Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Ασφάλεια Υπολογιστών και Δικτύων*

Μεταπτυχιακή Διατριβή

Cyber Range Scenario Visualization GUI & Integration of Automated Red Attacks Scripts

Γεωργία Παπανεοφύτου

Επιβλέπουσα Καθηγήτρια
Αδαμαντίνη Περατικού

Νοέμβριος 2022

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Ασφάλεια Υπολογιστών και Δικτύων*

Μεταπτυχιακή Διατριβή

Cyber Range Scenario Visualization GUI & Integration of Automated Red Attacks Scripts

Γεωργία Παπανεοφύτου

Επιβλέπουσα Καθηγήτρια
Αδαμαντίνη Περατικού

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών στην Ασφάλεια Υπολογιστών και Δικτύων από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών του Ανοικτού Πανεπιστημίου Κύπρου.

Νοέμβριος 2022

ΛΕΥΚΗ ΣΕΛΙΔΑ

Περίληψη

Η διατριβή αυτή στοχεύει, πρώτον, στη δημιουργία γραφικής διεπαφής χρήστη (GUI) για την παρακολούθηση σεναρίων που εκτελούνται στο Cyber Range του Ανοικτού Πανεπιστημίου Κύπρου και δεύτερον στη δημιουργία ενός μηχανισμού αυτοματοποίησης των επιθέσεων που εκτελούνται μέσω scripts.

Για την επίτευξη των στόχων αυτών αναπτύχθηκε διαδικτυακή εφαρμογή, η υλοποίηση της οποίας έγινε σε τρεις φάσεις. Αρχικά δημιουργήθηκε το γραφικό περιβάλλον βάση δοθέντος προτύπου και ακολούθησε ο μηχανισμός εκτέλεσης των scripts κατά απαίτηση του χρήστη. Τέλος, σχεδιάστηκε και υλοποιήθηκε ένας μηχανισμός αυτοματοποίησης των επιθέσεων, που χρησιμοποιεί ρυθμίσεις και τα συμβάντα που καταγράφονται στο Cyber Range, για να εκτελέσει επιθέσεις χωρίς να χρειάζεται ανθρώπινη παρέμβαση σε πραγματικό χρόνο.

Summary

This dissertation aims, firstly, in the creation of a Graphical User Interface (GUI) for the observation of scenarios running in the Cyber Range of Open University of Cyprus and secondly in the creation of a mechanism that automates the execution of attack scripts.

To fulfill above goals, a web application was developed in three phases. Initially, the graphical environment was created based on a given prototype, followed by a mechanism for executing scrips on user's demand. Finally, a mechanism for automating the attacks was designed and implemented, which uses configurations and events recorded in the Cyber Range, in order to execute attacks without the need of human interaction in real time.

Ευχαριστίες

Θερμές ευχαριστίες στην επιβλέπουσα καθηγήτρια Αδαμαντίνη Περατικού για τη στήριξη και καθοδήγησή της σε όλη τη διαδικασία.

Περιεχόμενα

Κεφάλαιο 1
1 Εισαγωγή

Τα cyber ranges είναι περιβάλλοντα που προσομοιώνουν δίκτυα, συστήματα και εργαλεία με σκοπό να προσφέρουν ένα ελεγχόμενο εικονικό περιβάλλον που μπορεί να αξιοποιηθεί για εκπαίδευση επαγγελματιών κυβερνοασφάλειας, αξιολόγηση της ασφάλειας του προσομοιωμένου περιβάλλοντος αλλά και για δοκιμή νέων τεχνολογιών. Χρησιμοποιούνται από κυβερνητικούς οργανισμούς, ιδιωτικές εταιρίες καθώς και για ακαδημαϊκούς σκοπούς. Οι στόχοι αυτοί υλοποιούνται συνήθως μέσω ασκήσεων, που πολλές φορές παίρνουν τη μορφή παιχνιδιού με τη συμμετοχή διαφόρων ομάδων. Οι κύριες ομάδες είναι η λευκή, η οποία δημιουργεί το περιβάλλον και τα σενάρια εκπαίδευσης, η μπλε, που είναι υπεύθυνη για τον εντοπισμό ευπαθειών και την αποτροπή επιθέσεων και η κόκκινη ομάδα, η οποία επιτίθεται στο σύστημα και προσπαθεί να εκμεταλλευτεί τις ευπάθειες που υπάρχουν στο περιβάλλον. Μια επιθυμητή δυνατότητα των cyber range περιβαλλόντων είναι η αυτοματοποίηση των ρόλων και των εργαλείων που χρησιμοποιούν, ούτως ώστε να επιτρέπεται η χωριστή εκπαίδευση τους, αλλά και η επανάληψη των σεναρίων με εύκολο και συνεπή τρόπο. Επιπλέον, η ύπαρξη εργαλείων που επιτρέπουν την επίβλεψη σε πραγματικό χρόνο, βοηθά την επιβεβαίωση της ομαλής λειτουργίας των ασκήσεων και μειώνει το χρόνο της αξιολόγησης των διαδικασιών από τους υπευθύνους.

1.1 Σκοπός Έρευνας

Σκοπός της διατριβής αυτής είναι η αυτοματοποίηση, σε cyber range περιβάλλον, attack scripts που χρησιμοποιούνται από την κόκκινη ομάδα, καθώς και η δημιουργία γραφικής διεπαφής χρήστη (GUI) για την αποδοτική παρακολούθηση των σεναρίων που εκτελούνται από τους διαχειριστές.

1.2 Βασικά Ερευνητικά Ερωτήματα

Τα βασικά ερευνητικά ερωτήματα που θα απαντηθούν είναι

1. Ποια scripts χρησιμοποιούνται από την κόκκινη ομάδα για επιθέσεις σε δίκτυα και συστήματα;

2. Πώς αυτά μπορούν να αυτοματοποιηθούν και ενσωματωθούν σε cyber range περιβάλλον;

3. Πώς μπορεί ο διαχειριστής να παρακολουθεί σε πραγματικό χρόνο και αποδοτικά την εξέλιξη των σεναρίων που εκτελούνται στο cyber range περιβάλλον;

1.3 Αναγκαιότητα και Σπουδαιότητα Έρευνας

Τα cyber range περιβάλλοντα είναι ιδανική λύση για την εκπαίδευση επαγγελματιών ασφάλειας πληροφοριακών συστημάτων. Η δημιουργία σεναρίων αντιπροσωπευτικών της πραγματικότητας και η παρακολούθηση της εξέλιξής τους καθώς εκτελούνται, αποτελούν, όμως, δύσκολες και χρονοβόρες διαδικασίες. Η έλλειψη ευχρηστίας, απλότητας και αυτοματοποίησης είναι το βασικό πρόβλημα που πολλές φορές αποτρέπει την προσομοίωση πραγματικών συνθηκών, την αναπαραγωγή των σεναρίων, καθώς και την άμεση παρακολούθηση και αξιολόγηση των ασκήσεων από τους διαχειριστές. Η διατριβή αυτή στοχεύει στην ενσωμάτωση και αυτοματοποίηση, σε cyber range περιβάλλον, των scripts που συνήθως χρησιμοποιούνται από τις ομάδες επίθεσης, καθώς και τη διευκόλυνση της παρακολούθησης των ασκήσεων από τους διαχειριστές μέσω γραφικού περιβάλλοντος.

Κεφάλαιο 2
2 Ανασκόπηση Βιβλιογραφίας

Το κεφάλαιο αυτό ξεκινά με κάποιους ορισμούς και συνεχίζει με το θεωρητικό υπόβαθρο, που θα βοηθήσουν στην καλύτερη κατανόηση των όσων θα ακολουθήσουν. Ακολουθεί έρευνα που αφορά την κόκκινη ομάδα και τα εργαλεία που χρησιμοποιεί στις επιθέσεις της και κλείνει με την ανασκόπηση της βιβλιογραφίας για σχετικές έρευνες που αφορούν τα cyber ranges και σχολιάζεται αν παρέχουν γραφικό περιβάλλον και χρησιμοποιούν αυτοματοποιημένες επιθέσεις.

2.1 Ορισμοί

Παρατίθενται οι παρακάτω σχετικοί ορισμοί, που θα χρησιμοποιηθούν στη συνέχεια.

2.1.1 Red Team

Σκοπός της κόκκινης ομάδας είναι να μιμηθεί τον τρόπο σκέψης και δράσης ενός επιτιθέμενου και να προσπαθήσει να εντοπίσει κενά ασφαλείας στο σύστημα. Χρειάζεται ανεπτυγμένες ικανότητες και βαθιά γνώση της τεχνολογίας και των εργαλείων άμυνας και επίθεσης, καθώς και των τακτικών που ακολουθούνται σε πραγματικές επιθέσεις. Προβαίνει σε εκμετάλλευση ευπαθειών για να βλάψει το σύστημα ή να υποκλέψει δεδομένα, αφήνοντας όσο το δυνατόν λιγότερα ίχνη για να αποφύγει τον εντοπισμό της παραβίασης από τη μπλε ομάδα.

2.1.2 Blue Team

Η μπλε ομάδα αποσκοπεί στην υπεράσπιση και προστασία του συστήματος από τις κακόβουλες επιθέσεις. Καλείται πρωτίστως να προλάβει την παρείσδυση, εντοπίζοντας πρώτη τις ευπάθειες και απαλείφοντάς τις προτού τύχουν εκμετάλλευσης από την κόκκινη ομάδα. Σε περίπτωση παραβίασης πρέπει να την εντοπίσει και να αντιδράσει το

γρηγορότερο δυνατόν, περιορίζοντας τις συνέπειές της και αποκαθιστώντας τη λειτουργικότητα του συστήματος.

2.1.3 White Team

Η λευκή ομάδα είναι υπεύθυνη για τη δημιουργία και την ομαλή εκτέλεση της άσκησης. Θέτει τους στόχους, δημιουργεί το ανάλογο σενάριο και γνωστοποιεί τους κανόνες που πρέπει να ακολουθηθούν από την κόκκινη και μπλε ομάδα. Παρακολουθεί την έκβαση καθ' όλη τη διάρκεια και με το πέρας της άσκησης θα κληθεί να αναλύσει τα αποτελέσματα.

2.1.4 Capture the Flag (CTF)

Διαγωνισμοί κυβερνοασφάλειας με τη μορφή παιχνιδιού. Οι συμμετέχοντες, μεμονωμένα άτομα ή ομάδες, λύνουν προκλήσεις (challenges) ασφαλείας σε προσομοιωμένα περιβάλλοντα και κερδίζουν βαθμούς βρίσκοντας τις «σημαίες». Παραλληλίζεται με το παιχνίδι εξωτερικού χώρου, όπου η κάθε ομάδα προσπαθεί να πάρει τη (φυσική) σημαία της άλλης ομάδας -χωρίς να συλληφθεί- για να νικήσει. Σε κάθε σενάριο προσομοιάζονται επιλεγμένοι κίνδυνοι που συναντώνται στον κυβερνοχώρο, ανάλογα με τους στόχους της άσκησης. Μπορεί να υπάρχουν δύο ομάδες, κόκκινη και μπλε, με την πρώτη να προσπαθεί να πάρει τις σημαίες, τις οποίες η δεύτερη πρέπει να προστατέψει ή μόνο κόκκινη ομάδα (ή άτομο) που ψάχνει να βρει τις σημαίες χωρίς να υπάρχει αντίπαλος που να τις προστατεύει.

2.1.5 Cyber Defence Exercise (CDX)

Άσκηση κυβερνοασφάλειας που εστιάζει στη δοκιμή των αμυντικών ικανοτήτων των συμμετεχόντων. Χρησιμοποιείται για τον έλεγχο της ετοιμότητας των μπλε ομάδων να προστατέψουν τα συστήματα και να χειριστούν περιστατικά κακόβουλης παρείσφρησης.

2.1.6 Penetration Testing

Εξουσιοδοτημένοι επαγγελματίες χρησιμοποιούν εργαλεία και τεχνικές που χρησιμοποιούν κακόβουλοι δράστες, με σκοπό να εντοπίσουν πρώτοι τις αδυναμίες του συστήματος και να ενημερώσουν τους ενδιαφερόμενους. Περιλαμβάνει την εκτενή συλλογή πληροφοριών (reconnaissance), την αναζήτηση ευπαθειών (vulnerability

scanning), την εκμετάλλευση τους (exploit), την απόκρυψη των ιχνών της παραβίασης και τέλος την ανάλυση και τη δημιουργία αναφοράς για τα ευρήματα του ελέγχου.

2.1.7 Social Engineering

Τεχνικές που χρησιμοποιούνται για την εξαπάτηση ανθρώπων και την παραπλάνησή τους με στόχο να εκθέσουν ευαίσθητες πληροφορίες ή να δώσουν πρόσβαση σε συστήματα σε ανεξουσιοδότητα άτομα. Οι δράστες συλλέγουν συνήθως πληροφορίες για τα υποψήφια θύματα, ετοιμάζουν ένα σενάριο-ιστορία και στη συνέχεια το επικοινωνούν προσπαθώντας να κερδίσουν την εμπιστοσύνη των θυμάτων τους και να τους παρασύρουν σε επιβλαβείς ενέργειες. Μερικές από τις πιο γνωστές τεχνικές είναι το phishing που στοχεύει στην απόσπαση πληροφοριών συνήθως μέσω ηλεκτρονικού ταχυδρομείου, το vishing (voice – phishing μέσω τηλεφώνου), καθώς και η προσποίηση ότι είναι κάποιος άλλος (impersonation).

2.2 Cyber Ranges

Η ολοένα αυξημένη συνδεσιμότητα, διαμοιρασμός δεδομένων και προσφορά διαδικτυακών υπηρεσιών, εκτός από τις ευκολίες, αυξάνει και την έκθεση στον κίνδυνο. Η ανάπτυξη της τεχνολογίας δε βοηθά μόνο στη διευκόλυνση της ζωής μας με νέες υπηρεσίες, αλλά και τους διαθέσιμους πόρους και πεδία επίθεσης κακόβουλων δραστών του διαδικτύου.

Σύμφωνα με την αναφορά ENISA Threat Landscape 2022 [1], οι κορυφαίοι κίνδυνοι που εντοπίζονται στην κυβερνοασφάλεια είναι τα Ransomware, Malware, Social Engineering και κυρίως phishing, απειλές κατά των δεδομένων (data breach – leak), απειλές κατά της διαθεσιμότητας και άρνηση εξυπηρέτησης (Denial of Service, BGP hijacking), καθώς επίσης και η παραπληροφόρηση (Disinformation – misinformation) που ενισχύεται διαρκώς με τη χρήση της τεχνητής νοημοσύνης και τέλος οι επιθέσεις στην εφοδιαστική αλυσίδα (Supply-chain attacks) με έμφαση στις υπηρεσίες που χρησιμοποιούν τεχνολογίες νέφους (cloud). Όλοι οι τομείς δραστηριοτήτων φαίνεται να έχουν επηρεαστεί περισσότερο ή λιγότερο από τα καταγεγραμμένα περιστατικά, ενώ οι επιτιθέμενοι προέρχονται κυρίως από τις τέσσερις βασικές ομάδες των χορηγούμενων

από το κράτος (state-sponsored), κυβερνοεγκληματίες (cybercrime actors), hackers επί πληρωμή και hacktivists.

Για την αντιμετώπιση όλων αυτών των κινδύνων, οι δημόσιοι και ιδιωτικοί φορείς οφείλουν να ενισχύσουν την προστασία των υπηρεσιών που παρέχουν και των δεδομένων των χρηστών τους. Για να τα καταφέρουν χρειάζονται καλά εκπαιδευμένους επαγγελματίες, καταρτισμένους κατάλληλα να δράσουν προληπτικά, αλλά και άμεσα σε περίπτωση επιτυχούς παραβίασης των συστημάτων τους από κακόβουλους χρήστες.

Με σκοπό τη συνεχή εκπαίδευση επαγγελματιών του χώρου και τη μείωση της έλλειψης δεξιοτήτων που παρατηρείται, κράτη και οργανισμοί επιδίδονται στη δημιουργία ασκήσεων, που συχνά παίρνουν τη μορφή διαγωνισμών, γνωστών ως Capture The Flag ή άλλες φορές εστιάζουν στην ενίσχυση των αμυντικών ικανοτήτων της μπλε ομάδας δημιουργώντας Cyber Defence Exercises. Οι ασκήσεις πραγματοποιούνται σε εικονικά περιβάλλοντα -απομονωμένα από τα πραγματικά συστήματα- που υλοποιούνται με προσομοίωση ή εξομοίωση ανάλογα με το σκοπό και τις δυνατότητες και κατά αντιστοιχία με τον στρατιωτικό όρο, ονομάζονται Ψηφιακά Πεδία Βολής ή κοινώς Cyber Ranges.

Μια τέτοια άσκηση είναι το Locked Shields [2], που οργανώνεται από το NATO και διεξάγεται ετησίως από το 2010. Στα πλαίσια του διαγωνισμού, μπλε ομάδες από μέλη-κράτη του οργανισμού καλούνται να διαχειριστούν μεγάλης κλίμακας περιστατικό παραβίασης και όλες τις συνέπειές του.

Για την εκπαίδευση κόκκινων ομάδων, το NATO διοργανώνει από το 2016 την ετήσια άσκηση Crossed Swords, τα σενάρια της οποίας βασίζονται σε πραγματικά γεγονότα κυβερνοεπιθέσεων. Επαγγελματίες από αυτές τις ομάδες συμμετέχουν αργότερα ως επιτιθέμενοι στην άσκηση Locked Shields [3].

Τα τελευταία χρόνια διαγωνισμός διοργανώνεται κι από το European Union Agency For Cybersecurity (ENISA), το European Cyber Security Challenge (ECSC), όπου συμμετέχουν ομάδες νεαρών ατόμων από ευρωπαϊκές χώρες με σκοπό την ενθάρρυνση τους να επιλέξουν να εργαστούν στον τομέα της κυβερνοασφάλειας [4].

Η εταιρεία SANS παρέχει μια γκάμα cyber ranges με το εμπορικό Netwars να προσφέρεται για την εκπαίδευση αλλά και ανάπτυξη των ικανοτήτων των συμμετεχόντων. Η εκπαίδευση μπορεί να είναι ατομική ή ομαδική και η πλατφόρμα παρέχει τη δυνατότητα βοήθειας σε περίπτωση δυσκολιών. Το Netwars Tournament έχει τη μορφή διαγωνισμού, ενώ το Netwars Continuous απευθύνεται κυρίως σε μεμονωμένα άτομα που θέλουν να χρησιμοποιήσουν την πλατφόρμα για εξάσκηση των ικανοτήτων τους [5].

Η SANS κάθε χρόνο τα Χριστούγεννα δημιουργεί και το εορταστικό Holiday Hack Challenge που είναι δωρεάν και απευθύνεται σε όλα τα επίπεδα ικανοτήτων. Είναι διασκεδαστικό, αφού ο συμμετέχοντας πρέπει να βοηθήσει τον Άγιο Βασίλη να σώσει τα Χριστούγεννα από κακόβουλες επιθέσεις και επιπλέον υπάρχει επιβράβευση για τους τελικούς νικητές [6].

Σωρεία από cyber ranges έχουν αναπτυχθεί για σκοπούς εκπαίδευσης, όπως είναι το ανοιχτού κώδικα λογισμικό KYPO [7] ή το εμπορικό Cyberbit [8] ή για συγκεκριμένους τομείς δραστηριοτήτων όπως είναι το Maritime Cyber Range [9] και το Cyber-MAR για ναυτιλιακούς σκοπούς [10] ή το SPIDER [11] για την τεχνολογία 5G.

Η πληθώρα των cyber ranges καταδεικνύει τις πολλαπλές και διαφορετικές ανάγκες που υπάρχουν και είναι ενδεικτική της ανάπτυξης που θα τύχουν στο μέλλον. Αναμφίβολα, αναλόγως του είδους τους, οι ασκήσεις κυβερνοασφάλειας μπορούν να οδηγήσουν σε αναγνώριση ευπαθειών του συστήματος, αδυναμιών ατόμων και ομάδων, καθώς και στον εντοπισμό προβλημάτων στις διαδικασίες διαχείρισης των περιστατικών παραβίασης ασφαλείας, καταδεικνύοντας έτσι τα τρωτά σημεία του οργανισμού για τα οποία θα πρέπει να ληφθούν μετρά βελτίωσης και ενίσχυσης της ασφάλειας. Λόγω του απομονωμένου και ασφαλούς περιβάλλοντος στο οποίο διενεργούνται, μπορούν να διευρύνουν τις δεξιότητες των ομάδων και να δοκιμαστούν τα εργαλεία και οι τεχνολογίες που χρησιμοποιούνται για την προστασία του συστήματος, χωρίς να επηρεάζεται η κανονική λειτουργία του οργανισμού.

2.3 Red Team Attacks

Η κόκκινη ομάδα έχει ως στόχο την ανακάλυψη των τρωτών σημείων στην ασφάλεια του υπό εξέταση οργανισμού. Αυτό αφορά την κυβερνοασφάλεια στην τεχνολογία και την υποδομή, αλλά και τη φυσική ασφάλεια των κτιρίων και τους ίδιους τους ανθρώπους που εργάζονται στον οργανισμό. Αν, για παράδειγμα, τα κτίρια δεν είναι ασφαλή, μπορεί ένας εισβολέας να έχει άμεση πρόσβαση σε εξοπλισμό και δεδομένα και να μη χρειαστεί καν να προσπαθήσει να παρακάμψει την ασφάλεια του δικτύου για να τα αποκτήσει. Όσον αφορά τους ανθρώπους, αν δεν είναι ενημερωμένοι και υποψιασμένοι, μπορούν να πέσουν θύματα social engineering και να εκθέσουν τα διαπιστευτήρια τους και κατά συνέπεια άλλα ευαίσθητα δεδομένα που κατέχουν. Επομένως, για την επίτευξη των στόχων της η red team χρησιμοποιεί penetration testing, social engineering, αλλά και αφήνεται να καταφύγει σε οποιοδήποτε άλλο ευφάνταστο τρόπο κρίνει απαραίτητο για να σπάσει τις άμυνες και να καταφέρει να διεισδύσει στον οργανισμό.

2.3.1 MITRE ATT&CK

Το MITRE ATT&CK είναι μια βάση γνωστών επιθετικών τακτικών, τεχνικών και διαδικασιών (tactics, techniques, procedures - TTPs), που ταξινομούνται με βάση την πλατφόρμα (π.χ. Linux, Office 365) και το σκοπό τους (π.χ. Reconnaissance, Privilege escalation, Exfiltration). Η τακτική περιγράφεται ως το «γιατί» προβαίνει κάποιος σε αυτή την επίθεση, η τεχνική είναι το «πώς» υλοποιείται η τακτική και μπορεί να περιλαμβάνει και πιο λεπτομερείς υπο-τεχνικές, ενώ η διαδικασία είναι η συγκεκριμενοποίηση των τακτικών με εργαλεία και μηχανισμούς. Για κάθε ένα από αυτά περιγράφεται ο τρόπος που έχει χρησιμοποιηθεί, πώς μπορεί να ανιχνευθεί και να περιοριστεί. Η βάση μπορεί να χρησιμοποιηθεί για την ενημέρωση, αλλά και τη δημιουργία πλάνων επίθεσης από τις κόκκινες ομάδες [12].

