

Δημιουργήθηκαν εργασίες παρασκηνίου στο ξεχωριστό component αναπτυγμένο σε Laravel, το οποίο αναφέρθηκε στο 4.1. Οι εργασίες αυτές αφορούσαν τη διαγραφή παλαιών δεδομένων (με ημερομηνία τροποποίησης μεγαλύτερη από έναν χρόνο) και πιο συγκεκριμένα των γεγονότων (events) τα οποία προέρχονται από τους αισθητήρες και αποτελούν το μεγαλύτερο μέρος των αποθηκευμένων δεδομένων.

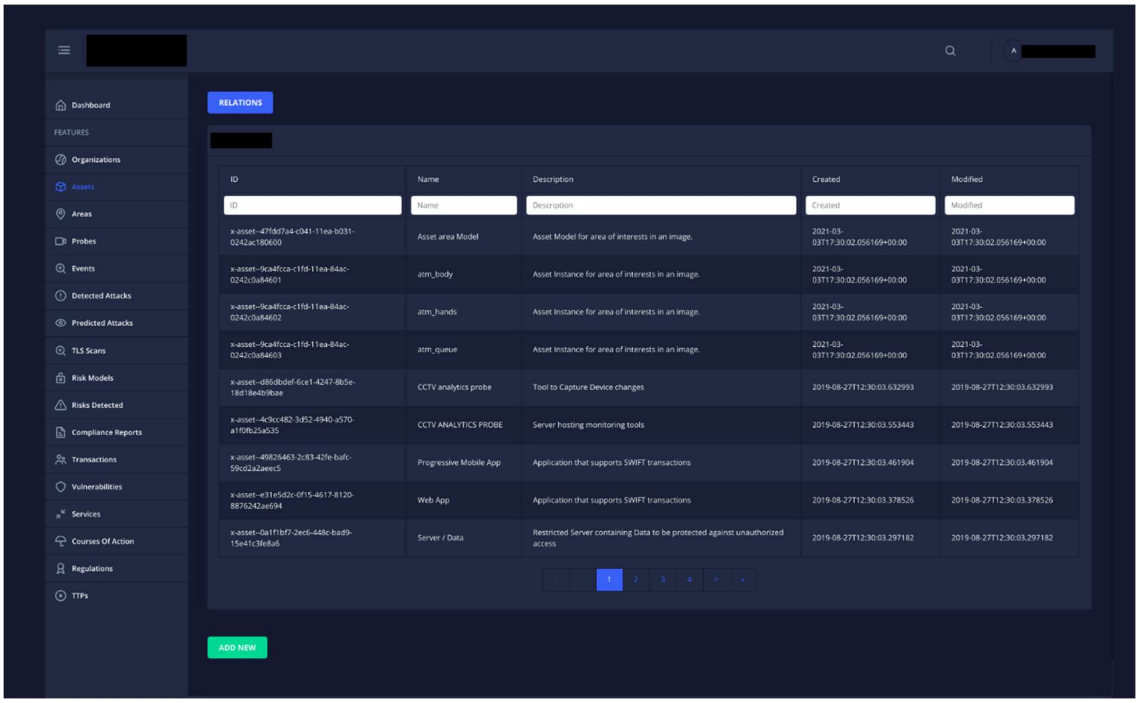
Η υλοποίηση μιας εργασίας παρασκηνίου είναι δύσκολο να παρουσιαστεί σε μία εργασία κειμένου, επομένως θα εξεταστούν οι λειτουργίες που υποστηρίζονται ήδη αλλά και οι τροποποιήσεις της διεπαφής χρήστη. Στην Εικόνα 19 μπορούμε να δούμε ένα στιγμιότυπο της αρχικής σελίδας της εφαρμογής διαδικτύου.

Κατά την είσοδο στην εφαρμογή ένας χρήστης μπορεί να δει τη γενική εικόνα του οργανισμού στον οποίο ανήκει. Αρχικά εμφανίζεται μία ανάλυση των υποδομών του οργανισμού μαζί με έναν χάρτη με τη γεωγραφική κατανομή τους. Χωρίζονται συγκεκριμένα με βάση το vendor αλλά και τον τύπο τους (πχ software, room κλπ.). Στη συνέχεια παρουσιάζεται μία ανάλυση των ευπαθειών του συστήματος (vulnerabilities) κατανεμημένες με βάση το domain τους (cyber ή physical) αλλά και τη σοβαρότητά τους (score) με βάση το πρότυπο CVSS3. Στο επόμενο διάγραμμα παρουσιάζονται οι ανιχνευμένες επιθέσεις της τελευταίας εβδομάδας, ομαδοποιημένες με βάση τον τύπο τους (σειριακές/παράλληλες κλπ.). Τέλος, εμφανίζονται τα events των τελευταίων δώδεκα ωρών. Επιλέγοντας ένα από τα events, εμφανίζονται η γεωγραφική τοποθεσία τους αλλά και λεπτομέρειες όπως η διάρκεια, ο χρόνος ανίχνευσης κ.α.