2.3.2 MITRE CVE

Επιπλέον, ο οργανισμός MITRE διατηρεί το πρόγραμμα Common Vulnerabilities and Exposures (CVE) [13], όπου καταγράφονται με τυποποιημένο τρόπο όλες οι δημοσιοποιημένες ευπάθειες που εντοπίζονται σε υλικό και λογισμικό διεθνώς. Οι εγγραφές CVE τροφοδοτούνται σε άλλες βάσεις, όπως είναι η U.S National Vulnerability Database (NVD), όπου αντιστοιχίζονται με περισσότερες λεπτομέρειες και μετρικές, όπως είναι η βαθμολογία από 0 μέχρι 10 με βάση το Common Vulnerability Scoring System (CVSS), που δηλώνει το επίπεδο κινδύνου της κάθε ευπάθειας, με το 10 να

σημαίνει την υψηλότερη επικινδυνότητα. Η βάση αυτή χρησιμοποιείται κατά κόρον από τα προγράμματα σάρωσης ευπαθειών (vulnerability scanners) που εντοπίζουν το επηρεασμένο υλικό και λογισμικό στο σύστημα που εξετάζουν και εμφανίζουν τις αδυναμίες τις οποίες θα μπορούσε να εκμεταλλευτεί ένας επιτιθέμενος.

2.3.3 Red Team Toolkit

Για τα cyber range περιβάλλοντα που ασχολούμαστε, θα εστιάσουμε σε εργαλεία που έχουν αναπτυχθεί για penetration testing. Στην αγορά κυκλοφορούν αρκετά προγράμματα που έχουν αναπτυχθεί για το σκοπό αυτό, αλλά θα επικεντρωθούμε σε λογισμικά ανοιχτού κώδικα, τα οποία θα μπορούν να ενσωματωθούν στο δικό μας λογισμικό. Προϋπόθεση για αυτό και την επίτευξη της επιθυμητής αυτοματοποίησης είναι η ικανότητα να εκτελεστούν χωρίς την ανάγκη διάδρασης με το χρήστη, μετά τον καθορισμό των αρχικών παραμέτρων.

theHarvester

Εργαλείο συλλογής πληροφοριών. Εμφανίζει πληροφορίες για το domain, emails, ονόματα εργαζομένων, banners κ.α. από δημόσιες πηγές με τη χρήση πολλαπλών μηχανών αναζήτησης [14].

SpiderFoot

Άλλο ένα εργαλείο για τη φάση του reconnaissance. Επικοινωνεί με περισσότερες από 100 δημόσιες πηγές πληροφοριών για να συλλέξει IPs, domains, διευθύνσεις ηλεκτρονικού ταχυδρομείου, αν αυτές έχουν καταγραφεί σε λίστες παραβίασης (HaveIBeenPwned) κλπ. Είναι γραμμένο σε Python και διατίθεται για Windows, macOS και Linux συστήματα. Από την έκδοση 3.0 μπορεί να εκτελεστεί αποκλειστικά στη γραμμή εντολών, χωρίς την εκκίνηση εξυπηρετητή που ήταν προηγουμένως απαραίτητη [15].

Nmap

Το όνομα του είναι συντομογραφία για το Network Mapper. Ξεκίνησε ως port scanner, αλλά πλέον είναι πλήρης σαρωτής δικτύου. Στέλνει πακέτα στο δίκτυο και αναλύοντας τις απαντήσεις που λαμβάνει, εντοπίζει τις συνδεδεμένες συσκευές, τα λειτουργικά συστήματα που τρέχουν, τις ανοιχτές θύρες και τις υπηρεσίες που εκτελούνται σε αυτές.

Επιπρόσθετα, με το scripting engine που διαθέτει μπορεί να ελέγξει για γνωστές ευπάθειες, μέχρι και να τις εκμεταλλευτεί, όπως για παράδειγμα να προβεί σε SQL Injection σε ανοιχτή θύρα MySQL. Διατίθενται εκδόσεις για όλα τα δημοφιλή λειτουργικά συστήματα.

John the Ripper

Εργαλείο ανάκτησης/σπασίματος κωδικών offline, δοσμένου ενός hash. Υποστηρίζει διαφόρους τύπους κατακερματισμού (hashing), όπως είναι οι MD5, DES, SHA κλπ., τους οποίους αναγνωρίζει και αυτόματα. Δέχεται κανόνες για το μοτίβο και το μέγεθος των κωδικών ώστε να περιοριστεί η αναζήτηση και περιλαμβάνει modes αναζήτησης, όπως είναι το brute-force ή η αναζήτηση με λεξικό (dictionary).

HashCat

Άλλο ένα εργαλείο ανάκτησης κωδικών, επίσης για offline χρήση. Περιλαμβάνει διάφορα modes εκτέλεσης, όπως brute-force, dictionary, mask ή rule-based attacks. Χρησιμοποιεί την κάρτα γραφικών για να μειώσει τον απαιτούμενο χρόνο αναζήτησης. Υποστηρίζει πάνω από 350 αλγορίθμους, αναμεσά τους οι SHA1-2-3, Kerberos, RAR κλπ.

THC Hydra

Χρησιμοποιείται για το σπάσιμο log-in κωδικών στο διαδίκτυο (online). Υποστηρίζει διάφορα πρωτόκολλα, όπως HTTP(S)-FORM-POST, FTP, SSH, RDP κλπ. Είναι πολύ γρήγορο στις brute-force επιθέσεις και στις επιθέσεις με βάση λεξικά ή rainbow tables. Είναι διαθέσιμο σε όλα τα γνωστά λειτουργικά συστήματα και μπορεί να εκτελεστεί στη γραμμή εντολών ή με γραφικό περιβάλλον.

Metasploit

Αναμφίβολα το δημοφιλέστερο framework για penetration testing. Μπορεί να χρησιμοποιηθεί για εφαρμογές, δίκτυα και εξυπηρετητές. Υπάρχει η δωρεάν εκδοχή Framework, αλλά και η εμπορική Pro εκδοχή και μπορεί να εγκατασταθεί σε συστήματα Windows και Linux. Διαθέτει γραφικό περιβάλλον, αλλά δύναται να εκτελεστεί και από τη γραμμή εντολών msfconsole. Μπορεί να χρησιμοποιηθεί για password cracking, vulnerability scanning, network enumeration, γένεση κώδικα (payload) που μπορεί να εκτελεστεί στο επιτυχώς παραβιασμένο σύστημα, εκτέλεση του exploit (εκμετάλλευση

της ευπάθειας), αλλά και post-exploit λειτουργίες. Παρέχει μεγάλη γκάμα από exploits και με τα modules που διαθέτει, ενσωματώνει και λειτουργίες από άλλα εργαλεία. Επιπλέον, μπορεί να ενισχυθεί με custom-made exploits.

Atomic Red Team

Είναι μια συλλογή από scripts που αντιστοιχούν σε στοιχεία της βάσης του MITRE ATT&CK [16]. Περιλαμβάνει πληθώρα scripts για διάφορα λειτουργικά συστήματα, προγράμματα και υπηρεσίες. Μπορεί κανείς να εκτελέσει τα scripts αντιγράφοντας τον κώδικα ή να χρησιμοποιήσει κάποιο εργαλείο, όπως το Invoke-AtomicRedTeam [17] για το Powershell των Windows.

Nikto

Open-source εργαλείο που ειδικεύεται στην αναζήτηση ευπαθειών σε ιστοσελίδες και εφαρμογές διαδικτύου. Εντοπίζει σφάλματα στη ρύθμιση των εξυπηρετητών, αναγνωρίζει την ταυτότητά τους και βρίσκει απαρχαιωμένες ή προβληματικές εκδόσεις του λογισμικού τους. Είναι γραμμένο στη scripting γλώσσα PERL και εκτελείται από τη γραμμή εντολών. Οι αναφορές του μπορούν να είναι σε διάφορες μορφές, όπως απλό κείμενο, HTTP ή CSV. Δεν είναι stealthy tool και μπορεί να ανιχνευθεί από συστήματα IPS/IDS [18].

sqlmap

Αυτοματοποιεί την αναζήτηση και εκμετάλλευση των SQL injection flaws σε ένα εύρος συστημάτων διαχείρισης βάσεων δεδομένων, όπως είναι η MySQL, PostgreSQL, MariaDB και Microsoft SQL Server. Είναι γραμμένο σε Python, οπότε μπορεί να εκτελεστεί στη γραμμή εντολών, αν και ανάλογα με την εντολή μπορεί να χρειαστεί διάδραση με το χρήστη. Μπορεί να χρησιμοποιηθεί για fingerprinting, εύρεση κωδικών χρηστών, υποκλοπή δεδομένων ή ακόμα και αρχείων από το σύστημα [19].

Exploit Database

Συλλογή από δημόσια exploits, που συντηρείται από την κοινότητα του Offensive Security. Μπορεί κανείς να αναζητήσει ευπάθειες με βάση την πλατφόρμα, το CVE id και άλλες λέξεις κλειδιά και να βρει αντίστοιχο κώδικα κελύφους που τις αποδεικνύει. Η απόδειξη αυτή θα μπορούσε να χρησιμοποιηθεί για εντοπισμό ή και εκμετάλλευση της

ευπάθειας ανάλογα. Έχει δημιουργηθεί και εργαλείο αναζήτησης, το SearchSploit, το οποίο μπορεί να εκτελεστεί στη γραμμή εντολών [20].

OWASP ZAP

Δωρεάν εργαλείο για penetration testing σε web εφαρμογές. Το Zed Attack Proxy είναι διαθέσιμο για όλα τα γνωστά λειτουργικά συστήματα. Παρέχεται με γραφικό περιβάλλον, αλλά για την απλούστερη χρήση του μπορεί να κληθεί και από τη γραμμή εντολών. Παρεμβάλλεται μεταξύ του φυλλομετρητή και του εξυπηρετητή, ούτως ώστε να λαμβάνει ή και να τροποποιεί τα πακέτα που ανταλλάσσονται μεταξύ τους και να αξιολογεί τις απαντήσεις του server. Διαθέτει λειτουργία Safe mode, κατά την οποία δεν εκτελούνται λειτουργίες που μπορεί να βλάψουν το σύστημα. Υπάρχει επίσης marketplace με επιπρόσθετα addons για την αύξηση των λειτουργιών του προγράμματος. Τέλος, έχει ξεκινήσει μια προσπάθεια για να μπορεί να χρησιμοποιηθεί το εργαλείο σε διαδικασίες αυτοματοποίησης, με την ανάπτυξη του Automation Framework addon, όπου μπορεί να ρυθμιστεί το ZAP μέσω YAML file και να εκτελεστεί με την επιλογή -autorun από τη γραμμή εντολών [21].

MHDDOS

Εργαλείο που αναπτύχθηκε για την εκτέλεση επιθέσεων Denial-of-Service και επί του παρόντος μπορεί να το πράξει με 56 διαφορετικούς τρόπους. Οι επιθέσεις του μπορεί να γίνουν στο επίπεδο 4 – Transport, χρησιμοποιώντας για παράδειγμα TCP/UDP flooding ή τη δημιουργία συνδέσεων που παραμένουν για ώρα ανοιχτές, ή μπορούν να πραγματοποιηθούν στο επίπεδο 7 – Application του OSI Model, με τη χρήση HTTP methods ή εκμεταλλευόμενο cookies. Είναι ανοιχτού κώδικα, υλοποιημένο σε Python και μπορεί να ληφθεί και εγκατασταθεί από το GitHub [22].

2.4 Σχετικές Έρευνες

Το 2013, το Αυστραλιανό Τμήμα Άμυνας δημοσίευσε μια μελέτη [22], όπου καταγράφει και κατηγοριοποιεί διάφορα Cyber ranges με βάση την υλοποίησή τους και καταλήγει στο συμπέρασμα ότι οι εξομοιωτές μπορούν να προσφέρουν περιβάλλοντα πλησιέστερα στα πραγματικά, ενώ οι προσομοιωτές με λιγότερο κόστος παρέχουν ευελιξία και περισσότερες δυνατότητες κλιμάκωσης.

Οι Yamin κ.α. προβαίνουν σε συστηματική μελέτη της βιβλιογραφίας και ταξινομούν cyber ranges βασισμένοι στην αρχιτεκτονική, τα σενάρια και άλλες δυνατότητες, όπως είναι το teaming, η επίβλεψη και η διαχείριση. Συμπεραίνουν ότι υπάρχει ανάγκη για μοντελοποίηση της συμπεριφοράς των ομάδων [23].

Η cloud-based πλατφόρμα KYPO, από την ομάδα ασφάλειας του πανεπιστημίου Masaryk της Τσεχίας, παρέχει ευελιξία και επεκτασιμότητα για την προσομοίωση δικτύων και συστημάτων με τη χρήση components. Χρησιμοποιείται για επιμόρφωση φοιτητών και εκπαίδευση επαγγελματιών. Περιλαμβάνει διεπαφή χρήστη για επίβλεψη και ανατροφοδότηση σε πραγματικό χρόνο [7], [24]. Διατίθεται ως λογισμικό ανοιχτού κώδικα και βασίζεται στην πλατφόρμα νέφους OpenStack.

Το DETERLab αναπτύχθηκε από το πανεπιστήμιο της Νότιας Καρολίνας με κύριο στόχο την έρευνα και εκπαίδευση στον τομέα της ασφάλειας πληροφοριακών συστημάτων. Βασίζεται στο λογισμικό Emulab και προσφέρει μεταξύ άλλων τη γένεση κυκλοφορίας στο δίκτυο, εργαλεία επίβλεψης, καθώς και δυνατότητα συνεργασίας με άλλα περιβάλλοντα που τρέχουν με Emulab [25].

Τα δημόσια πανεπιστήμια του Michigan διατηρούν ένα από τα πλέον καθιερωμένα cyber ranges της ακαδημαϊκής κοινότητας, το Merit Network. Μέσα από τα εικονικά περιβάλλοντα Alphaville και Griffinville, δίνεται η δυνατότητα στους εκπαιδευόμενους να εξασκήσουν τις ικανότητές τους στο cybersecurity [26].

Προσφάτως, ο Ευρωπαϊκός Οργανισμός Κυβερνοασφάλειας (ECSO), δημοσίευσε ένα έγγραφο [27] με σκοπό να παρέχει γενικότερη γνώση για τα cyber ranges, τη χρήση και τις τεχνολογίες τους. Αναδεικνύει τις ανάγκες που υπάρχουν, ανάλογα με το σκοπό χρήσης των περιβαλλόντων cyber range και τις επιθυμητές δυνατότητες σε κάθε περίπτωση. Ανάμεσά τους είναι η αυτοματοποίηση των επιθέσεων της κόκκινης ομάδας, η οποία επιτρέπει την εκπαίδευση και προετοιμασία της μπλε ομάδας χωρίς να χρειάζεται η εμπλοκή ομάδας επίθεσης. Η αυτοματοποίηση αυτή μειώνει το κόστος και εξαλείφει την εξάρτηση της έκβασης των ασκήσεων από τις προσωπικές ικανότητες των ατόμων που την αποτελούν. Προς επίτευξη της αυτοματοποίησης αυτής, κύρια τεχνική

που χρησιμοποιείται είναι η δημιουργία γράφων. Ο σωστός σχεδιασμός και η δυναμική γένεσή τους, όμως, παραμένει ανοιχτό ζήτημα.

Οι Ugur κ.α. παρουσιάζουν και συγκρίνουν δύο μεθόδους δημιουργίας γράφων. Η πρώτη προσέγγιση βασίζεται σε κανόνες, ενώ στη δεύτερη μέθοδο ενσωματώνεται μηχανική μάθηση. Χρησιμοποιείται η βάση ευπαθειών NVD, για να καθοριστούν οι προϋποθέσεις και τα αποτελέσματα κάθε πιθανής επίθεσης. Λόγω έλλειψης τυποποιημένης γλώσσας στην περιγραφή ευπαθειών, χρειάστηκε αρκετή χειρωνακτική εργασία για την εξαγωγή κανόνων [28].

Το cyber range CRATE [29], που έχει αναπτυχθεί από το Swedish Defense Research Agency, χρησιμοποιεί το εργαλείο SVED [30] το οποίο παρέχει στους χρήστες τη δυνατότητα δημιουργίας γράφων επίθεσης και της εκτέλεσής τους μέσω τρίτων προγραμμάτων, όπως είναι το OpenVAS και το Metasploit. Δεν παρέχεται η δυνατότητα δυναμικής δημιουργίας γράφων επίθεσης.

Το CALDERA [31] επικεντρώνεται στην εξομοίωση επιθέσεων που γίνονται μετά την επιτυχημένη εισβολή σε ένα σύστημα. Μετά από κάθε επίθεση γίνεται ανατροφοδότηση της βάσης γνώσεων με το αποτέλεσμα, ούτως ώστε να χρησιμοποιηθεί για την επιλογή του επόμενου βήματος που θα ακολουθηθεί από τον εικονικό εισβολέα. Η μεγαλύτερη πρόκληση για τους δημιουργούς είναι η επιλογή του καλύτερου βήματος για την επίτευξη του στόχου του κάθε σεναρίου.

Οι Enoch κ.α. παρουσιάζουν το HARMer framework [32] για την αυτοματοποίηση επιθέσεων και σκοπεύουν στην εύρεση του επόμενου βήματος σε πραγματικό χρόνο, ούτως ώστε να υπολογίζουν και τις δράσεις της μπλε ομάδας, καθώς και τα αποτελέσματα των προηγούμενων βημάτων. Λαμβάνοντας υπόψη τη δυσκολία κλιμάκωσης των γράφων και δέντρων επίθεσης, χρησιμοποιούν το ιεραρχικό μοντέλο δύο επιπέδων HARM [33] για τη λήψη των αποφάσεων τους. Η διαδικασία που ακολουθούν αποτελείται από τέσσερα βήματα. Στο πρώτο βήμα γίνεται συλλογή πληροφοριών για το δίκτυο και τις ευπάθειες μέσω εργαλείων ή απευθείας από τους διαχειριστές. Στο δεύτερο βήμα μέσω του HARM, χρησιμοποιούνται οι πληροφορίες από το προηγούμενο βήμα για να δημιουργηθούν όλα τα πιθανά μονοπάτια επίθεσης και

γίνεται αξιολόγησή τους με βάση μετρικές όπως είναι το ρίσκο, η πιθανότητα επιτυχίας κλπ. Ακολουθεί η φάση της σχεδίασης της επίθεσης με τις πράξεις που πρέπει να ακολουθηθούν και στο τελευταίο βήμα γίνεται η εκτέλεση της επίθεσης με Metasploit. Μετά από κάθε επίθεση υπάρχει ανατροφοδότηση, ούτως ώστε να εξαλειφθούν μονοπάτια που σίγουρα θα είναι ανεπιτυχή με βάση το αποτέλεσμα. Η επιλογή του επόμενου βήματος βασίζεται στις μετρικές που αποδίδονται σε κάθε μονοπάτι, που μπορεί να είναι μεμονωμένες ή σύνθετες και εξαρτάται από τη γνώση που υποθέτουμε ότι κατέχει ο επιτιθέμενος για το δίκτυο που μπορεί να είναι πλήρης ή μερική. Μερικές μετρικές που χρησιμοποιούνται είναι το συντομότερο μονοπάτι, η σοβαρότητα της ευπάθειας βάση του CVSS, το κόστος της επίθεσης και το κέρδος που θα έχει ο επιτιθέμενος από την εκμετάλλευση της ευπάθειας. Δοκιμές του εργαλείου έγιναν σε εικονικά δίκτυα που δημιουργήθηκαν στο Amazon Web Services.

Το Trogdor χαρακτηρίζεται ως ένα mission-centric σύστημα αυτοματοποίησης της κόκκινης ομάδας. Στοχεύει στην υποστήριξη λήψης αποφάσεων για τη θωράκιση του οργανισμού απέναντι στις κυβερνοεπιθέσεις, αποτυπώνοντας την επίδραση των ευπαθειών στους κρίσιμους πόρους του συστήματος. Παράγει και παρουσιάζει οπτικοποιημένα μονοπάτια επίθεσης, προβαίνοντας σε αυτόματη ανάλυση γνωστών ευπαθειών και κρίσιμων κόμβων, χρησιμοποιώντας οντότητες για την περιγραφή του περιβάλλοντος. Με τεχνητή νοημοσύνη γίνεται αξιολόγηση των κινδύνων και τεχνικές visual analytic εφαρμόζονται για την παρουσίαση των αποτελεσμάτων στους εμφανιζόμενους γράφους.

Το Cyber Automated Red Team Tool (CARTT) σχεδιάστηκε ως ένα ελαφρύ και φορητό εργαλείο για την αναγνώριση και εκτίμηση των ευπαθειών σε συστήματα που δεν είναι συνδεδεμένα στο διαδίκτυο. Χρησιμοποιεί το Metasploit Framework και πιο συγκεκριμένα το ενσωματωμένο module του OpenVAS για την αναζήτηση ευπαθειών. Παρέχει ένα απλό γραφικό περιβάλλον γραμμένο σε Python και οι λειτουργίες του είναι το host discovery με ping scan, OS discovery με NMAP ή p0f και vulnerability scanning. Τα αποτελέσματα εξάγονται σε αναφορά [34].

Το SPIDER cyber range εξειδικεύεται στον τομέα των τηλεπικοινωνιών και την τεχνολογία 5G. Απευθύνεται σε ειδικευμένα, αλλά και ανειδίκευτα άτομα, εκπαιδεύοντας

τα μέσω παιχνιδοποίησης είτε για ενημέρωση και διεύρυνση των γνώσεων τους στις τεχνολογίες 5G, είτε για την ανάπτυξη επαγγελματικών δεξιοτήτων ασφάλειας. Περιλαμβάνει τη μονάδα Continuous Risk Assessment Engine, που υπολογίζει την έκθεση της εξομοιωμένης υποδομής σε δεδομένο σενάριο, σκοπεύοντας στον επηρεασμό στη λήψη αποφάσεων. Διαθέτει έτοιμα σενάρια και επιτρέπει τη χρήση εργαλείων που δεν προϋπάρχουν στην πλατφόρμα. Έχει, επίσης, τη δυνατότητα αυτόματης παραγωγής αμυντικών και επιθετικών δραστηριοτήτων με τη μορφή κυκλοφορίας δικτύου μέσω τεχνητής νοημοσύνης και μηχανικής μάθησης. Στο dashboard μπορεί κανείς να δει την πρόοδο των ασκήσεων, τη βαθμολογία και τα ρίσκα, ενώ υπάρχουν και περιορισμένα views που διατίθενται στα μέλη των διαγωνιζόμενων ομάδων [11] [35].

Εμπορικά cyber ranges με διαφοροποιημένες δυνατότητες χρησιμοποιούνται κυρίως για κατάρτιση επαγγελματιών κυβερνοασφάλειας ή από ιδιωτικές εταιρίες για έλεγχο των συστημάτων τους. Μερικά από τα γνωρίσματά τους είναι η συνεχής ενημέρωση για νέες ευπάθειες, προσομοίωση πραγματικής κίνησης στο δίκτυο, αυτοματοποίηση επιθέσεων της κόκκινης ομάδας, αλλά και επίβλεψη μέσω οπτικοποίησης σε πραγματικό χρόνο. Μια συγκριτική μελέτη των παροχών υφιστάμενων cyber ranges γίνεται από τον Chaskos [36].

Τέλος, η έλλειψη πόρων, καθώς και η μεγάλη διαφοροποίηση στις δυνατότητες των υπαρχόντων cyber ranges δημιουργεί την ανάγκη συνεργασίας μεταξύ τους. Μια τέτοια συνθήκη ευνοεί τη γρηγορότερη ανάπτυξη του τομέα και τη ρεαλιστικότερη προσέγγιση των συνθηκών του πραγματικού κόσμου. Προς αυτή την κατεύθυνση κινούνται τα Cyber Ranges Federation Project του Ευρωπαϊκού Οργανισμού Άμυνας (EDA) για τη συνεργασία των εθνικών cyber ranges ευρωπαϊκών χωρών [37] και το, επίσης ευρωπαϊκό, ECHO Project [38].