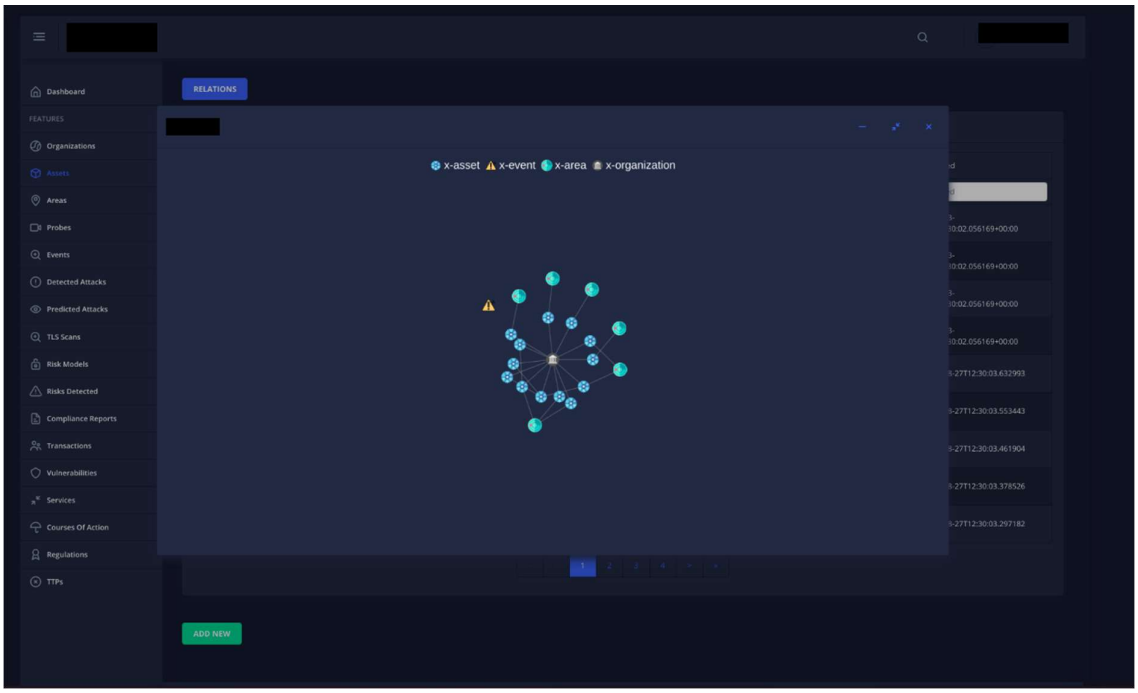


Εικόνα 19: Αρχική σελίδα εφαρμογής διαδικτύου

Στην Εικόνα 20 και Εικόνα 21, παρουσιάζεται η σελίδα λεπτομερειών. Σελίδες λεπτομερειών υπάρχουν για όλες τις οντότητες που θεωρούμε σημαντικές για έναν οργανισμό. Στην περίπτωση του χρηματοπιστωτικού τομέα, είναι οι οντότητες που ορίζονται από το FINSTIX και παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Η σελίδα λεπτομερειών περιέχει έναν πίνακα με τα στιγμιότυπα της αντίστοιχης οντότητας. Ο πίνακας αυτός παρέχει λειτουργίες αναζήτησης (search), ταξινόμησης (sort) αλλά και προβολής λεπτομερειών και σχέσεων με άλλα στιγμιότυπα του συστήματος (graph). Υποστηρίζει δε σελιδοποίηση (backend pagination), για καλύτερη απόδοση και οπτικοποίηση δεδομένων.



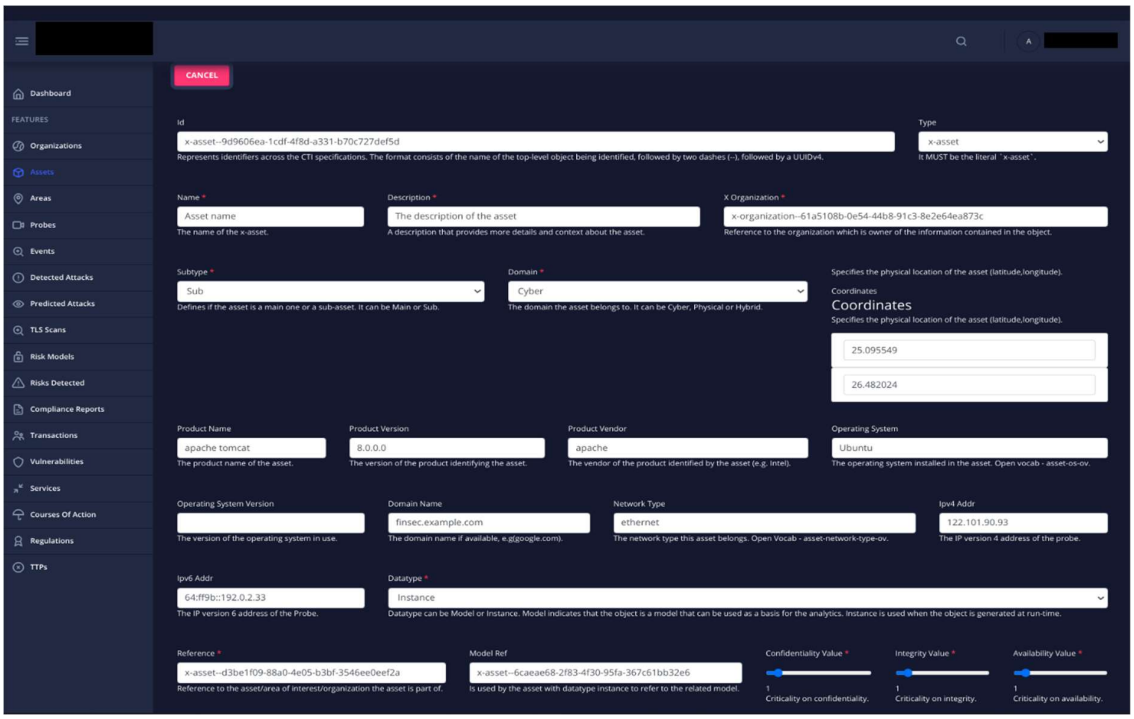
Εικόνα 20: Πίνακας στιγμιότυπων



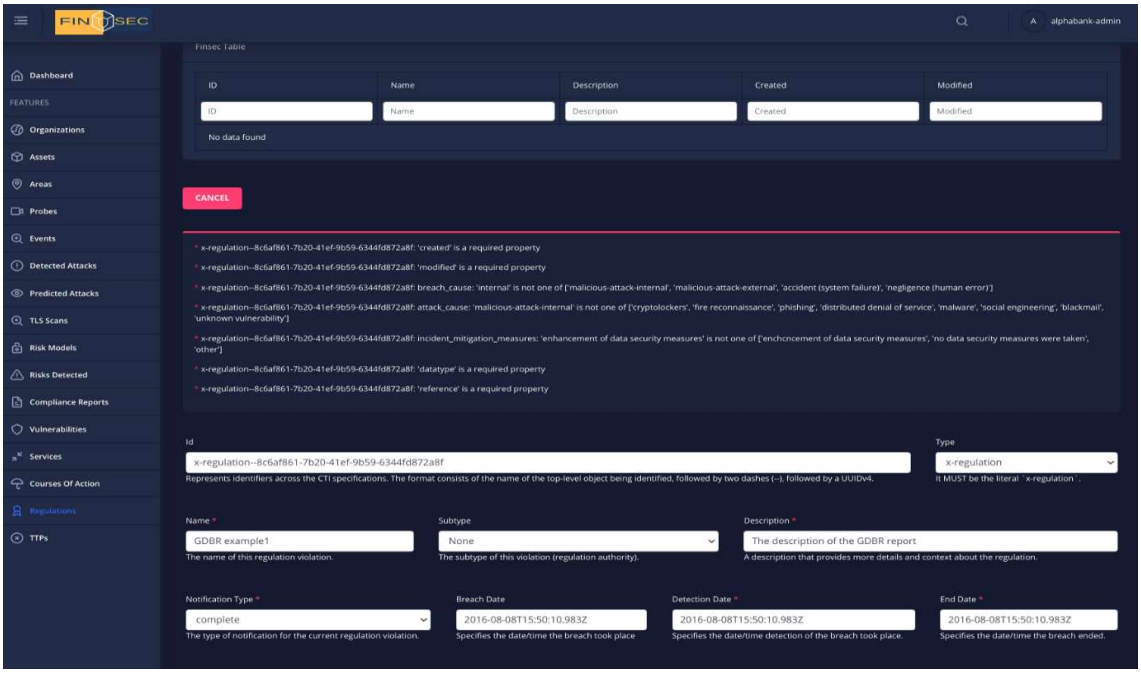
Εικόνα 21: Γράφος σχέσεων μεταξύ οντοτήτων

Στην Εικόνα 22 εμφανίζεται η φόρμα εισαγωγής νέου στιγμιότυπου. Είναι διαθέσιμη για όλες τις οντότητες του οργανισμού. Η φόρμα αυτή δημιουργείται αυτόματα από τα JSON schemas που έχουμε

περιγράψει στο προηγούμενο κεφάλαιο και παρέχεται επίσης αυτόματα ο μηχανισμός ελέγχου εγκυρότητας των στοιχείων προς εισαγωγή (validation), όπως φαίνεται και στην Εικόνα 23.