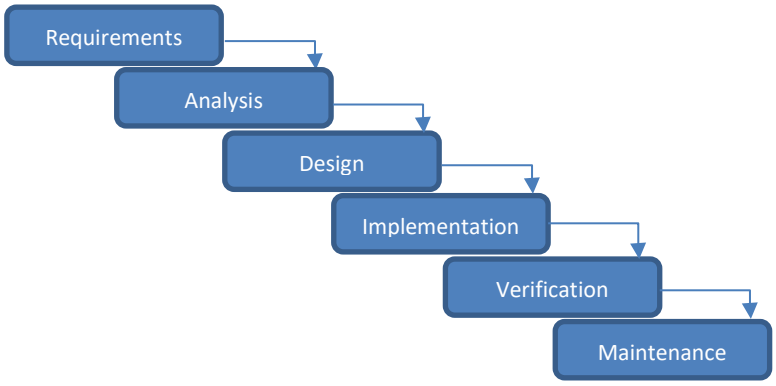
Συγκεκριμένα, στα πλαίσια του ECHO έχει αναπτυχθεί το ECHO Federated Cyber Range, ένα marketplace που φιλοδοξεί να λειτουργήσει ως ενδιάμεσος για τη δημιουργία σεναρίων που θα χρησιμοποιούν ταυτόχρονα cyber ranges από διαφορετικούς παρόχους. Μέσω μιας διαδικτυακής πύλης προσφέρει προκαθορισμένα σενάρια ή επιτρέπει τη διαπραγμάτευση για την εύρεση των κατάλληλων λύσεων για τις αναζητούμενες υπηρεσίες [39].

Κεφάλαιο 3
3 Μεθοδολογία

Για την ανάπτυξη λογισμικού υπάρχουν διάφορες μεθοδολογίες, οι οποίες προτείνουν συγκεκριμένα στάδια που πρέπει να ακολουθηθούν για καλύτερο σχεδιασμό και αποτελεσματικότερη διαχείριση της διαδικασίας. Οι μεθοδολογίες αυτές ορίζουν στην ουσία τον κύκλο ζωής του λογισμικού κι η επιλογή της καταλληλότερης εξαρτάται από διάφορες παραμέτρους, όπως είναι το είδος του έργου και το μέγεθός του, η ομάδα που θα συμμετέχει, το κόστος και ο διαθέσιμος χρόνος.

3.1 Μοντέλο Καταρράκτη

Το μοντέλο Καταρράκτη (Waterfall) αποτελεί την παλαιότερη μεθοδολογία και αποδίδεται στον Winston W. Royce για τη δημοσίευσή του το 1970 [40]. Περιλαμβάνει τα στάδια του καθορισμού απαιτήσεων, ανάλυσης, σχεδιασμού του συστήματος, υλοποίησης, ελέγχου και συντήρησης. Τα στάδια εκτελούνται σειριακά, προχωρώντας στο επόμενο μόνο αφότου ολοκληρωθεί το προηγούμενο βήμα και έχει τεκμηριωθεί πλήρως σε έγγραφα. Ο άκαμπτος αυτός διαμερισμός το καθιστά ακατάλληλο για έργα στα οποία αναμένονται τροποποιήσεις. Επομένως, θα μπορούσε να χρησιμοποιηθεί επιτυχώς σε μικρά έργα με πολύ καλά καθορισμένες απαιτήσεις.



3.2 Ευέλικτες Μέθοδοι

Οι Ευέλικτες Μέθοδοι (Agile) στοχεύουν στην καλύτερη διαχείριση των τροποποιήσεων που συνήθως προκύπτουν κατά την ανάπτυξη ενός έργου και στη γρηγορότερη παράδοση κατά μέρη, αντί ολόκληρου του έργου στη λήξη της προθεσμίας. Βασίζονται σε σύντομους επαναληπτικούς και αυξητικούς κύκλους (iterative and incremental), στους οποίους παραδίδονται τμήματα του λογισμικού, με αποτέλεσμα την άμεση ανατροφοδότηση από τους χρήστες και ευκολότερη προσαρμογή.
Μερικές από τις πιο γνωστές Ευέλικτες Μεθόδους είναι οι SCRUM, Lean, Feature Driven Development (FDD), eXtreme Programming (XP) και Adaptive Software Development (ASD). Παρατίθενται πιο κάτω δύο από αυτές.

3.2.1 SCRUM

Η μεθοδολογία SCRUM θεωρεί τη διαδικασία ανάπτυξης συστημάτων απρόβλεπτη και περίπλοκη και αποδέχεται ότι ένα έργο δεν μπορεί να οριστεί πλήρως από την αρχή, με τις τροποποιήσεις στο πλάνο να είναι αναπόφευκτες [41]. Το πρώτο στάδιο περιλαμβάνει την προσθήκη υπο-εργασιών στο Backlog, από το οποίο αργότερα θα επιλεγούν αυτές με τη μεγαλύτερη προτεραιότητα για υλοποίηση στο επόμενο στάδιο. Ακολουθεί η επαναληπτική φάση των Sprints που διαρκούν συνήθως 2-4 εβδομάδες, κατά την οποία γίνεται ανάλυση, σχεδίαση, ανάπτυξη, έλεγχος και τεκμηρίωση των επιλεγμένων εργασιών. Το Sprint κλείνει με αξιολόγηση της προόδου και την προσθήκη νέων εργασιών στο Backlog, αν χρειάζεται. Όταν το έργο κριθεί έτοιμο προς παράδοση, γίνεται μετάβαση στο τελικό στάδιο, όπου εκτελείται ο έλεγχος ενσωμάτωσης, δημιουργείται ο οδηγός χρήστη, γίνεται εκπαίδευση χρηστών και ό,τι άλλο χρειάζεται για την καινούρια έκδοση του λογισμικού.

Το SCRUM είναι από τις πλέον διαδεδομένες μεθοδολογίες ανάπτυξης λογισμικού και για αυτό το λόγο υπάρχουν αρκετά εργαλεία που στοχεύουν στη βοήθεια των ομάδων για την υλοποίησή της. Μερικά παραδείγματα είναι τα Azure Boards [42], JIRA [43] και Trello [44].

3.2.2 Feature Driven Development

Αποτελείται από πέντε στάδια, με το πρώτο να είναι η ανάπτυξη ενός γενικού μοντέλου στην αρχή του έργου. Με βάση αυτό, στην επόμενη φάση δημιουργείται μια λίστα χαρακτηριστικών (features). Το κάθε χαρακτηριστικό πρέπει να είναι αρκετά μικρό σε μέγεθος για να μπορεί να υλοποιηθεί σε διάστημα το πολύ δύο εβδομάδων. Στο τρίτο στάδιο ετοιμάζεται το πλάνο ανάπτυξης για επιλεγμένα χαρακτηριστικά από τη λίστα με βάση τις προτεραιότητες. Τα δύο τελευταία στάδια είναι επαναληπτικά και αφορούν το σχεδιασμό και την κατασκευή των επιλεγμένων χαρακτηριστικών στη διάρκεια των δύο εβδομάδων. Αποτελούν το ευέλικτο μέρος του μοντέλου, όπου γίνεται η προσαρμογή σε τυχόν αλλαγές και στο τέλος η παράδοση μιας νέας έκδοσης του συστήματος.

3.2.3 Extreme Programming (XP)

Αναπτύχθηκε από τον Kent Beck το 1996 και δημοσιεύθηκε στο βιβλίο του "Extreme Programming Explained" το 1999, ενώ η δεύτερη έκδοση κυκλοφόρησε το 2004. Βασίζεται στις πέντε αρχές:

1. Επικοινωνία: Μεταξύ των μελών της ομάδας, αλλά και με τους πελάτες.
2. Απλότητα: Σχεδιασμός του συστήματος όσο πιο απλά και ξεκάθαρα γίνεται. Αποφυγή σχεδιασμού για μελλοντικές λειτουργίες που δεν έχουν ζητηθεί.
3. Ανατροφοδότηση: Μέσω ελέγχου των μονάδων που αναπτύσσονται από την πρώτη μέρα, με χρήση test-driven development. Με γρήγορη παράδοση λειτουργιών στους πελάτες, για άμεση ανατροφοδότηση και καθορισμό πιθανών αλλαγών.
4. Θάρρος: Στη λήψη αποφάσεων για αλλαγή στρατηγικής όταν κάτι δεν δουλεύει, στην αποδοχή της ανατροφοδότησης όποια κι αν είναι, στο να λύσεις θέματα που επηρεάζουν την ομάδα και σε οτιδήποτε άλλο προκύψει.
5. Σεβασμός: Μεταξύ όλων των εμπλεκομένων.

Απευθύνεται σε μικρές ομάδες και δίνει ιδιαίτερη βαρύτητα στην ομαδική εργασία, καθώς και τον ενδελεχή έλεγχο των μονάδων και των λειτουργιών. Σκοπός του είναι η παράδοση του απαιτούμενου λογισμικού όπως πρέπει να είναι και στον καθορισμένο χρόνο. Αρχικά δημιουργούνται User stories με τη λειτουργικότητα που απαιτεί ο πελάτης και την εκτίμηση του χρόνου που υπολογίζουν οι προγραμματιστές ότι χρειάζονται για να υλοποιηθούν. Στη συνέχεια δημιουργείται το release plan με βάση τις προτεραιότητες του πελάτη και σχεδιάζονται τα iterations 1-3 εβδομάδων, όπου καθορίζεται τι θα γίνει σε αυτό το χρόνο.

3.3 Μεθοδολογία Παρούσας Διατριβής

Στην παρούσα μεταπτυχιακή διατριβή δεν ακολουθήθηκε πιστά κάποια από τις μεθοδολογίες, δεδομένου ότι εργάστηκε μόνο ένα άτομο με την επίβλεψη της υπεύθυνης καθηγήτριας και ο χρόνος εργασίας ήταν ακαθόριστος. Παρόλα αυτά η διαδικασία προσομοιάζει με τις ευέλικτες μεθόδους, έχοντας αρχικά διαχωρίσει το έργο σε τρεις μεγάλες υπο-εργασίες/χαρακτηριστικά, των οποίων ο λεπτομερής σχεδιασμός και υλοποίηση έγιναν σε φάσεις. Για το κάθε χαρακτηριστικό χρειάστηκε περαιτέρω διάσπαση σε μικρότερα, πιο λεπτομερή χαρακτηριστικά, ο καθορισμός των οποίων γινόταν πριν την ενασχόληση με τη συγκεκριμένη εργασία. Τα τρία βασικά χαρακτηριστικά του έργου που αναπτύχθηκε για την παρούσα μεταπτυχιακή διατριβή είναι:

1. Γραφικό περιβάλλον για παρακολούθηση των σεναρίων
2. Εύρεση και προγραμματιστική εκτέλεση red team scripts.
3. Αυτοματοποίηση των scripts.

Για το γραφικό περιβάλλον χρειάστηκε η δημιουργία πολλαπλών οθονών με βάση δοθέν πρότυπο, σύνδεση με τον εξυπηρετητή και τη βάση δεδομένων, προσθήκη της δυνατότητας προβολής δεδομένων σε πραγματικό χρόνο.

Για τα scripts προηγήθηκε έρευνα για εύρεση στη βιβλιογραφία και το διαδίκτυο, στη συνέχεια σχεδιάστηκε το σύστημα που θα υποστήριζε την εκτέλεσή τους μέσα από το υπό ανάπτυξη πρόγραμμα κι έπειτα ακολούθησε η υλοποίηση του συστήματος αυτού. Εν τέλει, έγινε προσθήκη λειτουργιών στο γραφικό περιβάλλον για τη δυνατότητα εντολής εκτέλεσης των script από τους χρήστες.

Για την αυτοματοποίηση σχεδιάστηκε και δημιουργήθηκε ένα σύστημα που να επιτρέπει την εκτέλεση των scripts είτε σε καθορισμένο χρόνο είτε ως αντίδραση σε κάποιο συμβάν. Τέλος, προστέθηκε η δυνατότητα προβολής των προγραμματισμένων λειτουργιών στο γραφικό περιβάλλον.

Κεφάλαιο 4
4 Υλοποίηση

Η υλοποίηση του συστήματός μας αφορά την πλατφόρμα Cyber Range του Ανοικτού Πανεπιστημίου Κύπρου, επομένως έχει δημιουργηθεί με γνώμονα τη συμβατότητα με το εν λόγω περιβάλλον και τη χρήση δεδομένων που συλλέγονται σε αυτή.

4.1 Περιβάλλον

Το ManageIQ είναι πλατφόρμα ανοικτού κώδικα από τη Red Hat, που χρησιμοποιείται για τη διαχείριση υβριδικών περιβαλλόντων πληροφορικής, που μπορεί να αποτελούνται από τεχνολογίες όπως εικονικά μηχανήματα, δημόσια υπολογιστικά νέφη (public clouds) ή containers [45]. Το Cyber Range του Ανοικτού Πανεπιστήμιου Κύπρου το χρησιμοποιεί σε συνδυασμό με τον πάροχο (provider) oVirt -επίσης από τη Red Hat- για τη δημιουργία εικονικών περιβαλλόντων με βάση τα σενάρια των ασκήσεων.

Στα μηχανήματα (hosts) που συμμετέχουν στα σενάρια εγκαθίστανται πράκτορες (agents) Wazuh, που αποσκοπούν στον εντοπισμό συμβάντων ασφαλείας. Αντίστοιχα, στις συσκευές δικτύου εγκαθίστανται Suricata agents, που παρακολουθούν τα πακέτα που διακινούνται στο δίκτυο και μέσω ανάλυσης εντοπίζονται ύποπτες ή κακόβουλες κινήσεις. Όλα τα συμβάντα καταγράφονται από το σύστημα του Cyber Range και χρησιμοποιούνται για την ανάλυση της πορείας των ασκήσεων, καθώς και τη βαθμολογία των συμμετεχόντων.

Η εφαρμογή που υλοποιήσαμε αναμένεται να λειτουργεί σε ένα μηχάνημα με λειτουργικό σύστημα Linux που θα βρίσκεται σε επικοινωνία με την πλατφόρμα ManageIQ και τη βάση καταγραφής δεδομένων από το εικονικό περιβάλλον, ούτως ώστε να αντλεί πληροφορίες από αυτό. Πρόσβαση θα έχει ο διαχειριστής των ασκήσεων, ο οποίος θα μπορεί να παρακολουθεί σε γραφικό περιβάλλον συμβάντα που συλλέγονται από τα

τρέχοντα σενάρια, καθώς και να εκτελεί scripts επιθέσεων προς άλλα μηχανήματα στο δίκτυο.

4.2 Διαδικτυακή Εφαρμογή

Ο εξυπηρετητής αναπτύχθηκε στη γλώσσα προγραμματισμού Python, καθώς είναι αυτή που χρησιμοποιείται σε άλλα προγράμματα της ομάδας του Cyber Range του ΑΠΚ και επομένως θα είναι πιο εύκολη η συντήρηση και πιθανή μετεξέλιξή της. Η έκδοση της Python που χρησιμοποιήθηκε είναι η 3.8.10. Χρησιμοποιήθηκε επίσης το framework Flask, ένα από τα δημοφιλέστερα frameworks για διαδικτυακές εφαρμογές σε Python.

Το γραφικό περιβάλλον υλοποιήθηκε με τη βιβλιοθήκη React, επίσης ιδιαίτερα δημοφιλής για τη δημιουργία διεπαφών χρήστη και το framework Material UI, που προσφέρει μια ευρεία γκάμα έτοιμων συστατικών (components).

4.2.1 Δημιουργία Εφαρμογής

Η ανάπτυξη της εφαρμογής έγινε στο Windows Subsystem for Linux, όπου χρησιμοποιήθηκε η διανομή Ubuntu.

Προαπαιτούμενα:

1. Node.js & npm

   Η Node.js είναι Javascript runtime environment για τη δημιουργία και εκτέλεση διαδικτυακών εφαρμογών με δυναμικό περιεχόμενο. Το npm χρησιμοποιείται για να έχουμε πρόσβαση στο npm Software Registry, όπου διαμοιράζονται βιβλιοθήκες με Javascript κώδικα.

   Εγκατάσταση
   $ sudo apt-get install -y nodejs

   Επαλήθευση
   $ node -v
   $ npm -version

2. Yarn

   Διαχειριστής πακέτων (package manager) για τη Node.js, που βοηθά στη διαχείριση των εξαρτήσεων. Χρησιμοποιείται για εγκατάσταση, αφαίρεση και ενημέρωση των εξωτερικών βιβλιοθηκών που χρησιμοποιούνται από την εφαρμογή.

   Εγκατάσταση

   $ npm install –global yarn

   Επαλήθευση

   $ yarn --version

3. Python & pip

   Η γλώσσα προγραμματισμού Python είναι προεγκατεστημένη στα περισσότερα λειτουργικά συστήματα. Το pip είναι ο διαχειριστής πακέτων για την Python.

   Ενημέρωση εγκατάστασης python

   $ sudo apt update

   $ sudo apt -y upgrade

   $ python3 -V

   Εγκατάσταση pip

   $ sudo apt install -y python3-pip

   Εγκατάσταση επιπλέον εργαλείων για ανάπτυξη εφαρμογών σε Python:

   $ sudo apt install -y build-essential libssl-dev libffi-dev python3-dev

4. MySQL database development files

   Για να μπορούμε να χρησιμοποιήσουμε το MySQL σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) που βασίζεται στην SQL, χρειαζόμαστε το metapackage default-libmysqlclient-dev.

   Εγκατάσταση

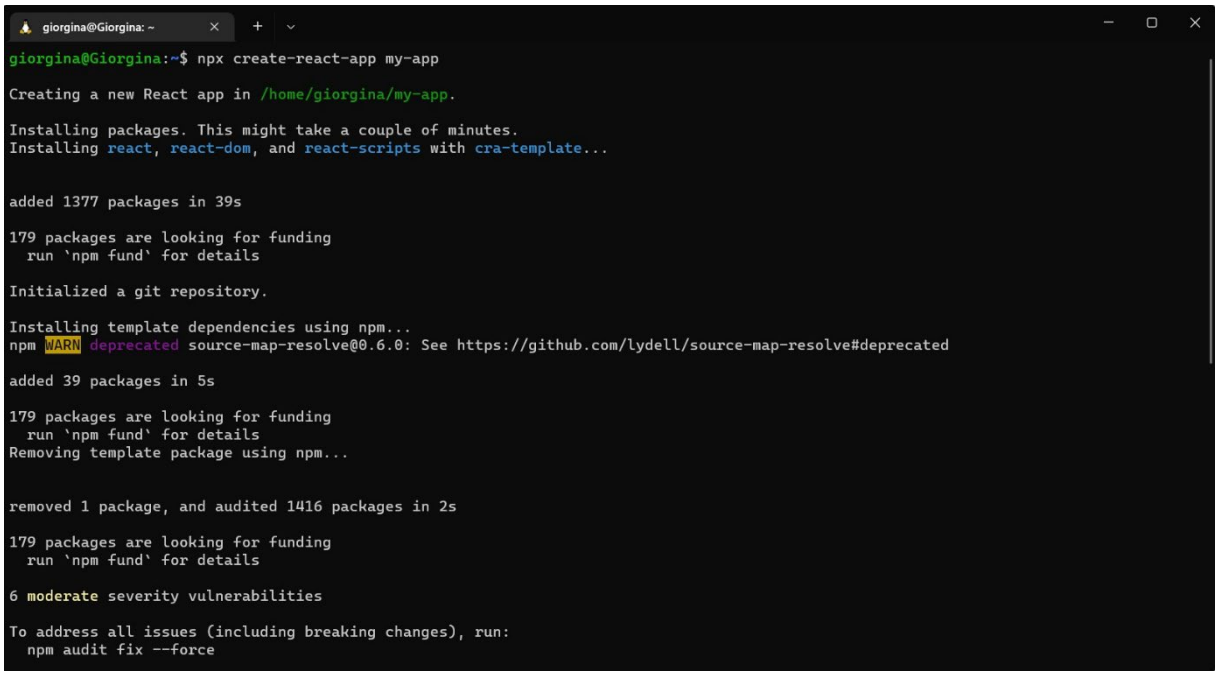
$ sudo apt-get install default-libmysqlclient-dev

Η βάση δεδομένων πρέπει να είναι σε λειτουργία κατά την εκτέλεση του προγράμματος μας, για να μπορούμε να λαμβάνουμε δεδομένα. Μπορεί να τεθεί να ενεργοποιείται αυτόματα κατά την εκκίνηση του συστήματος ή αλλιώς την ξεκινούμε χειροκίνητα με την εντολή

$ sudo service mysql start

Μετά την εγκατάσταση των απαραίτητων πακέτων στο λειτουργικό σύστημα, δημιουργούμε ένα καινούριο project σύμφωνα με τις οδηγίες που βρίσκονται στον επίσημο ιστότοπο της React. Χρησιμοποιούμε την εντολή

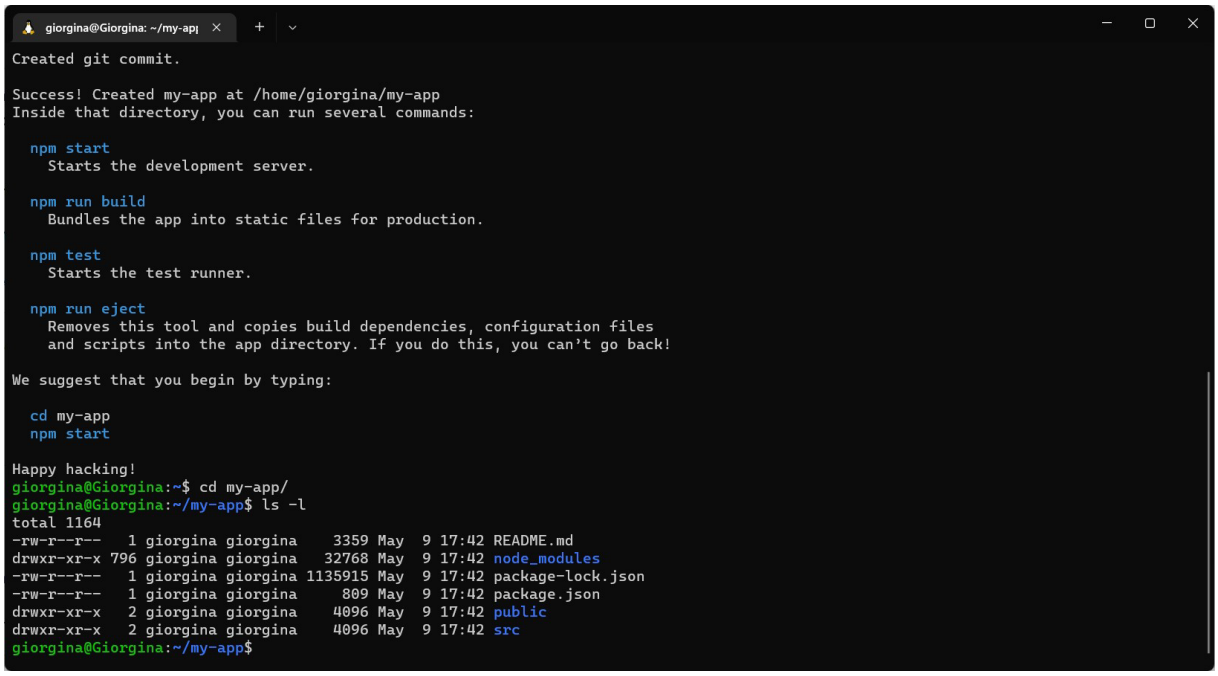
$ npx create-react-app my-app



Σχήμα 1 Δημιουργία React εφαρμογής

Βλέπουμε τα αρχεία που δημιουργήθηκαν για το front-end μέρος της εφαρμογής

$ cd my-app
$ ls -l



Σχήμα 2 Εμφάνιση δομής React εφαρμογής

Στη συνέχεια δημιουργούμε έναν καινούριο φάκελο μέσα στο φάκελο my-app. Σε αυτόν θα κρατούμε τον κώδικα του back-end που θα είναι γραμμένος σε Python.