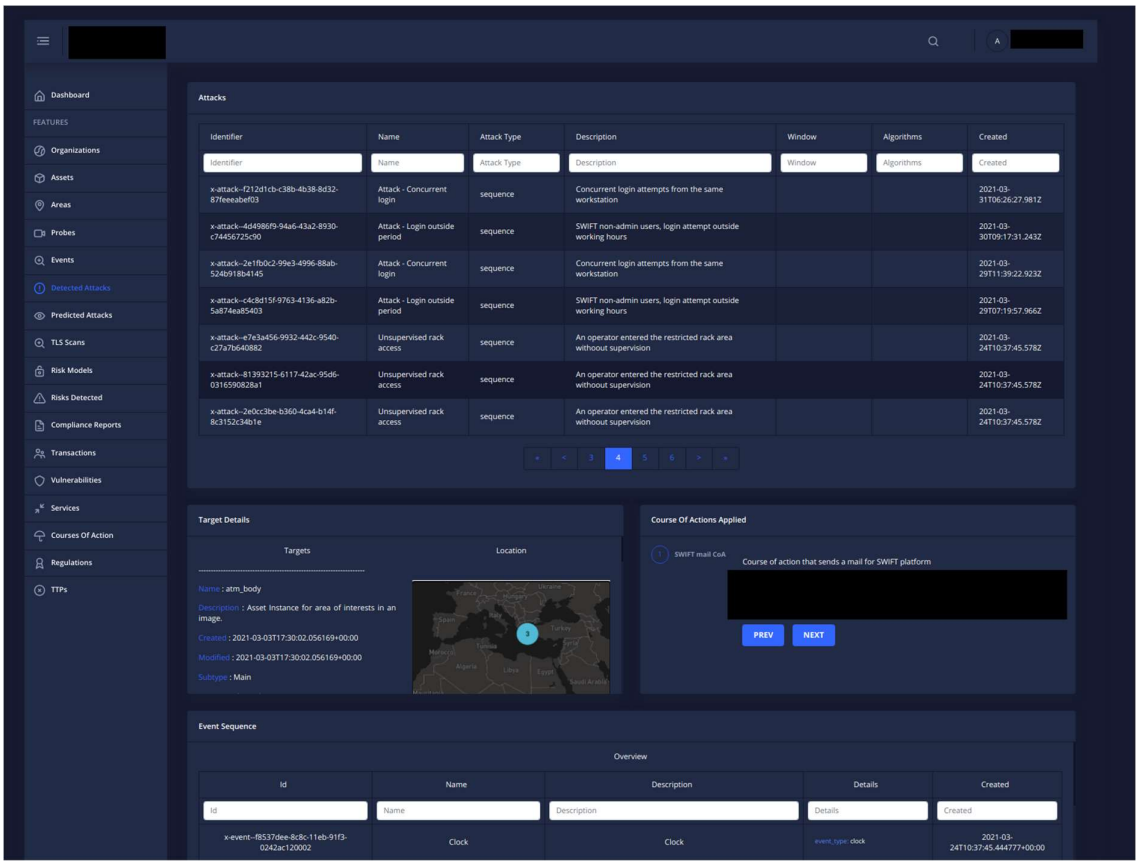


Εικόνα 22: Φόρμα εισαγωγής νέου στιγμιότυπου



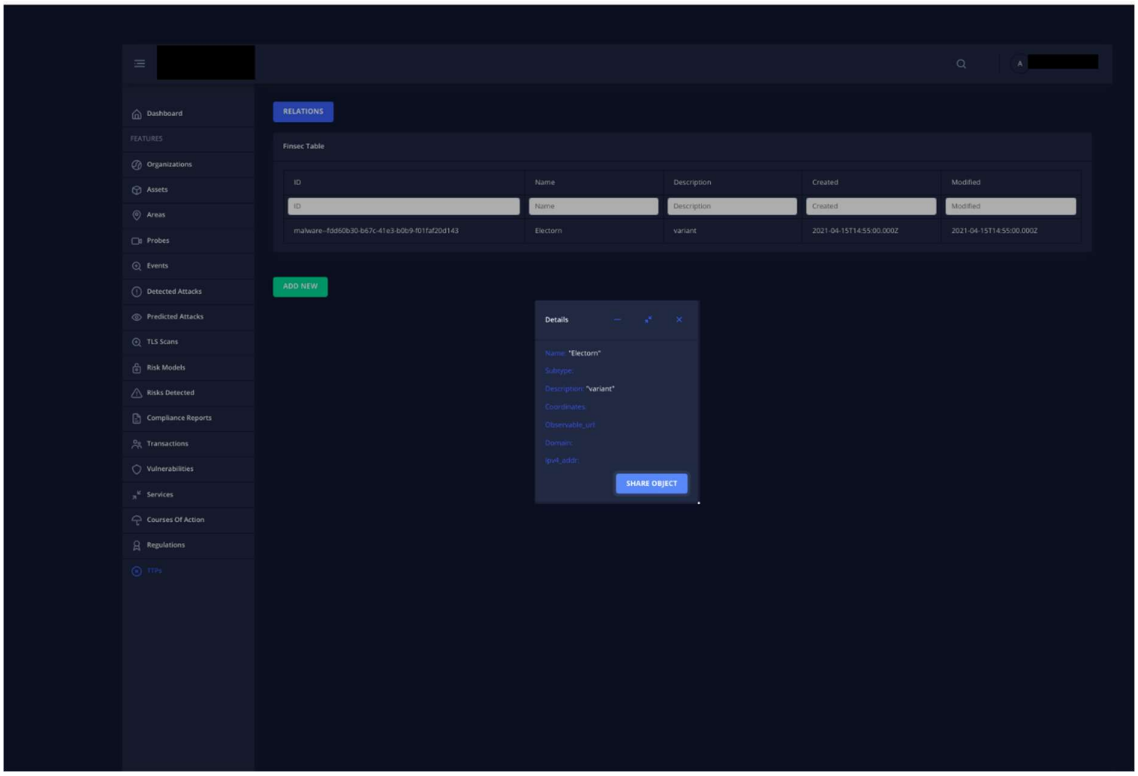
Εικόνα 23: Έλεγχος εγκυρότητας στοιχείων στιγμιότυπου

Για κάποιες από τις οντότητες δεν είναι αρκετή η παρουσίαση πληροφοριών του ίδιου του στιγμιότυπου, αλλά και των λογικών σχέσεων με άλλα στιγμιότυπα προκειμένου να είναι ξεκάθαρο στον τελικό χρήστη το πώς προέκυψε/δημιουργήθηκε το στιγμιότυπο αυτό από την εφαρμογή. Στην Εικόνα 24 παρουσιάζεται η σελίδα λεπτομερειών για τις επιθέσεις οι οποίες έχουν ανιχνευθεί. Με επιλογή μίας από αυτές τις επιθέσεις, εμφανίζονται οι υποδομές, οι οποίες επηρεάζονται, οι ενέργειες αντιμετώπισης που μπορεί να έχουν εφαρμοστεί, καθώς και η αλληλουχία των γεγονότων (events), η οποία οδήγησε στην ανίχνευση της συγκεκριμένης επίθεσης. Η ίδια λογική ακολουθείται για την οντότητα των ρίσκων τα οποία έχουν υπολογιστεί, τις επιθέσεις οι οποίες έχουν προβλεφθεί, αλλά και τις παραβάσεις κανονισμών οι οποίοι έχουν εντοπιστεί.

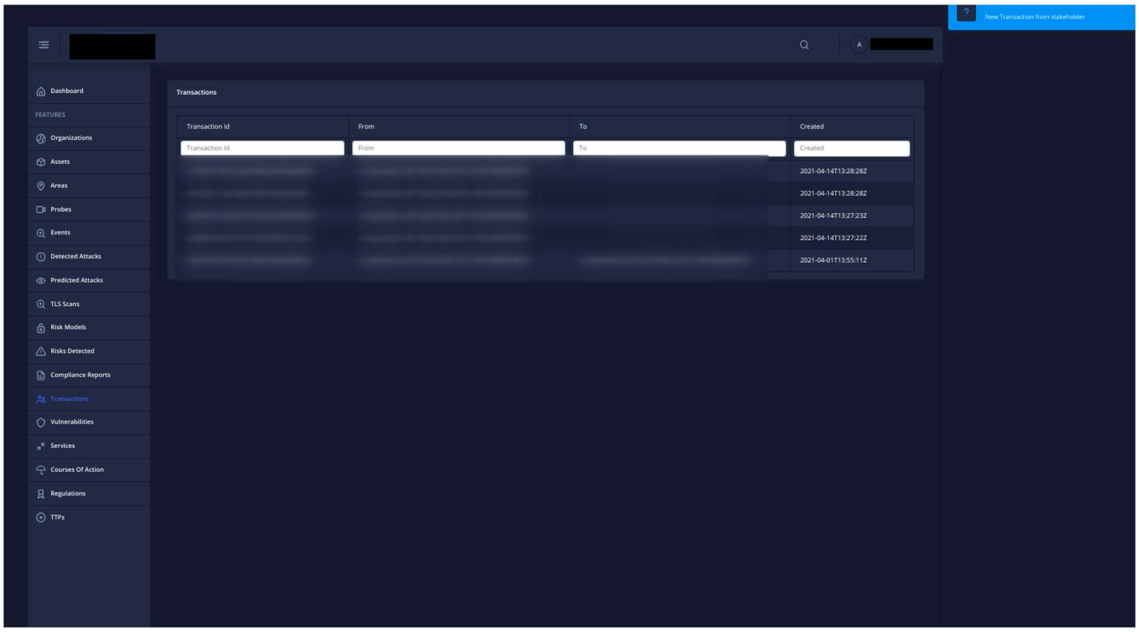


Εικόνα 24: Σελίδα λεπτομερειών / εμφάνιση σχέσεων και αλληλουχίας γεγονότων

Για συγκεκριμένες οντότητες όπως ρίσκα, επιθέσεις, TTPs, αλλά και παραβίασης κανονισμών, δίνεται η δυνατότητα να διαμοιραστεί η πληροφορία και σε άλλους οργανισμούς. Αυτό προϋποθέτει την έγκριση του χρήστη της εφαρμογής όπως φαίνεται και στην Εικόνα 25. Ενημερώσεις αντικειμένων, νέες επιθέσεις, νέοι υπολογισμοί ρίσκων αλλά και πληροφορία από άλλους οργανισμούς εμφανίζεται στον τελικό χρήστη σε πραγματικό χρόνο στο πάνω μέρος της οθόνης. Με επιλογή της ειδοποίησης μπορεί να δει τις λεπτομέρειες που επιθυμεί (Εικόνα 26).