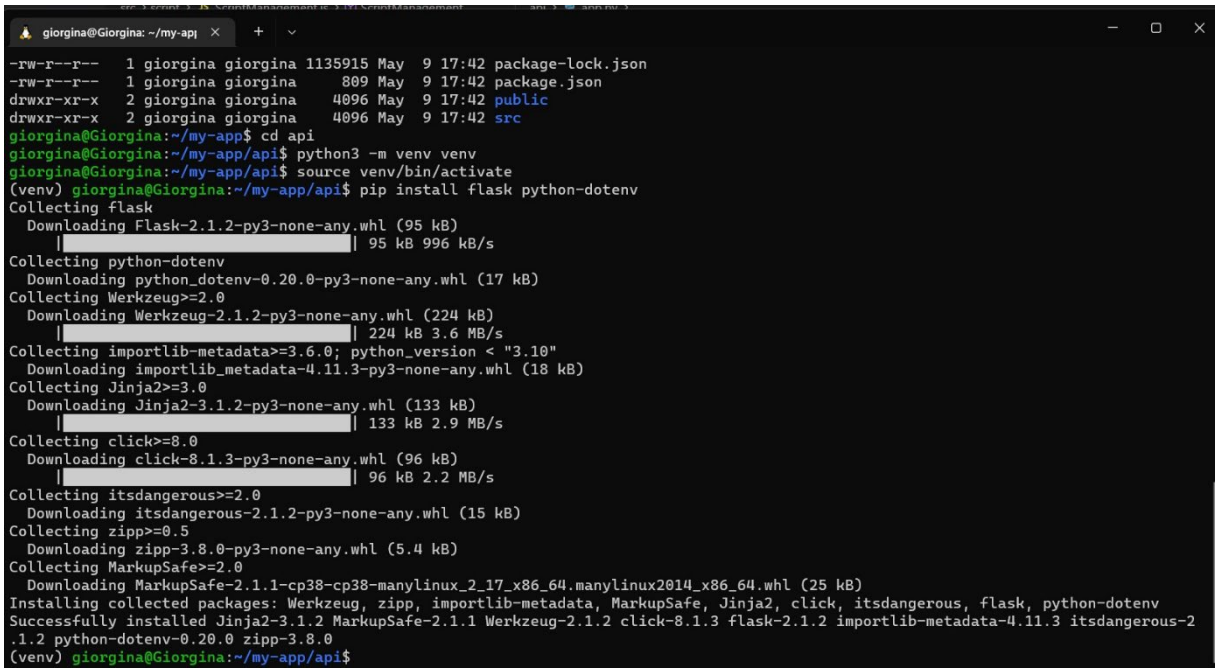
$ mkdir api

Δημιουργούμε ένα εικονικό περιβάλλον, όπου θα αποθηκεύονται οι βιβλιοθήκες Python που θα χρησιμοποιήσουμε, ούτως ώστε η εφαρμογή να είναι ανεξάρτητη από τυχόν άλλες εφαρμογές και να μην υπάρξουν διενέξεις με τις εκδόσεις.

$ cd api
$ python3 -m venv venv

Ενεργοποιούμε το εικονικό περιβάλλον και εγκαθιστούμε σε αυτό το framework Flask και το εργαλείο python-dotenv, το οποίο θα χρησιμοποιήσουμε για τη ρύθμιση μεταβλητών περιβάλλοντος.

$ source venv/bin/activate
(venv) $ pip install flask python-dotenv



Σχήμα 3 Δημιουργία Flask backend

Στο φάκελο api δημιουργούμε ένα απλό αρχείο main.py για να δοκιμάσουμε το setup, με το εξής περιεχόμενο