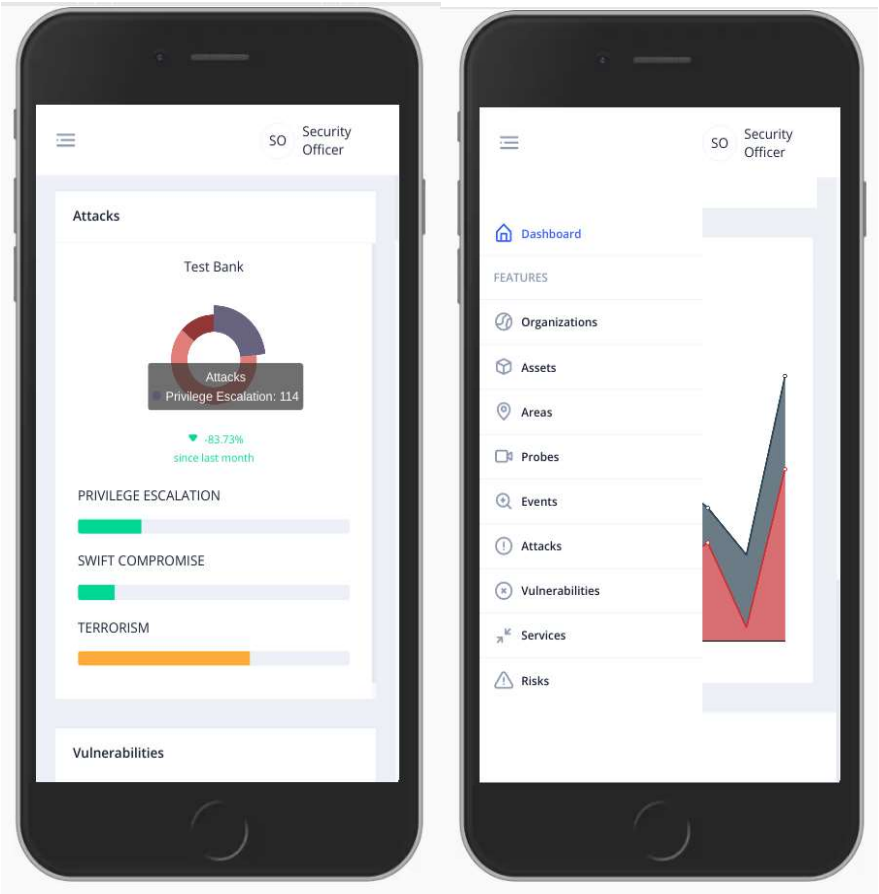


Εικόνα 25: Αίτηση έγκρισης αποστολής πληροφορίας προς άλλους οργανισμούς

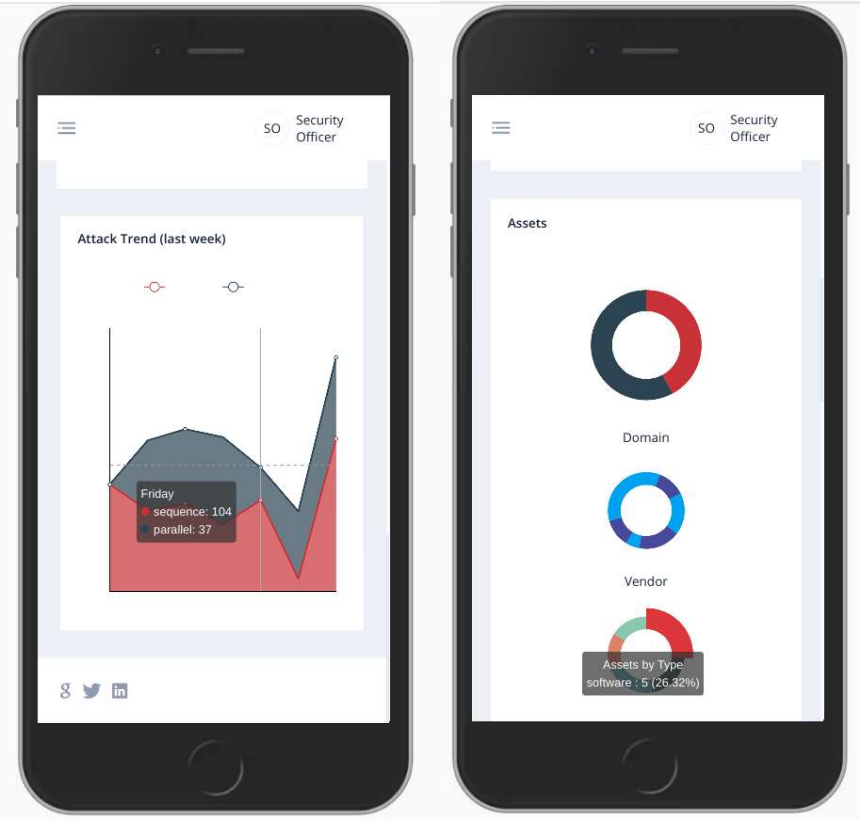


Εικόνα 26: Ειδοποίηση συναλλαγής σε πραγματικό χρόνο

Αφού ορίσαμε το μοντέλο δεδομένων από τη φύση του FINSEC Dashboard που περιεγράφηκε παραπάνω, υπάρχει υποστήριξη των νέων τομέων εφαρμογής (domain) ανάλογα και με τον οργανισμό. Ωστόσο, για την υποστήριξη μικρότερων οργανισμών, αλλά και την παροχή ευκολότερης πρόσβασης, δημιουργήσαμε μία εφαρμογή σε Ionic. Η εφαρμογή αυτή κάνει χρήση των περισσότερων κομματιών της διαδικτυακής εφαρμογής, καθώς το Ionic μπορεί να χρησιμοποιήσει Angular, ως εργαλείο ανάπτυξης. Έτσι περιοριστήκαμε σε τροποποιήσεις εμφάνισης (styling) με αλλαγή χρωμάτων για πιο άνετη πλοήγηση σε κινητή εφαρμογή, αφαίρεση διαγραμμάτων πολύπλοκων για χρήση σε κινητό, αλλά και τροποποίηση πινάκων, γραφημάτων και γράφων, ώστε να είναι λειτουργικά αλλά και οπτικά ικανοποιητικά σε μικρότερες συσκευές. Μερικές από τις τροποποιήσεις παρουσιάζονται στην Εικόνα 27 και Εικόνα 28.