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return "Hello world!"
```

Το Flask φορτώνει την εφαρμογή με βάση την τιμή της μεταβλητής περιβάλλοντος FLASK_APP. Για να μη χρειάζεται η ρύθμιση της μεταβλητής κάθε φορά που «τρέχουμε» την εφαρμογή, δημιουργούμε ένα αρχείο .flaskenv, το οποίο θα διαβάζεται με τη βοήθεια του python-dotenv που εγκαταστήσαμε προηγουμένως στο εικονικό περιβάλλον. Προσθέτουμε, επίσης τη μεταβλητή FLASK_ENV και τη θέτουμε σε λειτουργία development για να ενεργοποιείται και η λειτουργία εντοπισμού σφαλμάτων (debugging).

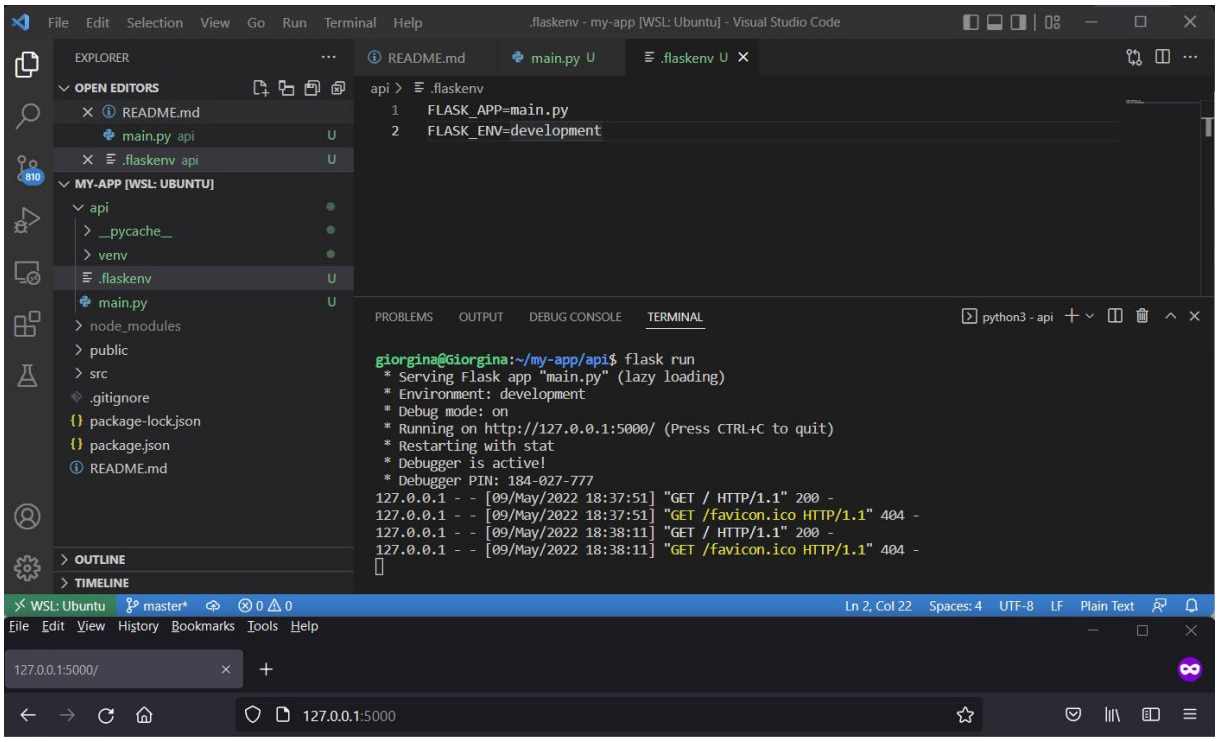
FLASK_APP=main.py

FLASK_ENV=development

Ελέγχουμε τον εξυπηρετητή ξεκινώντας τον με την παρακάτω εντολή

$ flask run

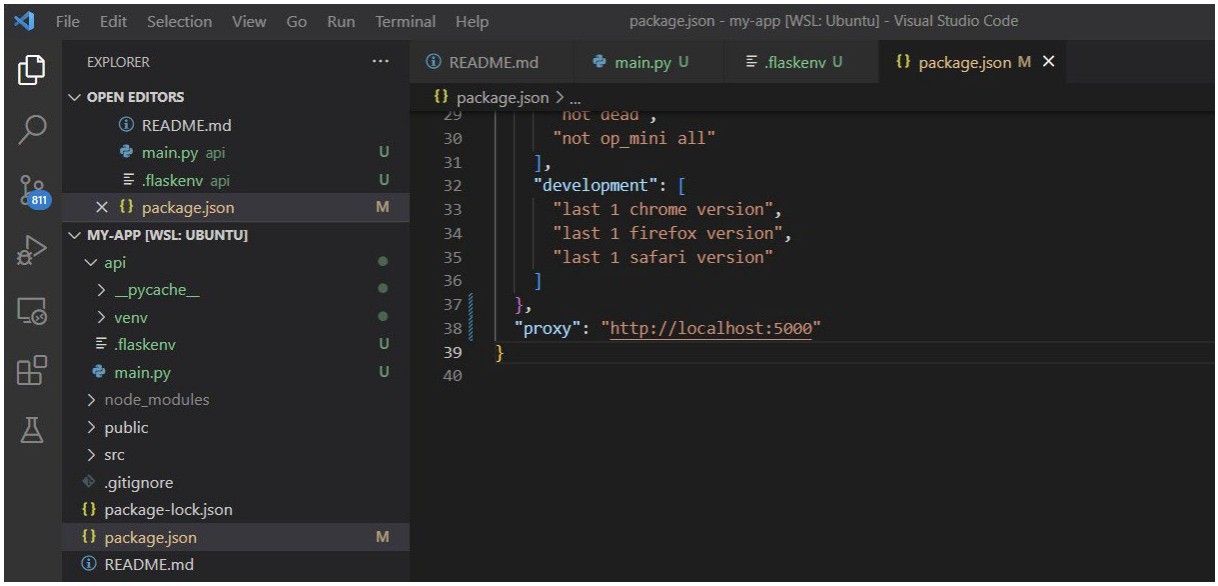
Σε ένα φυλλομετρητή (browser) εισάγουμε τη διεύθυνση localhost:5000 και βλέπουμε το μήνυμα “Hello world!”



Σχήμα 4 Ρύθμιση Flask environment και ενεργοποίηση εξυπηρετητή

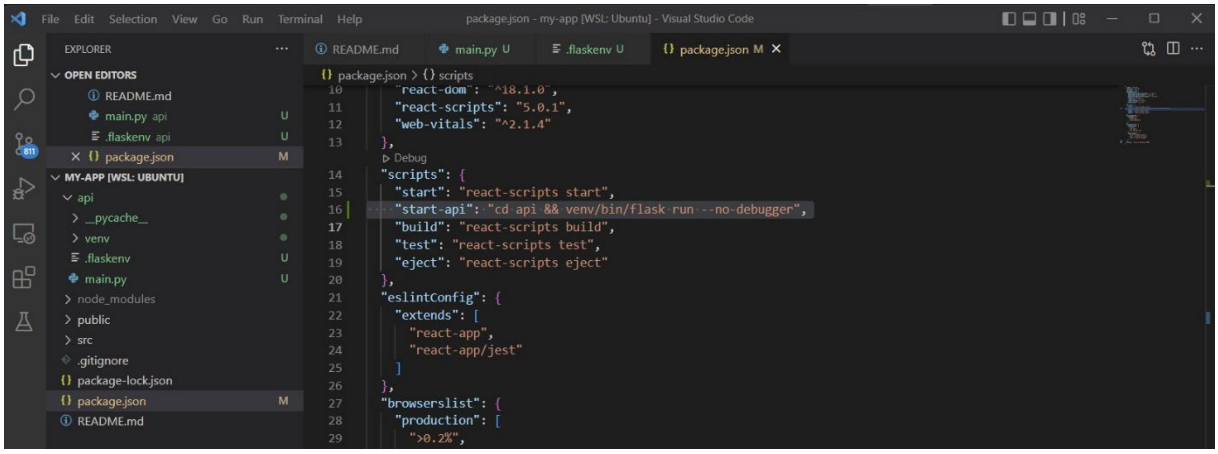
Συνεχίζουμε με μερικές μεταρρυθμίσεις που χρειάζονται για να την εύρυθμη λειτουργία της React και του Flask και την ενσωμάτωση του front-end με το back-end. Η React τρέχει ένα δικό της server στη θύρα 3000, ενώ όπως είδαμε ο Flask server τρέχει στη θύρα 5000. Στο αρχείο package.json προσθέτουμε τη διεύθυνση του Flask server ως proxy, ούτως

ώστε όσα requests δε γνωρίζει πώς να τα εξυπηρετήσει ο React server, να προωθούνται στο Flask server.



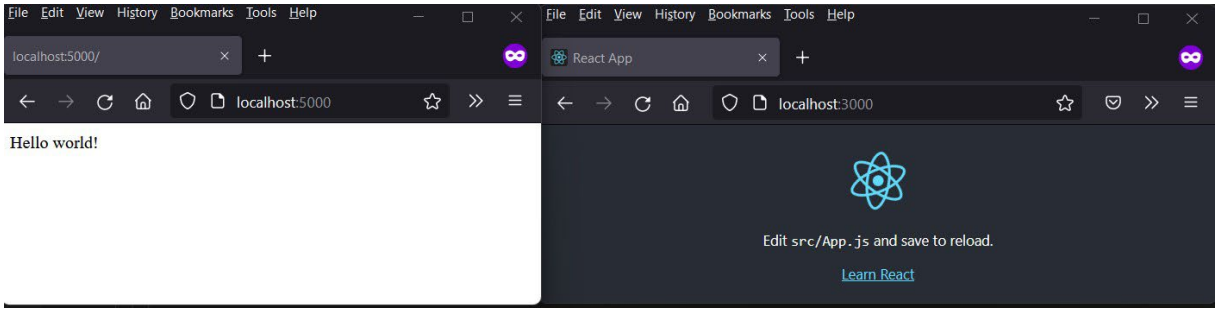
Σχήμα 5 Ρύθμιση React server για ανακατεύθυνση σε Flask

Τέλος προσθέτουμε την εντολή start-api στο package.json, στην ενότητα scripts, για να διευκολύνουμε την εκκίνηση του Flask server. Η εντολή αυτή οδηγεί στον υπο-φάκελο api που βρίσκεται ο εξυπηρετητής και τον ενεργοποιεί, χωρίς τη λειτουργία εντοπισμού σφαλμάτων, η οποία αναφέρεται στο φυλλομετρητή και δε χρειάζεται στην προκειμένη, εφόσον αυτός ο server δε σχετίζεται με το γραφικό περιβάλλον.



Σχήμα 6 Δημιουργία εντολής συντόμευσης για την ενεργοποίηση του Flask server

Με το εργαλείο yarn ενεργοποιούμε και τους δύο εξυπηρετητές με τις εντολές yarn start και yarn start-api και βλέπουμε το αποτέλεσμα, όπως φαίνεται παρακάτω



Σχήμα 7 Εμφάνιση του αποτελέσματος για την ενεργοποίηση των Flask και React servers

Όλα τα αρχεία κώδικα που αναφέρονται στο back-end server θα βρίσκονται στο φάκελο api, ενώ αυτά που αφορούν το front-end θα τοποθετούνται στο φάκελο src. Η επικοινωνία μεταξύ των δύο θα γίνεται μέσω ασύγχρονων HTTP requests (ajax calls) από το front-end στο back-end.

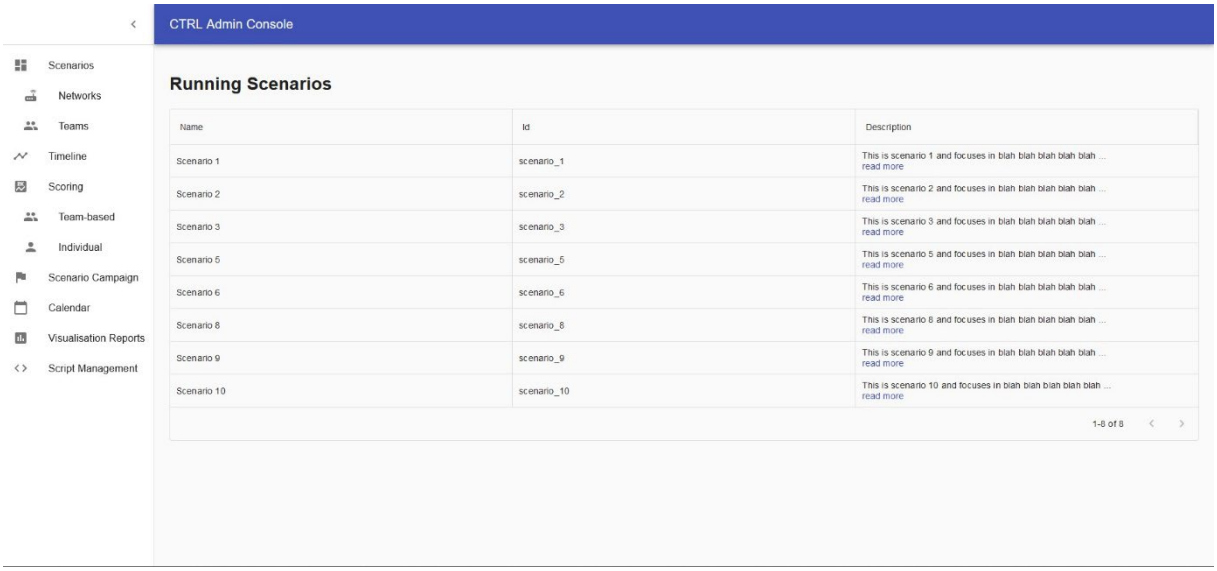
4.2.2 Γραφικό Περιβάλλον

Το πρότυπο του γραφικού περιβάλλοντος της εφαρμογής δόθηκε από την επιβλέπουσα καθηγήτρια ως καθοδηγητικό έγγραφο και στη συνέχεια προσαρμόστηκε κατά την ανάπτυξη, με βάση τις ανάγκες και τις δυνατότητες.

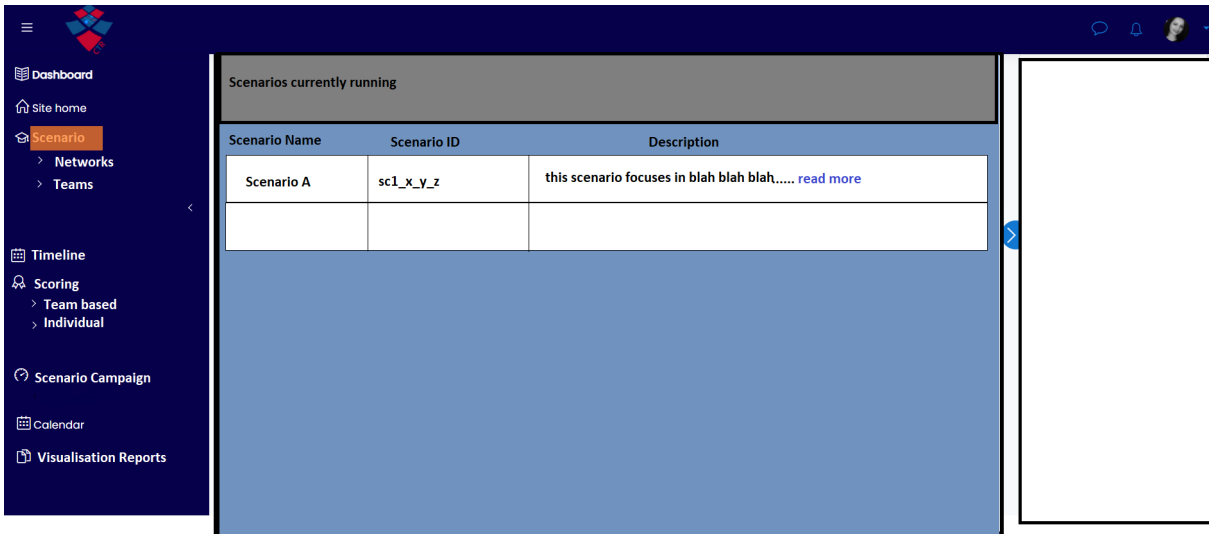
Περιλαμβάνει συνολικά 11 οθόνες και υπο-οθόνες.

1. Scenarios

   Είναι η κεντρική σελίδα της εφαρμογής. Παρουσιάζει τα τρέχοντα σενάρια στην πλατφόρμα, σε μορφή πίνακα. Για κάθε σενάριο εμφανίζονται το όνομα, η ταυτότητα του σεναρίου και μια περιγραφή. Ένας σύνδεσμος βρίσκεται στο πεδίο της περιγραφής που οδηγεί σε άλλο ιστότοπο για περισσότερες πληροφορίες σχετικά με το σενάριο. Ο ιστότοπος αυτός αναμένεται να είναι η πλατφόρμα Moodle.



Σχήμα 8 Υλοποίηση οθόνης Scenarios



Σχήμα 9 Πρότυπο οθόνης Scenarios

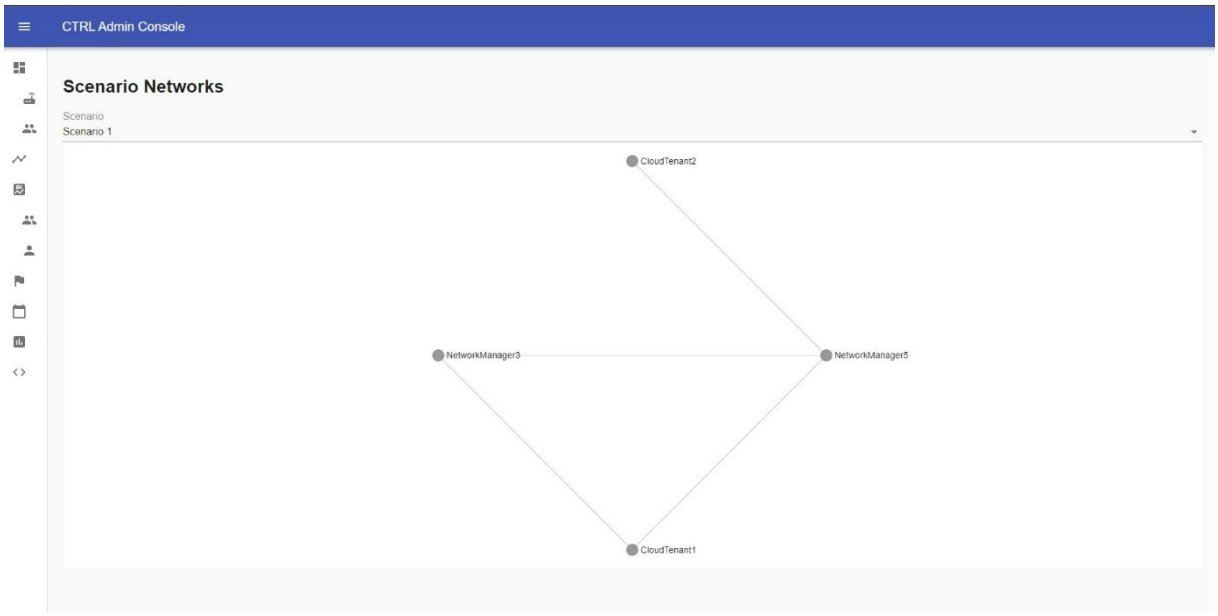
a. Networks

Σε αυτή την οθόνη ο χρήστης επιλέγει ένα σενάριο από τη λίστα και του παρουσιάζεται το γράφημα της τοπολογίας του δικτύου με τα στοιχεία που παρέχει η πλατφόρμα ManageIQ. Επιλέγοντας ένα στοιχείο της τοπολογίας, εμφανίζονται περισσότερες σχετικές πληροφορίες:

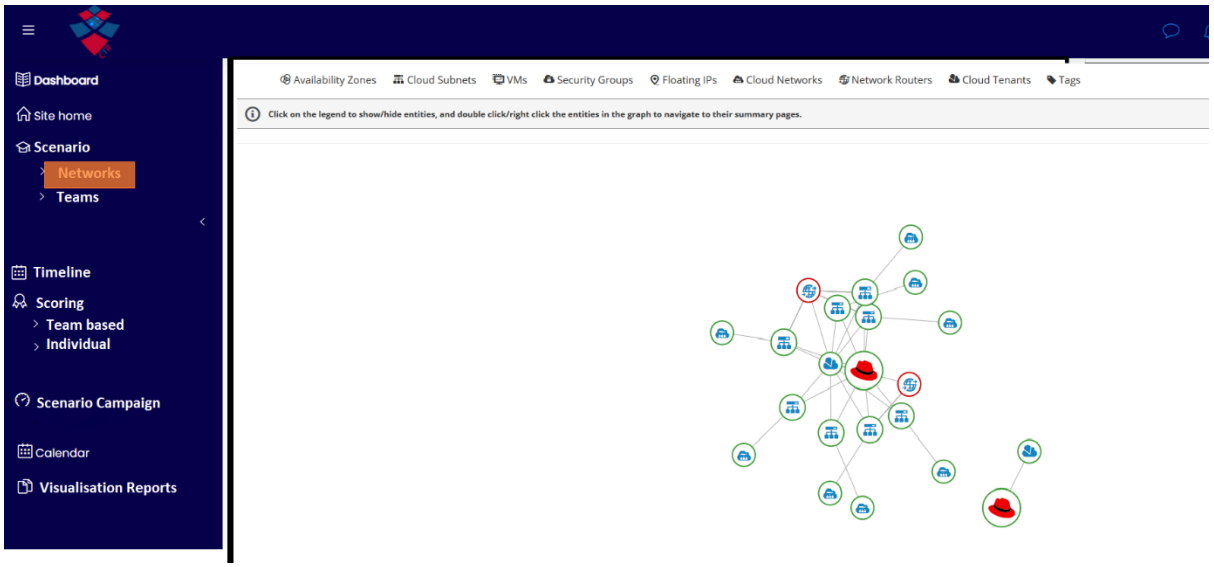
i. Id: Η ταυτότητα του μηχανήματος στο ManageIQ.
ii. Kind: Το είδος του στοιχείου.

iii. Status: Η κατάσταση του στοιχείου.

iv. Relationships: Συνδέσεις με άλλα στοιχεία του δικτύου.



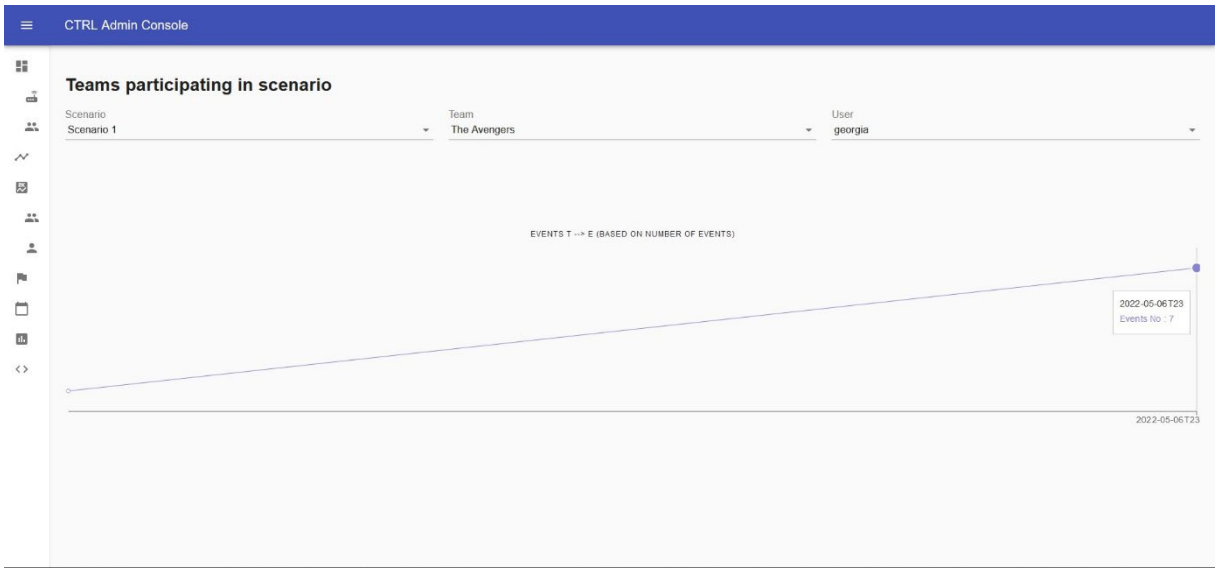
Σχήμα 10 Υλοποίηση οθόνης Scenario Networks



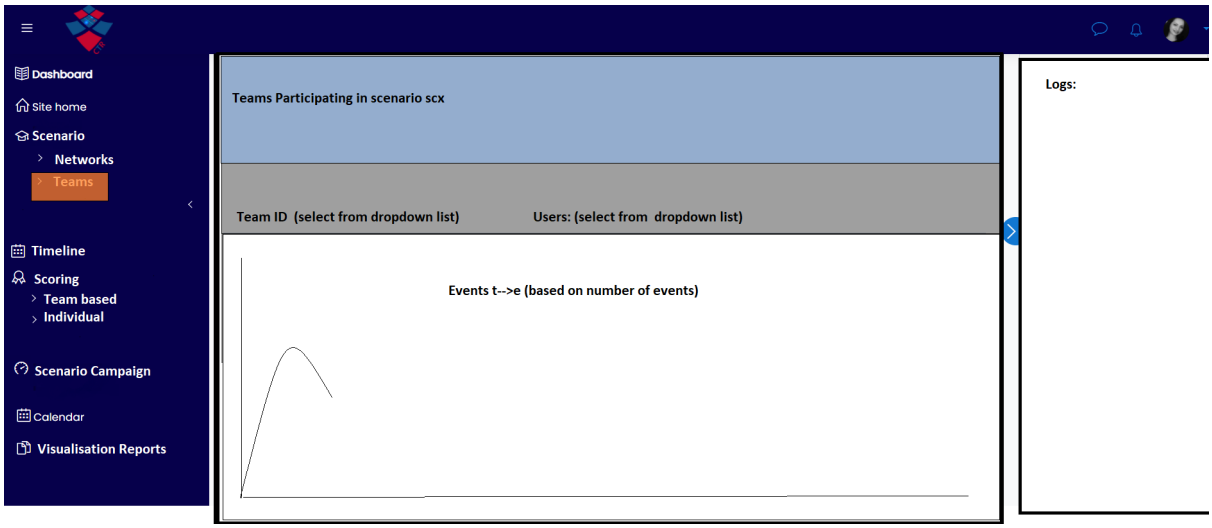
Σχήμα 11 Πρότυπο οθόνης Scenario Networks

b. Teams

Σκοπός της σελίδας αυτής είναι η παρουσίαση πληροφοριών που αφορούν τις ομάδες ενός επιλεγμένου σεναρίου. Ο χρήστης έχει τη δυνατότητα να επιλέξει μια ομάδα ή και συγκεκριμένο χρήστη και να δει τα συμβάντα που τους αφορούν.



Σχήμα 12 Υλοποίηση οθόνης Scenario Teams



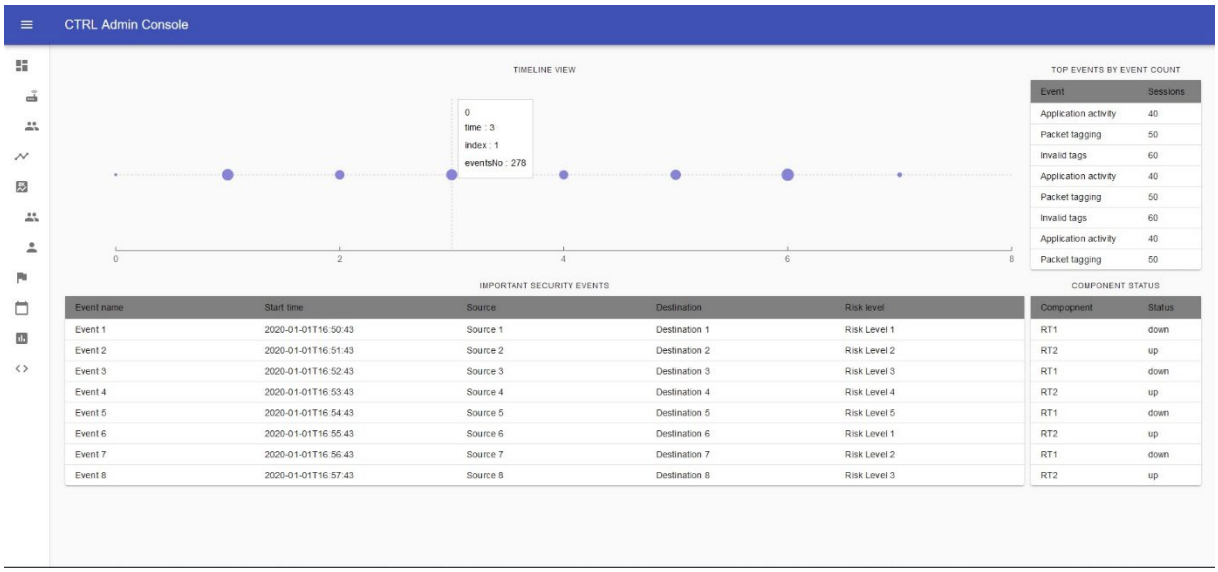
Σχήμα 13 Πρότυπο οθόνης Scenario Teams

2. Timeline

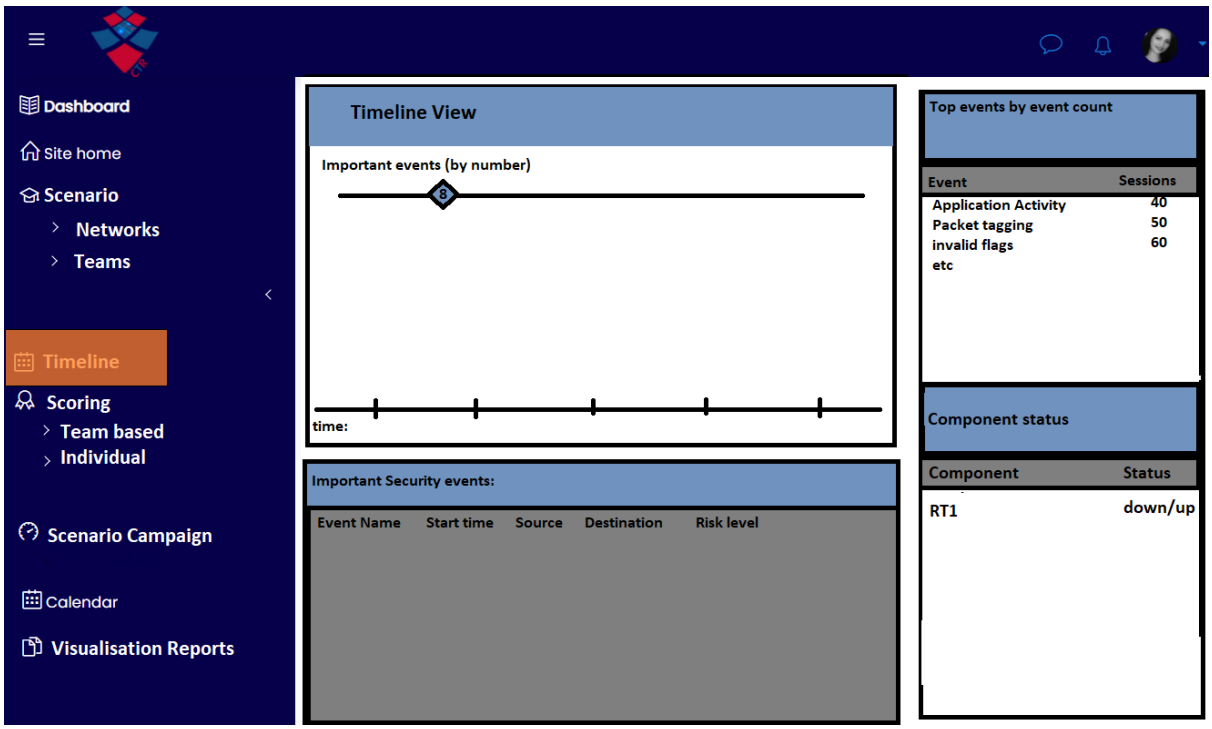
   Στο γράφημα Timeline View παρουσιάζεται ο αριθμός των γεγονότων ασφάλειας που συνέβησαν στο διάστημα του σεναρίου σε χρονοδιάγραμμα. Στον πίνακα “Top events by event count” εμφανίζονται τα γεγονότα τα οποία σημειώθηκαν τις περισσότερες φορές κατά την εκτέλεση της άσκησης, σε φθίνουσα σειρά.

Στον πίνακα “Important Security Events” καταγράφονται ένα – ένα τα γεγονότα με κύριες πληροφορίες το όνομα του γεγονότος, την ώρα εκκίνησης, την πηγή και τον προορισμό, καθώς και το επίπεδο ρίσκου που του έχει αποδοθεί. Ο χρήστης έχει τη δυνατότητα προβολής περαιτέρω πληροφοριών μέσω επιλογής (click) στη γραμμή του γεγονότος.

Στον πίνακα “Component status” μπορεί ο διαχειριστής να δει την κατάσταση των στοιχείων του δικτύου – αν είναι δηλαδή ενεργά, εκτός λειτουργίας η βρίσκονται υπό επίθεση.



Σχήμα 14 Υλοποίηση οθόνης Timeline



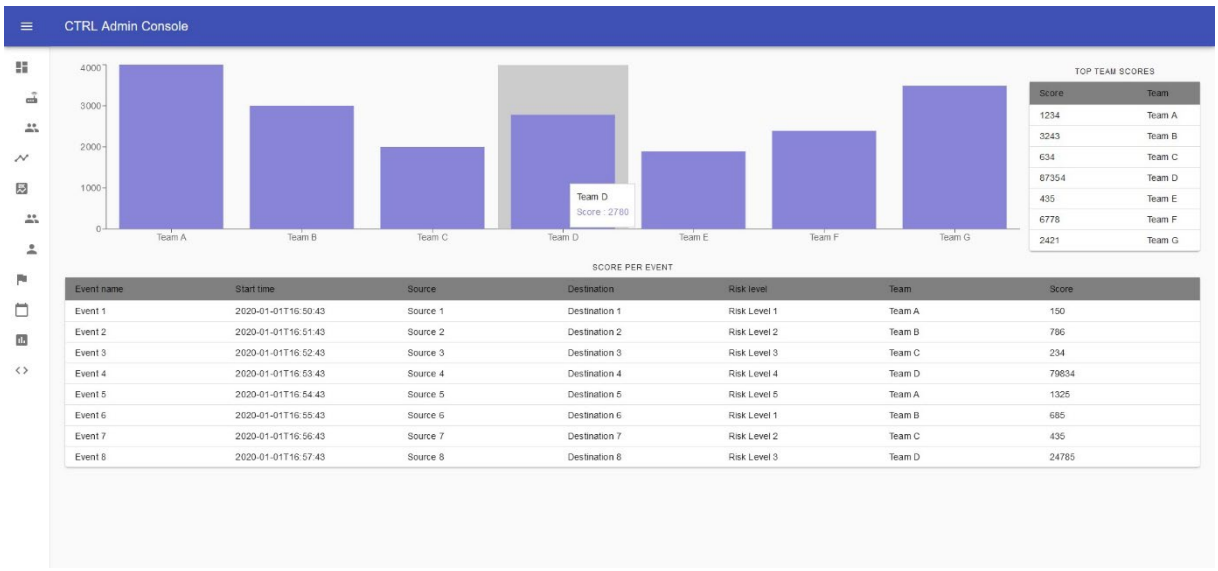
Σχήμα 15 Πρότυπο οθόνης Timeline

3. Scoring

   Σε αυτή την οθόνη δίνεται η δυνατότητα σύγκρισης των αποτελεσμάτων των ομάδων για συγκεκριμένη άσκηση/σενάριο. Ο διαχειριστής επιλέγει σενάριο από την παρεχόμενη λίστα και του παρουσιάζονται τα αποτελέσματα των ομάδων.

   Συγκεκριμένα, η σελίδα περιλαμβάνει γράφημα μπάρας, το οποίο δείχνει τη βαθμολογία ανά ομάδα και στην περίπτωση πατήματος μιας μπάρας, θα οδηγηθεί στην υπο-οθόνη Team-based Scoring, για να δει αναλυτικότερα πληροφορίες που αφορούν τις βαθμολογίες της επιλεγμένης ομάδας.

   Στη σελίδα περιλαμβάνεται επίσης ένας πίνακας στον οποίο φαίνονται οι ομάδες που έλαβαν τη μεγαλύτερη βαθμολογία για κάθε τύπο event. Στον πίνακα “Scores per event” καταγράφονται τα γεγονότα ασφάλειας που εκτυλίχθηκαν κατά την εκτέλεση του σεναρίου και έλαβαν βαθμούς. Για κάθε γεγονός, αποτυπώνεται η ώρα που ξεκίνησε, η πηγή, ο προορισμός, το επίπεδο ρίσκου, η ομάδα που το εκτέλεσε και η βαθμολογία που αντιστοιχεί στο συγκεκριμένο γεγονός.