Εικόνα 27: Αρχική σελίδα / Μπάρα πλοήγησης



Εικόνα 28: Προσαρμογή Γραφημάτων

Κεφάλαιο 5: Συμπεράσματα

Κατά την παρούσα εργασία έγινε μια μελέτη των υπαρχουσών λύσεων στην ασφάλεια υποδομών. Με βάση το τεχνολογικό background που μου δόθηκε από το Πανεπιστήμιο Πελοποννήσου, πρότεινα επεκτάσεις και παρέδωσα μία σειρά από λύσεις για την επέκταση της ασφάλειας υποδομών σε περισσότερους από έναν τομείς, λαμβάνοντας υπόψιν τόσο τις φυσικές όσο και τις διαδικτυακές απειλές.

Στο πλαίσιο της εργασίας αναπτύχθηκε μία εφαρμογή κινητών τηλεφώνων με χρήση τεχνολογίας react native, ενώ το μοντέλο STIX επεκτάθηκε, έτσι ώστε να επιτρέπει την εύκολη προσαρμογή του σε οποιονδήποτε τομέα.

Χρησιμοποιήθηκε μια νέα middleware εφαρμογή γραμμένη σε Laravel που σκοπό έχει να λαμβάνει δεδομένα είτε κατευθείαν από την πλατφόρμα ή από το Service επίπεδο, όπως ορίζεται στο έργο FINSEC.

Προτάθηκε επίσης η χρήση GraphQL για την επικοινωνία μεταξύ των services, αλλά και η υποστήριξη GraphQL προς εξωτερικά components, όπως η διεπαφή χρήστη (Dashboard).

Σαν επόμενα βήματα προτείναμε έναν νέο τρόπο επικοινωνίας μεταξύ οργανισμών με χρήση του συστήματος Apache Kafka, ενώ χωρίστηκαν αρχιτεκτονικά περαιτέρω σε Cyber και Physical τα επιμέρους προγράμματα του συστήματος, δίνοντας έτσι τη δυνατότητα εύκολης επέκτασης αλλά και χρήσης μόνο των στοιχείων εκείνων που επιθυμεί ο εκάστοτε οργανισμός.

Επίσης, πολλές ακόμα μικρές λεπτομέρειες θα μπορούσαν να καταστήσουν το παρόν σύστημα ακόμα πιο αποδοτικό, αλλά και φιλικότερο προς τον τελικό χρήστη. Ενδεικτικά αναφέρουμε την εισαγωγή αυτόματα πυροδοτούμενων εργασιών (cronjobs) για την ανίχνευση νέων ευπαθειών, τον εκ νέου υπολογισμό ρίσκων ή τη διαγραφή παλαιών δεδομένων. Στη διεπαφή χρήστη θα μπορούσε επίσης να δίνεται η δυνατότητα σε ένα διαχειριστή του οργανισμού να επιλέγει τη μορφή οπτικοποίησης των δεδομένων, ενώ ακόμα περισσότερα διαγράμματα θα μπορούσαν να εξαχθούν από την επεξεργασία των διαθέσιμων δεδομένων. Για τον σκοπό αυτό θα μπορούσαν να χρησιμοποιηθούν εργαλεία όπως το Graphana, των οποίων τα αποτελέσματα δεν εξαρτάται από τον τρόπο αποθήκευσης και παρέχουν μια μεγάλη γκάμα διαθέσιμων γραφημάτων.

Βιβλιογραφία

1. Karagiannis, I. , Mavrogiannis, K. , Soldatos, J. , Drakoulis, D. , Troiano, E. , Polyviou , A. , Blockchain based Sharing of Security Information for Critical Infrastructures of the Finance Sector, ESORICS 2019 on 23-27 Sep 2019, Luxembourg
2. Gazzarata G. , Troiano E. , Verderame L. , Aiello M., Vaccari I., Cambiaso E. , Merlo A. ,
3. Cyber-Physical Data Model for Financial Critical Infrastructures. 2021 Jan 28; 12618: 48–63. Published online 2021 Jan 28. doi: 10.1007/978-3-030-69781-5_4
4. Barnum, S.: Standardizing cyber threat intelligence information with the structured threat information expression (STIX). MITRE Corporation
5. Financial Services Information Sharing and Analysis Center. https://www.fsisac.com/. Accessed 09 July 2019
6. Jordan, B., Piazza, R., Wunder, B.: Stix Core Concepts. https://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.html. Accessed 09 July 2019 12. Wunder, J., Davidson, M., Jordan, B. TAXII™ Version 2.0. https://docs.google.com/document/d/1Jv9ICjUNZrOnwUXtenB1QcnBLO35RnjQcJLsa1mGSkI/edit. Accessed 09 July 2019
7. Common Vulnerability Scoring System. https://www.first.org/cvss/v3.0/cvss-v30-specification_v1.9.pdf. Accessed 09 July 2019

Διαδικτυακές πηγές

FINSEC Reference Architecture

https://finsecurity.eu/digital-finance-academy-for-security/finsec-ra-whitepaper/

REST

https://en.wikipedia.org/wiki/Representational_state_transfer

Angular	https://angular.io/
Laravel	https://laravel.com/
Docker	https://www.Docker.com/
Docker hub	https://hub.Docker.com/
Docker Compose	https://docs.Docker.com/compose/
JSON	https://www.JSON.org/JSON-en.html
JSON Schema	https://JSON-schema.org/
Ionic	https://ionicframework.com/
GraphQL	https://graphql.org/
Kafka	https://kafka.apache.org/

Zookeeper	https://zookeeper.apache.org/
Open API	https://www.openapis.org/
CVSS	https://www.first.org/cvss/
MITIGATE	https://cordis.europa.eu/project/id/653212
Hyperledger Fabric	https://www.hyperledger.org/use/fabric
Kubernetes	https://kubernetes.io/
JSON Web Tokens	https://jwt.io/
Sonarqube	https://docs.sonarqube.org/latest/
Anchore	https://anchore.com/
Dynatrace	https://www.dynatrace.com/
Datadog	https://www.datadoghq.com/
XML	https://www.w3.org/XML/
YAML	https://yaml.org/
Apache Echarts	https://echarts.apache.org/en/index.html
SASS	https://sass-lang.com/
LESS	https://lesscss.org/
Typescript	https://www.typescriptlang.org/
MongoDB	https://www.mongodb.com/
MySQL	https://www.mysql.com/
Graphana	https://grafana.com/