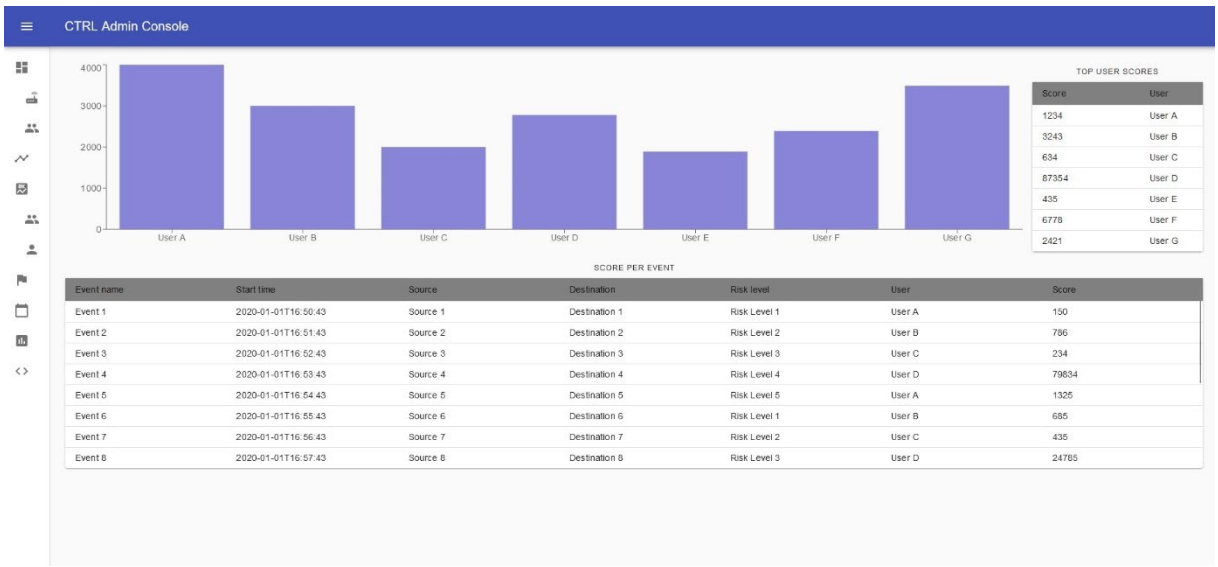


Σχήμα 16 Υλοποίηση οθόνης Scoring

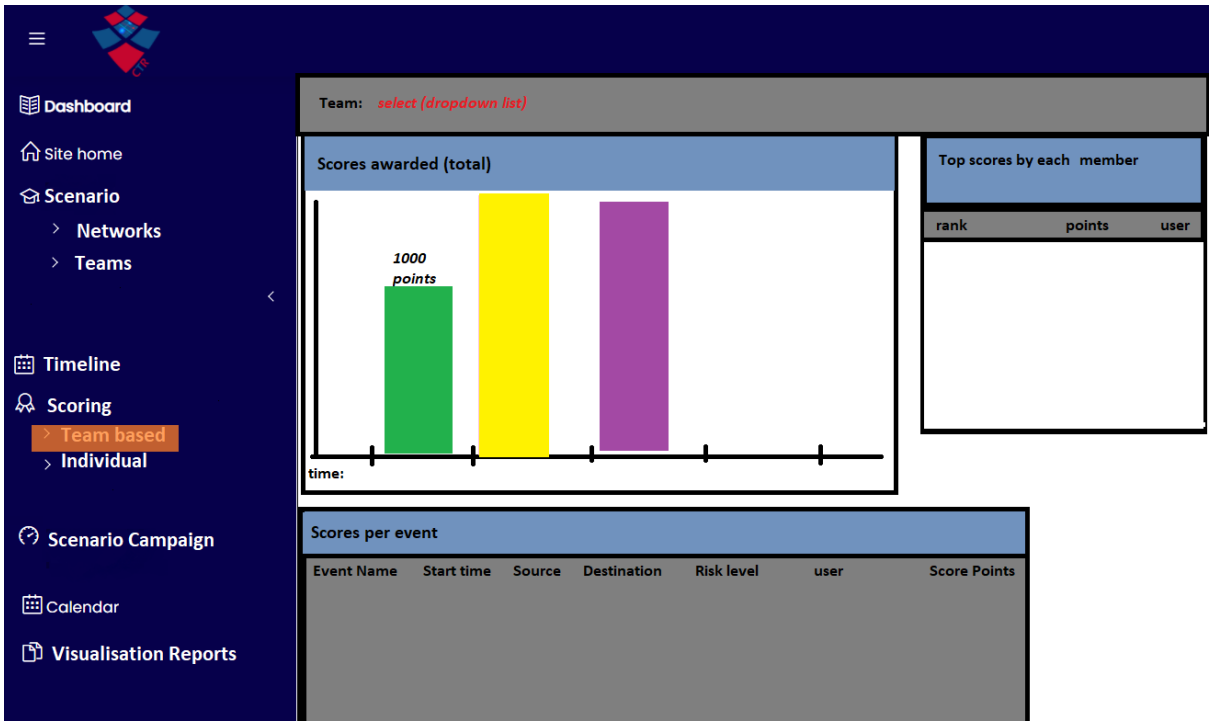
Σχήμα 17 Πρότυπο οθόνης Scoring

a. Team based

Ο χρήστης επιλέγει το σενάριο και την ομάδα που θέλει, για να ενημερωθεί για τους βαθμούς που έλαβε η ομάδα συνολικά, πόσοι βαθμοί κατανεμήθηκαν για κάθε γεγονός που εκτελέστηκε από την ομάδα, καθώς και τη σειρά κατάταξης των μελών με βάση τη βαθμολογία τους.



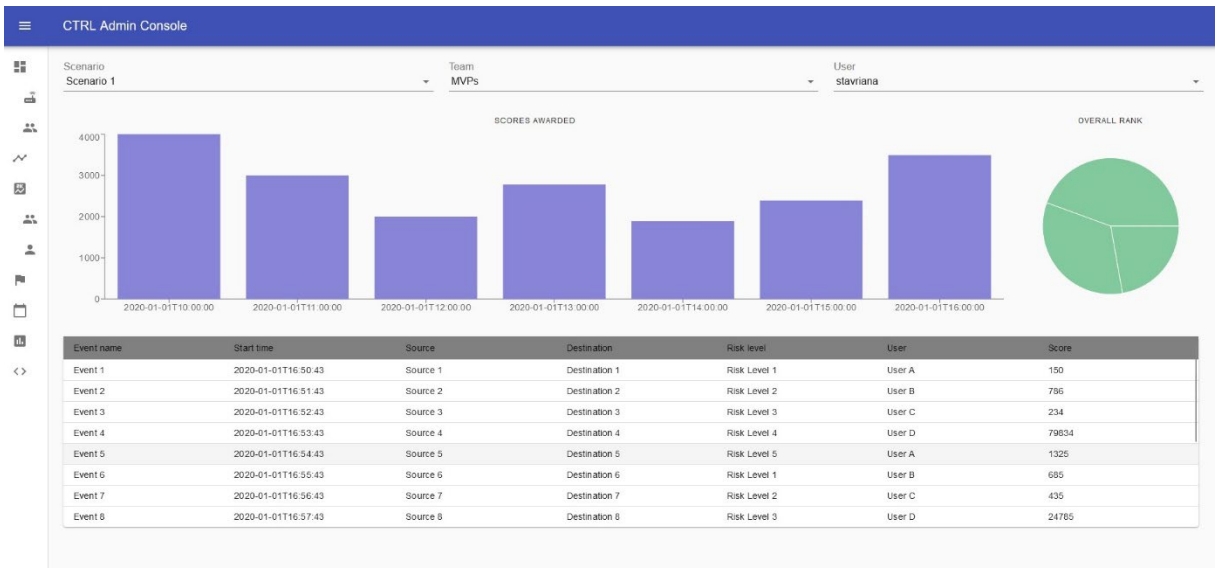
Σχήμα 18 Υλοποίηση οθόνης Team Scoring



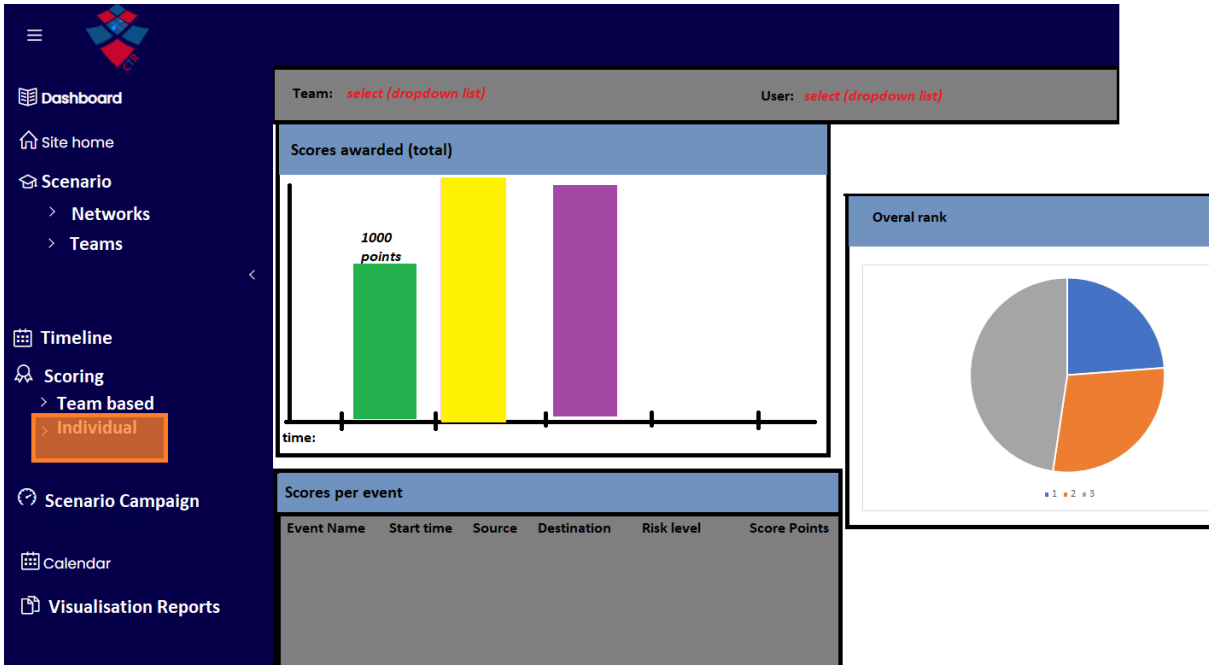
Σχήμα 19 Πρότυπο οθόνης Team Scoring

b. Individual

Ο χρήστης επιλέγει το σενάριο, την ομάδα και συγκεκριμένο χρήστη και αντίστοιχα με τις προηγούμενες οθόνες, βλέπει τους βαθμούς που μάζεψε και τα events στα οποία εμπλέκεται.



Σχήμα 20 Υλοποίηση οθόνης Individual Scoring

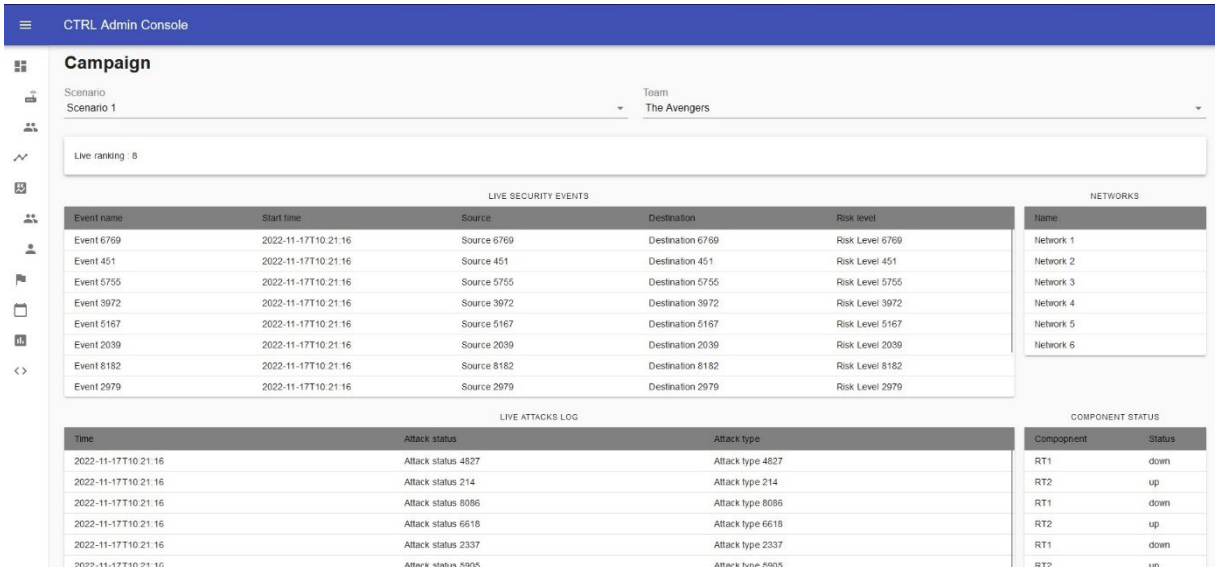


Σχήμα 21 Πρότυπο οθόνης Individual Scoring

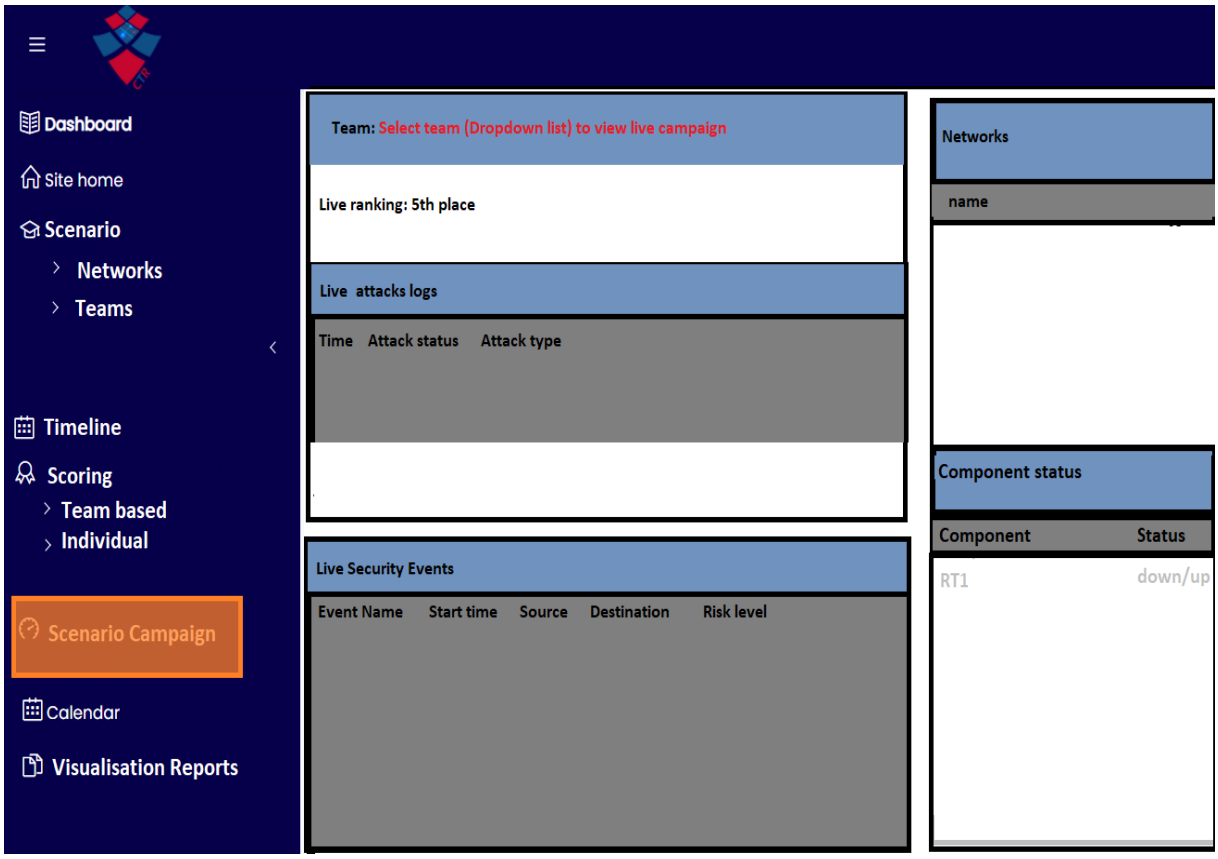
4. Scenario Campaign

   Στη σελίδα αυτή ο χρήστης επιλέγει σενάριο και ομάδα και βλέπει πληροφορίες σχετικά με την πορεία της ομάδας σε πραγματικό χρόνο. Συγκεκριμένα, παρουσιάζονται η τρέχουσα βαθμολογική θέση της ομάδας στην άσκηση, σημαντικά γεγονότα ασφάλειας, logs για τις επιθέσεις που πραγματοποιήθηκαν, τα δίκτυα και η κατάσταση των components. Γίνεται ανανέωση των

πληροφοριών κάθε 5 δευτερόλεπτα, με τη χρήση requests που αποστέλνονται στον εξυπηρετητή από τον client.



Σχήμα 22 Υλοποίηση οθόνης Scenario Campaign

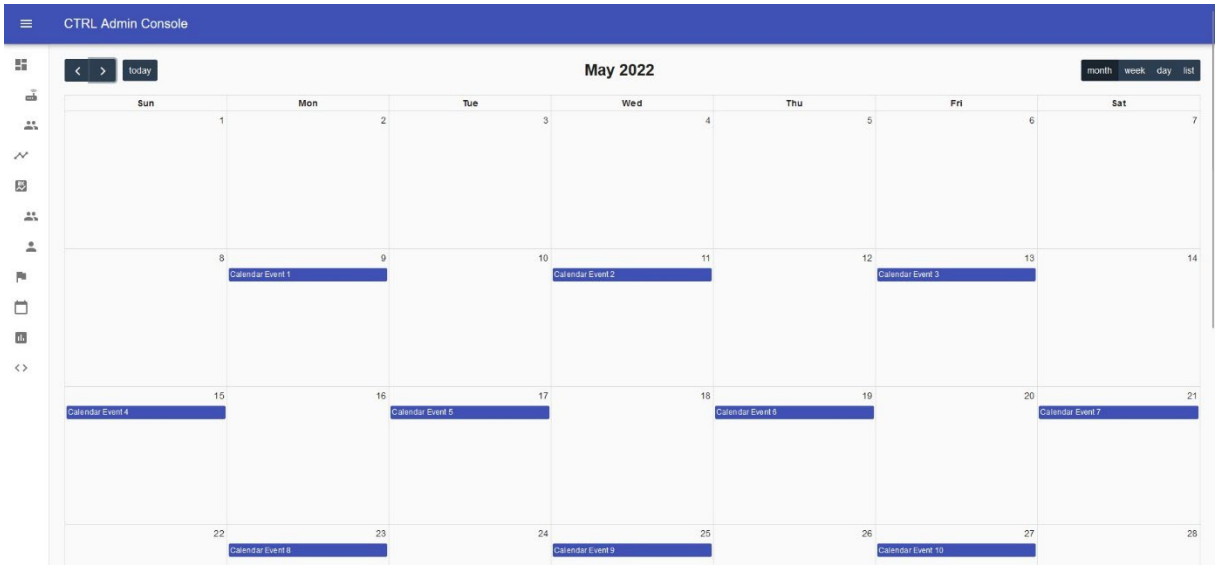


Σχήμα 23 Πρότυπο οθόνης Scenario Campaign

5. Calendar

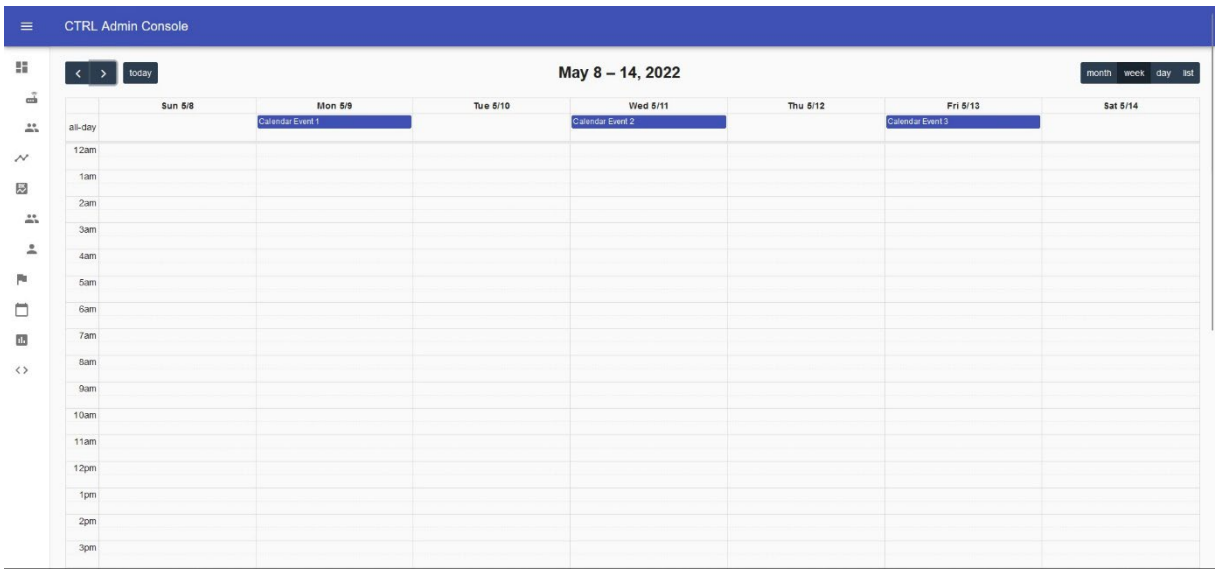
Γίνεται παρουσίαση των προγραμματισμένων δραστηριοτήτων σε ένα ημερολόγιο. Υπάρχουν τέσσερις διαφορετικές όψεις για την προβολή των εργασιών.

Με την επιλογή month εμφανίζονται όλες οι μέρες του μήνα σε κουτιά και τα συμβάντα ανάλογα στο καθένα.



Σχήμα 24 Υλοποίηση οθόνης Calendar - Month view

Με την επιλογή week παρουσιάζονται οι μέρες της επιλεγμένης εβδομάδας κάθετα με τις ώρες στον κατακόρυφο άξονα.



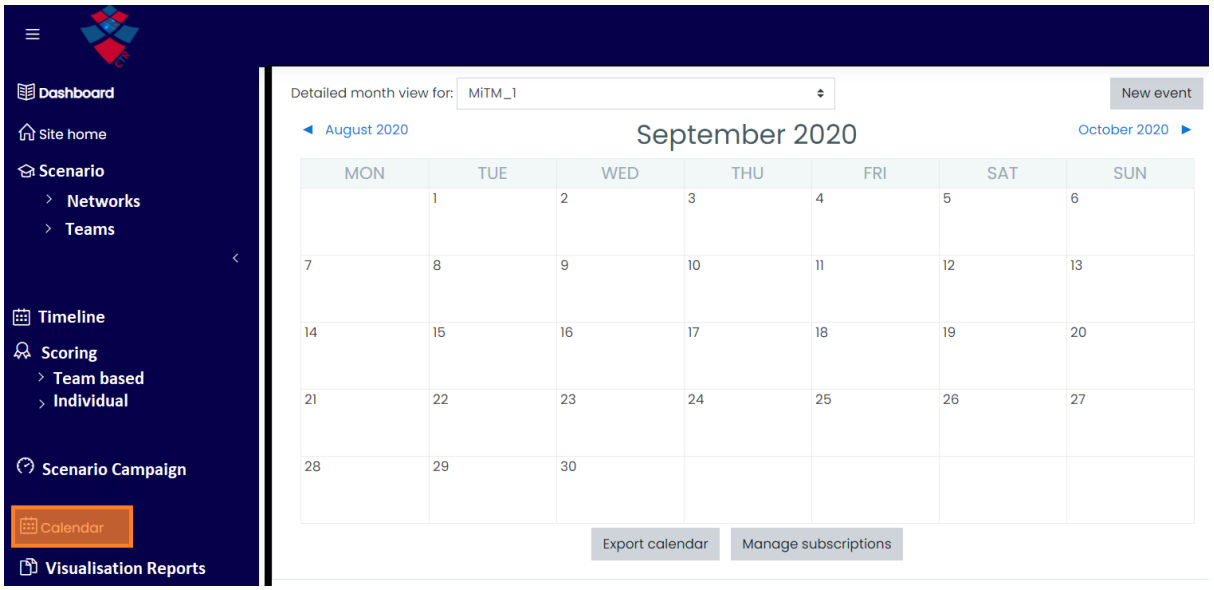
Σχήμα 25 Υλοποίηση οθόνης Calendar - Week view

Παρομοίως εμφανίζεται μια μόνο μέρα με την επιλογή day.

Σχήμα 26 Υλοποίηση οθόνης Calendar - Day view

Με την προβολή λίστας καταγράφονται όλες οι δραστηριότητες στη σειρά για την εβδομάδα, όπως φαίνεται παρακάτω

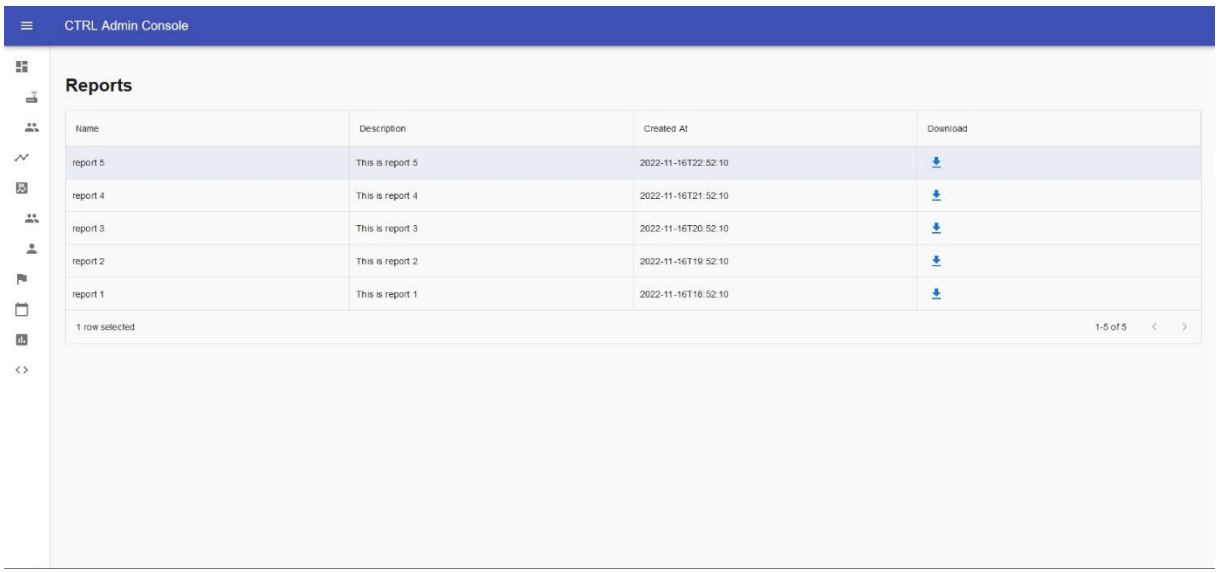
Σχήμα 27 Υλοποίηση οθόνης Calendar - List view



Σχήμα 28 Πρότυπο οθόνης Calendar

6. Visualization Reports

   Ο χρήστης μπορεί να λάβει στον υπολογιστή του (download) αναφορές που βρίσκονται στον πίνακα. Οι αναφορές δημιουργούνται εκτός εφαρμογής.

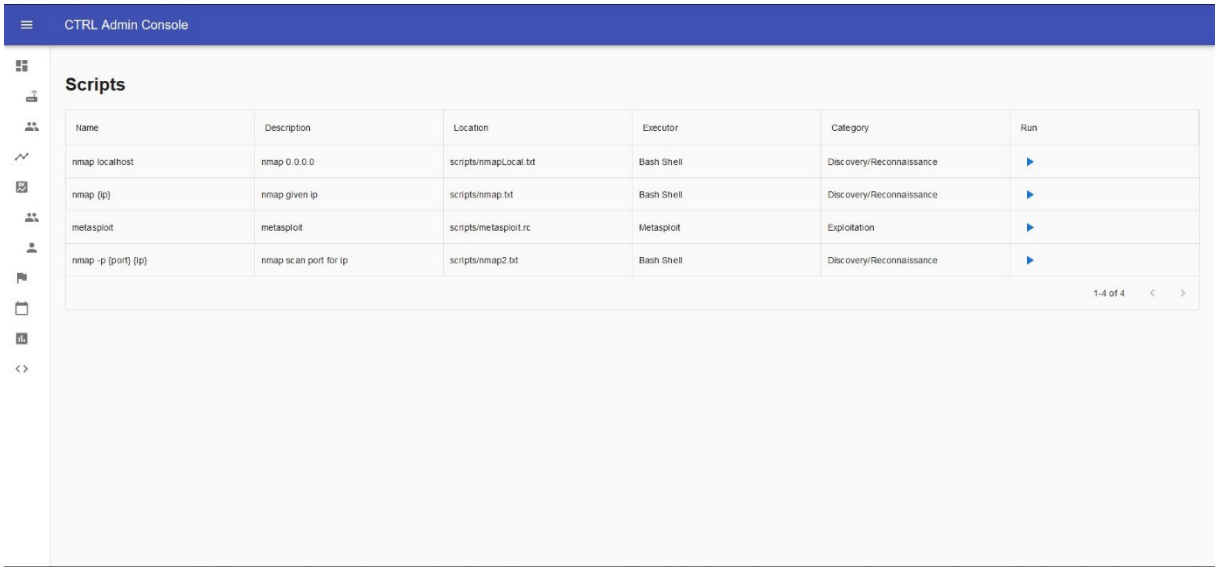


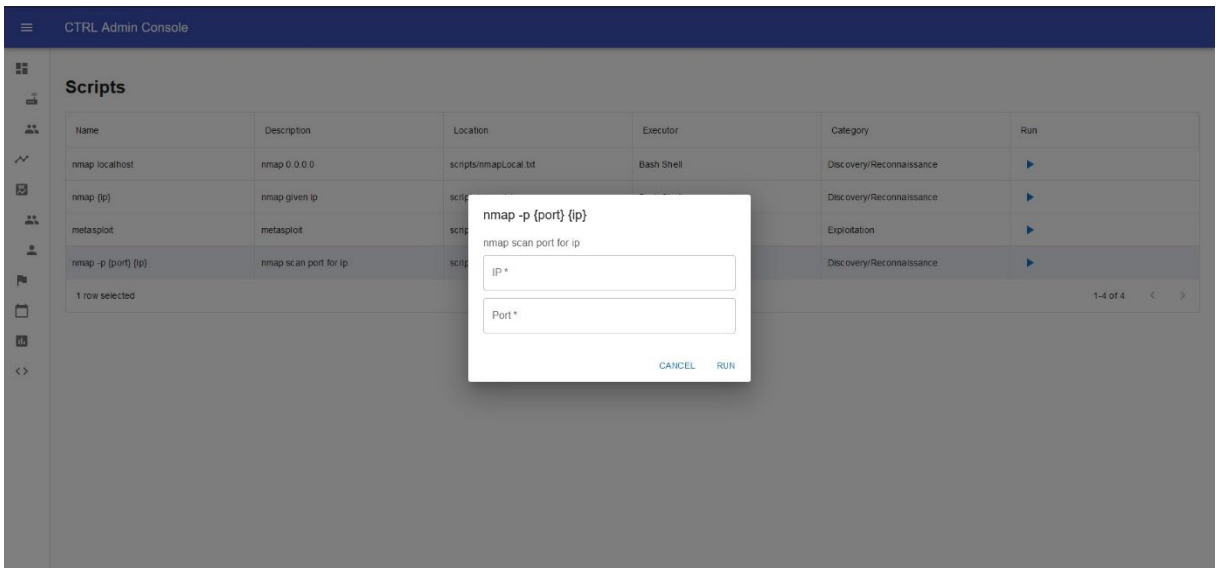
7. Script Management

   Περιλαμβάνει έναν πίνακα με τα αποθηκευμένα scripts τα οποία μπορεί να επιλέξει και να εκτελέσει ο χρήστης, θέτοντας τιμές σε πιθανές παραμέτρους.

   Ακολουθεί ένας πίνακας με τα scripts που έχουν προγραμματιστεί να εκτελεστούν ως αντίδραση σε κάποιο συμβάν που έχει καταγραφεί στην πλατφόρμα.

   Τέλος, υπάρχει ένας πίνακας με τις προγραμματισμένες εκτελέσεις επιθέσεων σε συγκεκριμένο χρόνο.





4.2.3 Αυτοματοποίηση Red Attacks Scripts

Στόχος της αυτοματοποίησης είναι η διευκόλυνση της διεκπεραίωσης των επιλεγμένων σεναρίων των ασκήσεων που σχεδιάζονται στο cyber range. Θέτουμε τις παρακάτω προδιαγραφές για το σχεδιασμό μας:

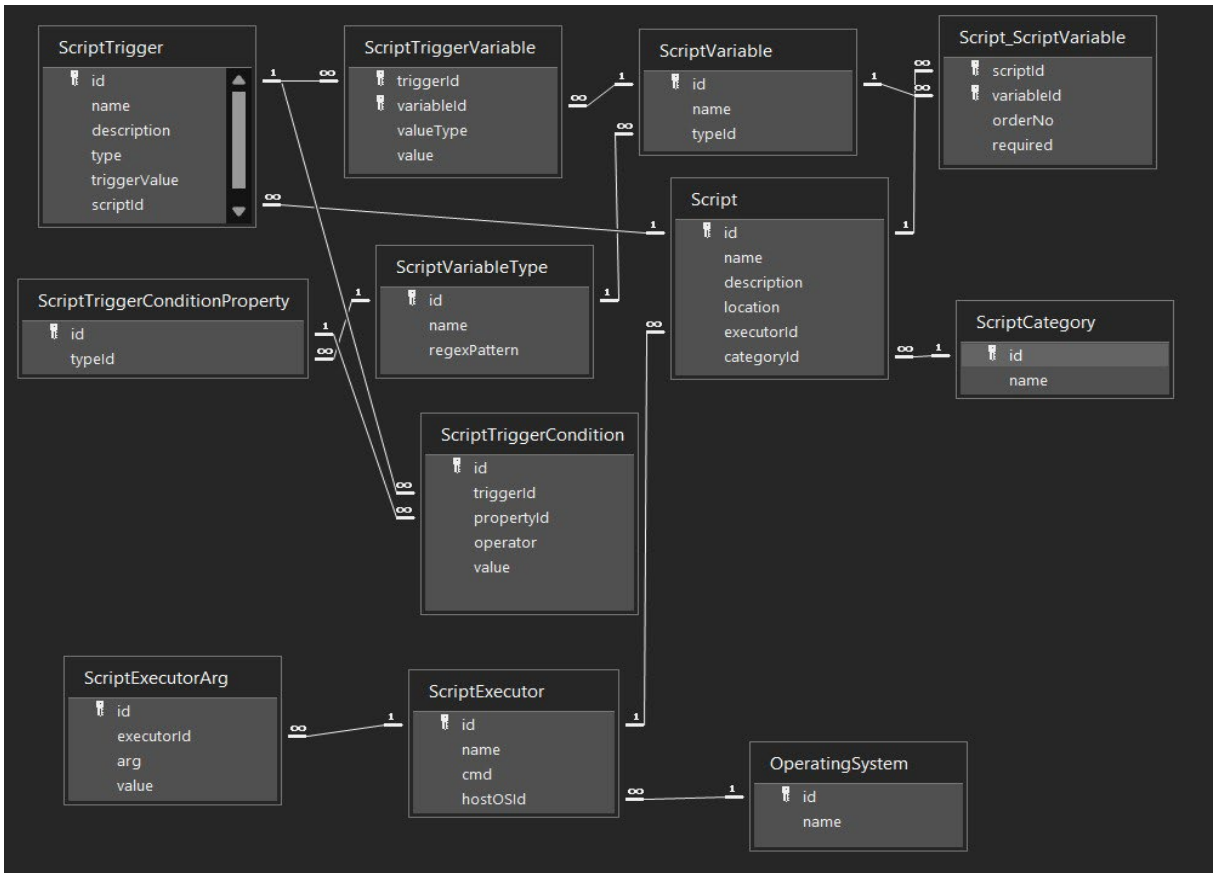
1. Ευκολία χρήσης: Άτομα με ελάχιστη γνώση να μπορούν να εκτελέσουν επιλεγμένες επιθέσεις.
2. Ευελιξία: Με μικρές αλλαγές να επιτυγχάνονται καινούριοι στόχοι, να αλλάζει το επίπεδο δυσκολίας ή να υπάρχει διαβάθμιση των σεναρίων.
3. Επαναληψιμότητα: Ο σχεδιασμός σεναρίων χρειάζεται μελέτη και κόπο, οπότε είναι επιθυμητό να δύναται να επαναληφθεί με διαφορετικές ομάδες ή ακόμα και για τον έλεγχο της προόδου της ίδιας ομάδας.
4. Επεκτασιμότητα: Πρέπει να μπορεί να γίνει επέκταση των λειτουργιών της εφαρμογής με άλλα εργαλεία.
5. Φορητότητα: Ο σχεδιασμός να επιτρέπει την εκτέλεση scripts σε διαφορετικά λειτουργικά συστήματα.

Για τον πρώτο στόχο γίνεται προσθήκη των διαθέσιμων scripts στο γραφικό περιβάλλον, όπου πατώντας το κουμπί εκτέλεσης, ο χρήστης θα ενημερωθεί για τις παραμέτρους που πρέπει να δώσει και τον τύπο τους και η εντολή του θα εκτελεστεί αμέσως.

Ευελιξία και επαναληψιμότητα μπορούμε να επιτύχουμε δίνοντας τη δυνατότητα στη white team να καθορίσει πότε πρέπει να εκτελεστεί μια επίθεση είτε προγραμματίζοντας τη στατικά σε συγκεκριμένο χρόνο, με βάση τον προγραμματισμό της άσκησης, είτε πιο δυναμικά ως αντίδραση σε κάποιο συμβάν που θα καταγραφεί στο σύστημα. Όπως έχει αναφερθεί προηγουμένως, στο συγκεκριμένο cyber range τα συμβάντα που καταγράφονται είναι από τους Suricata και Wazuh agents και η κανονικοποίησή τους μας επιτρέπει να χρησιμοποιήσουμε τις παραμέτρους τους ως συνθήκες ελέγχου για την εκτέλεση επίθεσης.

Υπολογίζοντας πιθανή επεκτασιμότητα και φορητότητα, αντιστοιχίζουμε το κάθε script στο λειτουργικό σύστημα στο οποίο μπορεί να εκτελεστεί, στο πρόγραμμα που θα το εκτελέσει, τις παραμέτρους που χρειάζονται και το είδος τους, καθώς επίσης και σε ποια φάση-κατηγορία ανήκει το script, ούτως ώστε σε περίπτωση πολλών scripts να διευκολύνεται η αναζήτηση.

Καταλήγουμε στον εξής σχεδιασμό της βάσης δεδομένων:



Σχήμα 29 Σχήμα βάσης δεδομένων της εφαρμογής

Με την εντολή subprocess.run της python, μπορούμε να εκτελέσουμε ένα script, δίνοντας τυχόν αρχικές παραμέτρους και λαμβάνοντας το αποτέλεσμα. Επιλέγουμε να ανακατευθύνουμε τα data streams stdout και stderr της κάθε εκτέλεσης σε ένα καινούριο αρχείο στο φάκελο api/scripts/output, για να μπορεί να γίνει επισκόπηση αργότερα. Επιπλέον, για αποφυγή κατασπατάλησης των πόρων του συστήματος, θέτουμε το χρονικό περιθώριο εκτέλεσης της κάθε εντολής να είναι το μέγιστο 2 λεπτά.

Για δοκιμή επιλέγουμε να ενσωματώσουμε μερικά εργαλεία από αυτά που έχουν παρουσιαστεί στο Κεφάλαιο 2 και μπορούν να εκτελεστούν μέσα από τη γραμμή εντολών. Συγκεκριμένα θα δοκιμάσουμε τα nmap, Metasploit και python scripts. Εννοείται πως πρέπει να προηγηθεί εγκατάσταση των εργαλείων για να χρησιμοποιηθούν.

Για την εγκατάσταση και επαλήθευση του nmap:

```
$ sudo apt-get install nmap
$ nmap --version
```

Για το Metasploit Framework ακολουθούμε τις οδηγίες που βρίσκονται στο επίσημο documentation https://docs.rapid7.com/metasploit/installing-the-metasploit-framework/#installing-the-metasploit-framework-on-linux :

```
$ curl https://raw.githubusercontent.com/rapid7/metasploit-
omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb >
msfinstall && chmod 755 msfinstall && ./msfinstall
$ msfconsole
```

Τα python scripts με κατάληξη py εκτελούνται με την εντολή python3 (ή python ανάλογα με το λειτουργικό σύστημα και την έκδοση), επομένως αυτό είναι και το εκτελέσιμο που θα δηλώσουμε στη subprocess.run.

Για τα metasploit scripts θα χρησιμοποιήσουμε την εντολή msfconsole με το argument -r, το οποίο δηλώνει resource και ακολουθεί το όνομα του script file με κατάληξη rc. Μέσα στο αρχείο μπορούν να περιληφθούν οποιεσδήποτε εντολές του Metasploit, όπως αυτές θα γράφονταν στην κονσόλα. Σε περίπτωση που περιλαμβάνονται παράμετροι, αυτές πρέπει να δηλωθούν εκτός από το script και στη βάση δεδομένων.

Το nmap δεν εκτελεί scripts με συγκεκριμένη κατάληξη. Για να εκτελέσουμε τις εντολές nmap μπορούμε να τις γράψουμε σε ένα αρχείο κειμένου με κατάληξη txt ή σε αρχείο ενός shell, όπως είναι το bash με κατάληξη sh. Προτιμούμε την πρώτη επιλογή, εφόσον είναι απλούστερη και μπορεί να χρησιμοποιηθεί το ίδιο αρχείο σε διαφορετικά shells.

Χρησιμοποιώντας τα shell environments των λειτουργικών συστημάτων, μπορούμε να εκτελέσουμε οποιοδήποτε αρχείο περιλαμβάνει εντολές συστήματος, όπως πράττουμε και με το nmap. Στο λειτουργικό σύστημα Ubuntu που εργαζόμαστε, χρησιμοποιούμε το bash, αλλά αντίστοιχα θα μπορούσαμε να χρησιμοποιήσουμε το PowerShell για Windows. Αυτό μας δίνει τη δυνατότητα να ενσωματώσουμε όλα τα εργαλεία που είναι διαθέσιμα σε γραμμή εντολών και δεν χρειάζονται περαιτέρω διάδραση με το χρήστη μετά την αρχική εντολή.

Όλα τα αρχεία script νοείται ότι πρέπει να είναι εκτελέσιμα. Σε λειτουργικό σύστημα Unix, μπορούμε να αποδώσουμε το permission με το
$ chmod +x *filename*

Η εκτέλεση σε συγκεκριμένο χρόνο απαιτεί την ύπαρξη ενός cron job, δηλαδή μιας χρονοπρογραμματισμένης περιοδικής εργασίας συστήματος με την οποία θα καλούμε το python script μας που θα ελέγχει για την ύπαρξη εκκρεμών εργασιών στη βάση δεδομένων και θα τις εκτελεί ανάλογα.

Η εκτέλεση επιθέσεων ως αντίδραση σε κάποιο event, απαιτεί την ενημέρωση της εφαρμογής μας όταν αυτά συμβαίνουν. Μια επιλογή είναι η ενημέρωση να γίνεται από τους συλλέκτες των συμβάντων του cyber range και την αποστολή τους σε συγκεκριμένο url της εφαρμογής σε πραγματικό χρόνο. Για να παραμείνουμε όμως στα όρια της εφαρμογής μας, θα χρησιμοποιήσουμε και εδώ το cron job, το οποίο μπορούμε να

θέσουμε να εκτελείται με τη μέγιστη συχνότητα τους ενός λεπτού για να μην υπάρχουν αισθητές καθυστερήσεις. Προγραμματιστικά θα ελέγχουμε τη βάση δεδομένων του cyber range για τα events και θα τα καταγράφουμε σε ένα άλλο Table στη δική μας βάση, που θα λειτουργεί ως ουρά για τα συμβάντα που πρέπει να επεξεργαστούμε.

Κεφάλαιο 5
5 Επίλογος

Στην παρούσα διατριβή έγινε μια επισκόπηση των cyber ranges, τη λειτουργία και τη χρησιμότητα τους. Εστιάσαμε ιδιαίτερα στη λευκή ομάδα που οργανώνει και χρειάζεται να παρακολουθεί τις ασκήσεις, καθώς και στην κόκκινη ομάδα και στα εργαλεία που αυτή χρησιμοποιεί. Ακολουθεί μια σύντομη ανακεφαλαίωση, αξιολόγηση της εργασίας μας και μελλοντικές κατευθύνσεις για περαιτέρω ανάπτυξη.

5.1 Ανακεφαλαίωση

Η αύξηση των διαδικτυακών υπηρεσιών και των δεδομένων των χρηστών που συλλέγονται, καθιστά αναγκαία την εξασφάλιση της διαθεσιμότητας, ακεραιότητας και εμπιστευτικότητας αυτών. Τα εικονικά περιβάλλοντα cyber ranges προσφέρουν ένα ασφαλές περιβάλλον για την αξιολόγηση της ασφάλειας συστημάτων και τη δοκιμή νέων τεχνολογιών και αποδεικνύονται χρήσιμο εργαλείο για την εκπαίδευση και την ενίσχυση των δεξιοτήτων επαγγελματιών κυβερνοασφάλειας. Η ανάπτυξη τέτοιων περιβαλλόντων, καθώς και η δημιουργία ασκήσεων σε αυτά είναι διαδικασίες απαιτητικές σε χρόνο και κόστος. Επομένως, χρειάζονται εργαλεία και πρακτικές για τη διευκόλυνση στη διεκπεραίωσή τους. Η ύπαρξη γραφικού περιβάλλοντος μπορεί να βοηθήσει τους διοργανωτές στην παρακολούθηση του σεναρίου και αξιολόγηση της έκβασης των ασκήσεων. Η αυτοματοποίηση επιθέσεων επιτρέπει την επαναληψιμότητα των σεναρίων και την εξάλειψη της ανάγκης ύπαρξης κόκκινης ομάδας, μειώνοντας το κόστος και το πρόβλημα της έλλειψης ανάλογων επαγγελματιών.

5.2 Αξιολόγηση

Υλοποιήσαμε ένα γραφικό περιβάλλον συμβατό με το cyber range του Ανοικτού Πανεπιστημίου Κύπρου, με σκοπό τη διευκόλυνση των διαχειριστών στην

παρακολούθηση και αξιολόγηση της εξέλιξης των σεναρίων. Το γραφικό περιβάλλον περιλαμβάνει γραφήματα, ενημέρωση για την κατάσταση του συστήματος και λεπτομέρειες για τα γεγονότα που εκτυλίσσονται. Ακολουθήθηκε δοθέν πρότυπο για τη σχεδίαση των οθονών, με κάποιες αποκλίσεις στην υλοποίηση, κυρίως λόγω δυσκολιών στο σχεδιασμό γραφημάτων. Συγκεκριμένα στο γράφημα δικτύου αναμενόταν η ίδια αναπαράσταση που φαίνεται στην πλατφόρμα ManageIQ, αλλά η δημιουργία παρόμοιου περιβάλλοντος και η σύνδεση σε αυτό δεν κατέστη δυνατή. Έτσι χρησιμοποιήθηκε ένα άλλο γράφημα δικτύου, χωρίς τα επιθυμητά εικονίδια. Στο γράφημα χρονοδιαγράμματος, χρησιμοποιήθηκε διάγραμμα διασποράς για παρόμοια αναπαράσταση με τη δοθείσα. Τέλος, δεν δημιουργήθηκε η οθόνη για τη σύνδεση χρηστών, γιατί δεν υλοποιήθηκε ο μηχανισμός αυθεντικοποίησης. Η εφαρμογή που αναπτύξαμε μπορεί να θεωρηθεί ως πρώιμη έκδοση, που δημιουργήθηκε για την παρουσίαση υφιστάμενων πληροφοριών που συλλέγονται από την υπάρχουσα υποδομή του cyber range. Πρέπει, βεβαίως, να σημειώσουμε ότι όλα τα δεδομένα προέρχονται από βάση δεδομένων που δημιουργήθηκε σε αντιστοιχία με το σχήμα της πραγματικής βάσης και δεν ήταν η πραγματική. Σίγουρα χρειάζεται να γίνουν πολλές βελτιώσεις εμφανισιακές, αλλά και προσθήκες στις λειτουργίες της για να ολοκληρωθεί. Μερικές από αυτές αναφέρονται στην επόμενη ενότητα 5.3.

Για τις επιθέσεις, μέσω βιβλιογραφικής και διαδικτυακής έρευνας, εντοπίσαμε scripts και εργαλεία που χρησιμοποιούνται από την κόκκινη ομάδα και προτείναμε έναν τρόπο ενσωμάτωσης τους στην εφαρμογή. Προσθέσαμε τη δυνατότητα αυτοματοποίησης της εκτέλεσης των επιθέσεων μέσω χρονοπρογραμματισμού ή ως αντιδράσεις σε συμβάντα που καταγράφονται κατά τη διάρκεια της άσκησης, βοηθώντας τη λευκή ομάδα να υλοποιήσει τα σενάριά της, χωρίς την ανάγκη κόκκινης ομάδας. Έμφαση δόθηκε στο σχεδιασμό του συστήματος και τον τρόπο με τον οποίο θα υλοποιηθεί η αυτοματοποίηση, όμως το δείγμα των scripts που χρησιμοποιήθηκε ήταν μικρό.

5.3 Μελλοντικά σχέδια

Είναι κατανοητό ότι ο σχεδιασμός και η υλοποίηση ενός ολοκληρωμένου συστήματος με πολλαπλά χαρακτηριστικά, όπως είναι η οπτικοποίηση των αποτελεσμάτων και η αυτοματοποίηση επιθέσεων, είναι εργασίες χρονοβόρες και ατέρμονες. Χρειάζονται

συνεχή αναβάθμιση, για να συμβαδίζουν με τα νέα δεδομένα και τεχνολογίες, αλλά και βελτιώσεις για καλύτερη εμπειρία χρήστη.

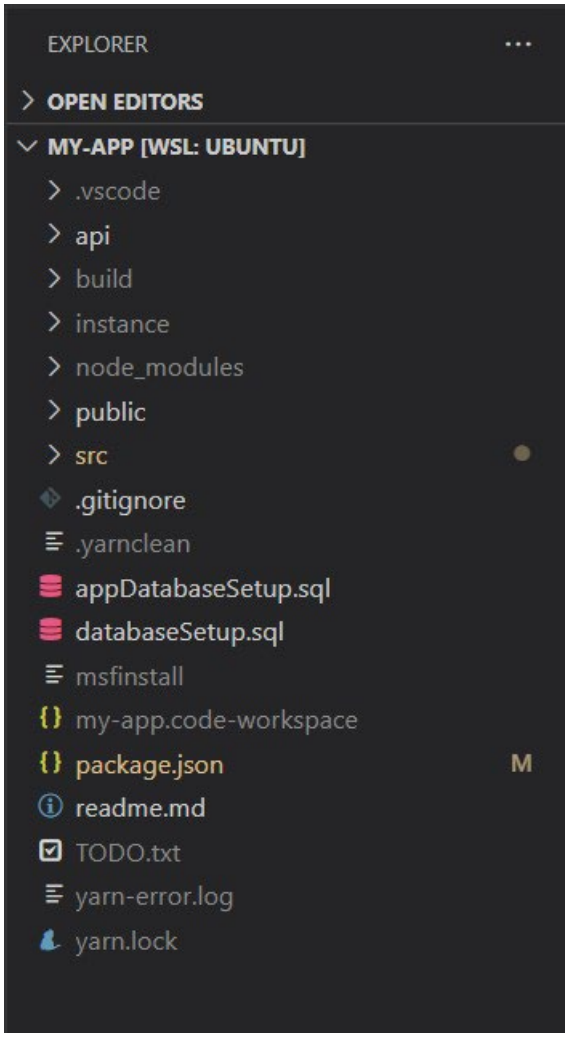
Μερικά παραδείγματα για βελτίωση του γραφικού περιβάλλοντος που υλοποιήσαμε, είναι η προσθήκη της δυνατότητας αναζήτησης στις διάφορες οθόνες, είτε αυτό αφορά τα logs, τα scripts ή τις αναφορές και η παρουσίαση όλων των εκτελέσεων των επιθέσεων με τη δυνατότητα προβολής της εξόδου τους. Επιπλέον, για τη χρήση της εφαρμογής σε production περιβάλλον πρέπει να προστεθεί μηχανισμός αυθεντικοποίησης και εξουσιοδότησης ατόμων. Αυτό ανοίγει το δρόμο για απόδοση ρόλων και δικαιωμάτων και τη δημιουργία διαφορετικών όψεων σε κάθε έναν από αυτούς.

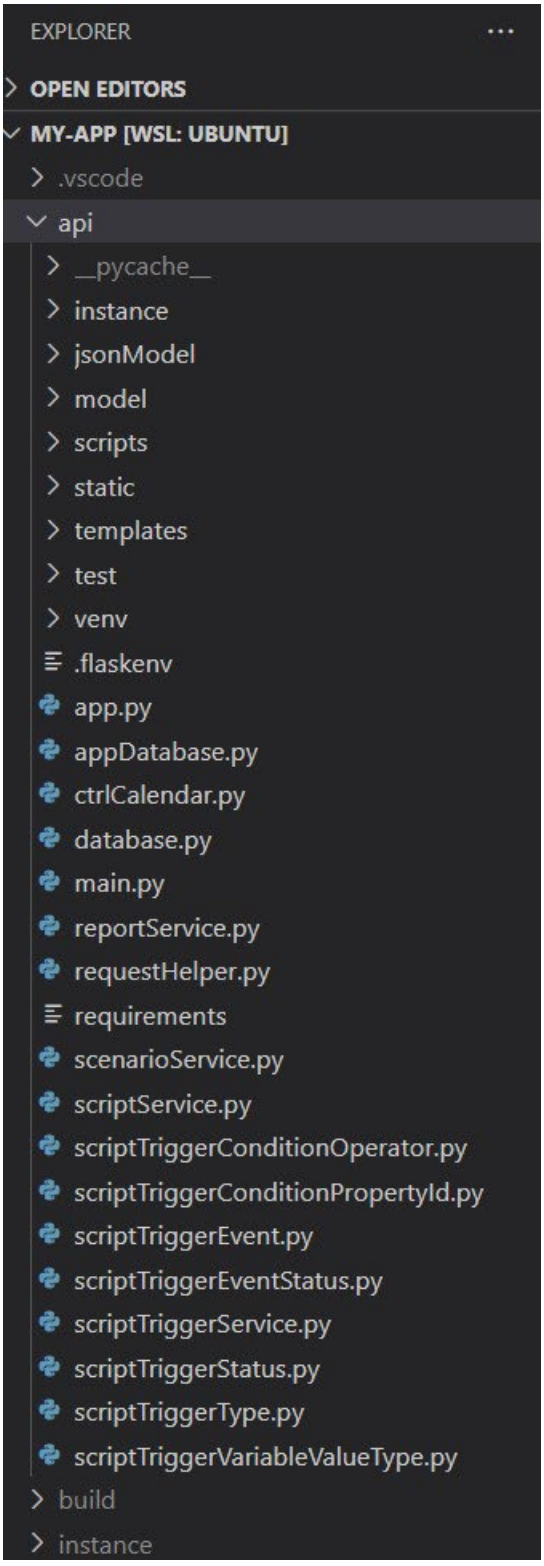
Όσον αφορά τις επιθέσεις, για την ανάπτυξη της εφαρμογής μας πρέπει σαφώς να προστεθούν περισσότερα και πιο πολύπλοκα scripts και να ενσωματωθούν επιπλέον εργαλεία και για άλλα λειτουργικά συστήματα. Σημειώνουμε ότι με την πρότασή μας αυτοματοποιείται η εκτέλεση των επιθέσεων, αλλά εκκρεμεί το θέμα της λήψης των αποφάσεων για το ποια επίθεση θα εκτελεστεί και πότε, η οποία πρέπει να καθοριστεί από τους διοργανωτές των ασκήσεων εκ των προτέρων. Όπως αναφέραμε στο κεφάλαιο της βιβλιογραφίας, το θέμα της αυτοματοποιημένης δημιουργίας των βημάτων επίθεσης, παραμένει υπό διερεύνηση και πιθανές λύσεις ίσως μπορεί να δώσει η τεχνολογία της τεχνητής νοημοσύνης και της μηχανικής μάθησης.

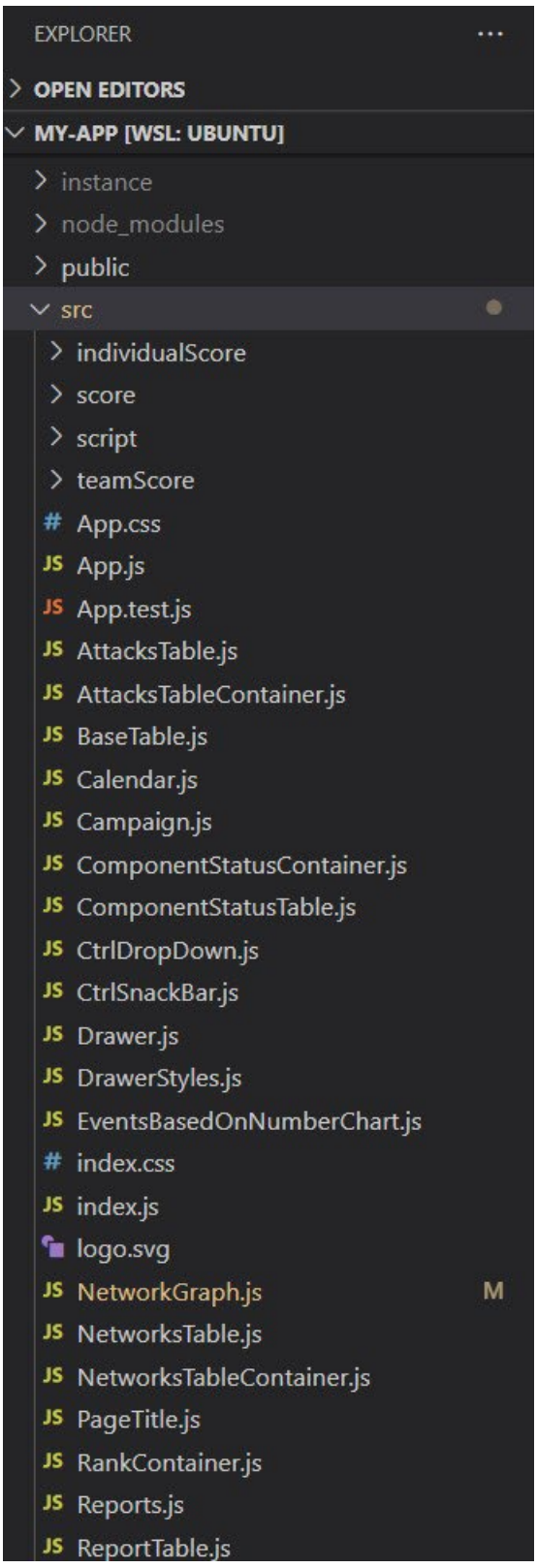
Παράρτημα Α
Α Κώδικας Εφαρμογής

Α.1 Δομή εφαρμογής

Στο φάκελο api τοποθετείται ο κώδικας του backend, ενώ στο src ο κώδικας του frontend. Στο φάκελο jsonModel υπάρχουν τα μοντέλα που ανταλλάσσονται μεταξύ server και client, ενώ στο model τα μοντέλα της βάσης δεδομένων. Στο φάκελο scripts, τοποθετούνται τα scripts και στο scripts/output η έξοδος των εκτελέσεών τους.




EXPLORER
OPEN EDITORS
MY-APP [WSL: UBUNTU]
.vscode
api
__pycache__
instance
jsonModel
model
scripts
static
templates
test
venv
.flaskenv
app.py
appDatabase.py
ctrlCalendar.py
database.py
main.py
reportService.py
requestHelper.py
requirements
scenarioService.py
scriptService.py
scriptTriggerConditionOperator.py
scriptTriggerConditionPropertyId.py
scriptTriggerEvent.py
scriptTriggerEventStatus.py
scriptTriggerService.py
scriptTriggerStatus.py
scriptTriggerType.py
scriptTriggerVariableValueType.py
build
instance


EXPLORER
OPEN EDITORS
MY-APP [WSL: UBUNTU]
instance
node_modules
public
src
individualScore
score
script
teamScore
App.css
App.js
App.test.js
AttacksTable.js
AttacksTableContainer.js
BaseTable.js
Calendar.js
Campaign.js
ComponentStatusContainer.js
ComponentStatusTable.js
CtrlDropDown.js
CtrlSnackBar.js
Drawer.js
DrawerStyles.js
EventsBasedOnNumberChart.js
index.css
index.js
logo.svg
NetworkGraph.js
M
NetworksTable.js
NetworksTableContainer.js
PageTitle.js
RankContainer.js
Reports.js
ReportTable.js

A.2 Κώδικας Backend

api/app.py

```
import os
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

def create_app():
    app = Flask(__name__)
    app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql://root:password@localhost/CTRL'

    from scenarioService import scenarios_api
    app.register_blueprint(scenarios_api)

    from ctrlCalendar import calendar_api
    app.register_blueprint(calendar_api)

    from scriptService import scripts_api
    app.register_blueprint(scripts_api)

    from scriptTriggerService import script_trigger_api
    app.register_blueprint(script_trigger_api)

    from reportService import report_api
    app.register_blueprint(report_api)

    try:
        os.makedirs(app.instance_path)
    except OSError:
        pass

    return app
```

api/appDatabase.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker
from sqlalchemy.ext.declarative import declarative_base

app_db_engine = create_engine('mysql://root:password@localhost/CTRL_APP',
convert_unicode=True)
app_db_session = scoped_session(sessionmaker(autocommit=False,
                                             autoflush=False,
                                             bind=app_db_engine))
App_db_base = declarative_base()
App_db_base.query = app_db_session.query_property()
```

```
def init_db():
    App_db_base.metadata.create_all(bind=app_db_engine)
```

api/ctrlCalendar.py

```
import logging
from flask import Blueprint
from flask import request
from flask import current_app
from flask import json
from flask import make_response
from model.calendarEvent import CalendarEvent
from jsonModel.calendarEventSchema import calendar_events_schema

calendar_api = Blueprint('calendar_api', __name__)

@calendar_api.route('/api/calendar', methods=['GET'])
def calendar():
    current_app.logger.info("get calendar")
    current_app.logger.info("request.args.get(start) : " +
request.args.get("start"))
    current_app.logger.info("request.args.get(end) : " +
request.args.get("end"))
    list =
CalendarEvent.query.filter(CalendarEvent.isoDateFrom>=request.args.get("start"
), CalendarEvent.isoDateTo<request.args.get("end")).all()
    current_app.logger.info("json list " +
str(json.dumps(calendar_events_schema.dump(list))))
    response = make_response(json.dumps(calendar_events_schema.dump(list)))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response
```

api/database.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker
from sqlalchemy.ext.declarative import declarative_base

engine = create_engine('mysql://root:password@localhost/CTRL',
convert_unicode=True)
db_session = scoped_session(sessionmaker(autocommit=False,
                                         autoflush=False,
                                         bind=engine))
Base = declarative_base()
Base.query = db_session.query_property()

def init_db():
    Base.metadata.create_all(bind=engine)
```

api/main.py

```
from os import abort
import time
import logging
import app

app2 = app.create_app()

logger = logging.getLogger('main')
logger.setLevel(logging.DEBUG)
ch = logging.StreamHandler()
ch.setLevel(logging.ERROR)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
%(message)s')
ch.setFormatter(formatter)
logger.addHandler(ch)
```

api/reportService.py

```
from flask import Blueprint
from flask import request
from flask import current_app
from flask import json
from flask import make_response
from model.ctrl.report import Report
from jsonModel.reportSchema import reports_schema


report_api = Blueprint('report_api', __name__)

def printRequestData(request):
    current_app.logger.info("request.json : " + str(request.json))
    # multi_dict = request.args
    # for key in multi_dict:
    #     current_app.logger.info(str(multi_dict.get(key)) + " : " +
str(multi_dict.getlist(key)))
    return

def getReport(id):
    return Report.query.get(id)


def getAllReports():
    return Report.query.order_by(Report.isoCreatedAt.desc()).all()


def getList(offset, limit):
```

```
    return Report.query.offset(offset).limit(limit)

def processOptions(request):
    current_app.logger.info("report options")
    response = make_response()
    response.headers.add("Access-Control-Allow-Origin", "*")
    response.headers.add('Access-Control-Allow-Headers', "*")
    response.headers.add('Access-Control-Allow-Methods', "*")
    current_app.logger.info("response : " + str(response))
    return response

def processGet(request):
    current_app.logger.info("get reports")
    reportId = request.args.get("reportId")

    if not reportId:
        list = getAllReports()
    else:
        report = getReport(reportId)
        list = [report]

    current_app.logger.info("report list : " + str(list))
    current_app.logger.info("json report list " +
                            str(json.dumps(reports_schema.dump(list))))
    response = make_response(json.dumps(reports_schema.dump(list)))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response

@report_api.route('/api/report', methods=['GET', 'OPTIONS'])
def run():
    current_app.logger.info("run report")
    printRequestData(request)
    if request.method == "OPTIONS":  # CORS preflight
        return processOptions(request)
    elif request.method == "GET":
        return processGet(request)
    else:
        return
```

api/requestHelper.py

```
from flask import abort
from flask import json
from flask import make_response

def handle_request_error(message, code):
    resultText = "{ \"result\" : \"" + str(message) + "\" }"
```

```
    response = make_response(json.dumps(resultText), code)
    response.headers.add('Access-Control-Allow-Origin', '*')
    abort(response)
```

api/scenarioService.py

```
import logging
from flask import Blueprint
from flask import request
from flask import current_app
from flask import json
from flask import make_response
from sqlalchemy import func
from model.ctrl.ctrlEvent import CtrlEvent
from model.ctrl.scenario import Scenario
from jsonModel.scenarioSchema import scenarios_schema
from jsonModel.teamSchema import teams_schema
from jsonModel.userSchema import users_schema
from jsonModel.network import Network
from model.ctrl.scenarioTeam import ScenarioTeam
from database import db_session
from model.ctrl.team import Team
from sqlalchemy.orm import defer
from sqlalchemy.orm import undefer
from sqlalchemy.orm import load_only

from model.ctrl.teamUser import TeamUser
from model.ctrl.user import User
from model.ctrl.wazuhAlert import WazuhAlert

scenarios_api = Blueprint('scenarios_api', __name__)


@scenarios_api.route('/api/scenarios', methods=['GET'])
def scenarios():
    current_app.logger.info("get scenarios")
    list = Scenario.query.filter(Scenario.status == 'RUNNING').all()
    current_app.logger.info(
        "json list " + str(json.dumps(scenarios_schema.dump(list))))
    response = make_response(json.dumps(scenarios_schema.dump(list)))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response


@scenarios_api.route('/api/networks', methods=['GET'])
def networks():
    current_app.logger.info("get networks")
    current_app.logger.info("request.json : " + str(request.json))
    network = getNetworkByScenarioId(request.json.data.get("scenarioId"))
```

```
    current_app.logger.info("network " + str(network))
    response = make_response(json.dumps(network))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response


@scenarios_api.route('/api/teams', methods=['GET'])
def teams():
    current_app.logger.info("get teams")
    current_app.logger.info("request.json : " + str(request.json))
    teams = getTeamsByScenarioId(request.args.get("scenarioId"))
    current_app.logger.info("teams " + str(teams))
    response = make_response(json.dumps(teams_schema.dump(teams)))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response


@scenarios_api.route('/api/users', methods=['GET'])
def users():
    current_app.logger.info("get users")
    current_app.logger.info("request.json : " + str(request.json))
    users = getUsersByScenarioIdAndTeamId(
        request.args.get("scenarioId"), request.args.get("teamId"))
    current_app.logger.info("users " + str(users))
    response = make_response(json.dumps(users_schema.dump(users)))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response


@scenarios_api.route('/api/event', methods=['GET'])
def event():
    current_app.logger.info("get events")
    current_app.logger.info("request.json : " + str(request.json))
    eventsPerHour = getEventCountPerHour(
        request.args.get("scenarioId"), request.args.get("teamId"),
        request.args.get("userId"),)
    current_app.logger.info("eventsPerHour " + str(eventsPerHour))
    response = make_response(json.dumps(eventsPerHour))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response


def getTeamsByScenarioId(scenarioId):
    query = db_session.query(Team).join(ScenarioTeam)
    query = query.filter(ScenarioTeam.scenarios_id == scenarioId)
    query = query.options(load_only(Team.id, Team.team_id, Team.name))
    return query.all()
```

```
def getUsersByScenarioIdAndTeamId(scenarioId, teamId):
    query =
db_session.query(User).join(TeamUser).join(Team).join(ScenarioTeam)
    query = query.filter(ScenarioTeam.scenarios_id == scenarioId)
    query = query.filter(TeamUser.teams_id == teamId)
    query = query.options(load_only(User.id, User.user_id, User.username))
    return query.all()


def formatEventCountPerHour(tupleList):
    '''
    input : [('2022-05-06T00', 1), ('2022-05-06T23', 17), ('2022-05-06T22',
2)]
    output : [{'time': '2022-05-06T00', 'eventsNo': 1}, {'time': '2022-05-
06T22', 'eventsNo': 2}, {'time': '2022-05-06T23', 'eventsNo': 17}]
    '''
    arr = []
    for a, b in tupleList:
        evdict = {
            "time": a,
            "eventsNo": b
        }
        arr.append(evdict)
    return sorted(arr, key=lambda x: x['time'])


def getEventCountPerHour(scenarioId, teamId, userId):
    '''
    returns
        [{
        'time': '2022-05-06T00', 'eventsNo': 1,
        },
        {
        'time': '2022-05-06T23', 'eventsNo': 17,
        }]
    '''
    timeFormat = '%Y-%m-%dT%H'
    query = db_session.query(func.from_unixtime(
        WazuhAlert.timestamp, timeFormat).label('time'),
func.count(CtrlEvent.id).label('count'))
    query = query.join(WazuhAlert)
    query = query.filter(CtrlEvent.scenarios_id == scenarioId)
    query = query.filter(CtrlEvent.teams_id == teamId)
    if (userId):
        query = query.filter(CtrlEvent.users_id == userId)
    query = query.group_by(func.from_unixtime(
        WazuhAlert.timestamp, timeFormat))
    return formatEventCountPerHour(query.all())
```

```
def getNetworkByScenarioId(scenarioId):
    manageIqResponse = getManageIqResponseSample()
    jsonManageIqResponse = json.loads(manageIqResponse)

    nodesObject = jsonManageIqResponse.data.items
    edges = jsonManageIqResponse.data.relations
    nodes = []
    for key in nodesObject:
        item = nodesObject.get(key)
        item["itemId"] = key
        nodes.append(item)
    return Network(nodes, edges)


def getManageIqResponseSample():
    return '''
    {
    "data": {
        "items": {
            "NetworkManager5": {
                "name": "vmengine Network Manager",
                "kind": "NetworkManager",
                "model": "ManageIQ::Providers::Redhat::NetworkManager",
                "miq_id": 5,
                "key": "NetworkManager5",
                "status": "Valid",
                "display_kind": "Redhat"
            },
            "CloudTenant2": {
                "name": "tenant",
                "kind": "CloudTenant",
                "model":
"ManageIQ::Providers::Openstack::CloudManager::CloudTenant",
                "miq_id": 2,
                "key": "CloudTenant2",
                "status": "OK",
                "display_kind": "CloudTenant"
            },
            "NetworkManager3": {
                "name": "threatrealm1.sec.ouc.ac.cy Network Manager",
                "kind": "NetworkManager",
                "model": "ManageIQ::Providers::Redhat::NetworkManager",
                "miq_id": 3,
                "key": "NetworkManager3",
                "status": "Valid",
                "display_kind": "Redhat"
            },
            "CloudTenant1": {
                "name": "tenant",
```

```
                "kind": "CloudTenant",
                "model":
"ManageIQ::Providers::Openstack::CloudManager::CloudTenant",
                "miq_id": 1,
                "key": "CloudTenant1",
                "status": "OK",
                "display_kind": "CloudTenant"
            },
            "CloudSubnet6": {
                "name": "sub_a",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 6,
                "key": "CloudSubnet6",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "CloudSubnet9": {
                "name": "miq_sub_ovn",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 9,
                "key": "CloudSubnet9",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "CloudSubnet11": {
                "name": "sub",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 11,
                "key": "CloudSubnet11",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "CloudSubnet10": {
                "name": "miq_local",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 10,
                "key": "CloudSubnet10",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
```

```
            "CloudSubnet7": {
                "name": "subnet",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 7,
                "key": "CloudSubnet7",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "CloudSubnet8": {
                "name": "sub_test_miq",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 8,
                "key": "CloudSubnet8",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "CloudSubnet2": {
                "name": "sub1",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 2,
                "key": "CloudSubnet2",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "CloudSubnet1": {
                "name": "sub2",
                "kind": "CloudSubnet",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudSubnet",
                "miq_id": 1,
                "key": "CloudSubnet1",
                "status": "Active",
                "display_kind": "CloudSubnet"
            },
            "NetworkRouter3": {
                "name": "miq_net_ovn",
                "kind": "NetworkRouter",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::NetworkRouter",
                "miq_id": 3,
                "key": "NetworkRouter3",
                "status": "Inactive",
```

```
                "display_kind": "NetworkRouter"
            },
            "NetworkRouter2": {
                "name": "miq_flat_router",
                "kind": "NetworkRouter",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::NetworkRouter",
                "miq_id": 2,
                "key": "NetworkRouter2",
                "status": "Inactive",
                "display_kind": "NetworkRouter"
            },
            "CloudNetwork88": {
                "name": "sub_test",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 88,
                "key": "CloudNetwork88",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork97": {
                "name": "miq_net_ovn",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 97,
                "key": "CloudNetwork97",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork99": {
                "name": "20.1",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 99,
                "key": "CloudNetwork99",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork98": {
                "name": "miq_ovn_flat",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 98,
```

```
                "key": "CloudNetwork98",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork94": {
                "name": "sub2_test",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 94,
                "key": "CloudNetwork94",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork95": {
                "name": "miq_flat_net",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 95,
                "key": "CloudNetwork95",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork89": {
                "name": "br_test3",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 89,
                "key": "CloudNetwork89",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            },
            "CloudNetwork90": {
                "name": "br_test4",
                "kind": "CloudNetwork",
                "model":
"ManageIQ::Providers::Openstack::NetworkManager::CloudNetwork::Private",
                "miq_id": 90,
                "key": "CloudNetwork90",
                "status": "Active",
                "display_kind": "CloudNetwork Private"
            }
        },
        "relations": [
            {
                "source": "NetworkManager5",
```

```
            "target": "CloudTenant2"
        },
        {
            "source": "NetworkManager3",
            "target": "CloudTenant1"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet6"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet9"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet11"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet10"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet7"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet8"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet2"
        },
        {
            "source": "CloudTenant1",
            "target": "CloudSubnet1"
        },
        {
            "source": "CloudTenant1",
            "target": "NetworkRouter3"
        },
        {
            "source": "NetworkRouter3",
            "target": "CloudSubnet6"
        },
        {
            "source": "NetworkRouter3",
```

```
            "target": "CloudSubnet9"
        },
        {
            "source": "NetworkRouter3",
            "target": "CloudSubnet10"
        },
        {
            "source": "CloudTenant1",
            "target": "NetworkRouter2"
        },
        {
            "source": "NetworkRouter2",
            "target": "CloudSubnet8"
        },
        {
            "source": "NetworkManager3",
            "target": "CloudSubnet6"
        },
        {
            "source": "CloudSubnet6",
            "target": "NetworkRouter3"
        },
        {
            "source": "CloudSubnet6",
            "target": "CloudNetwork88"
        },
        {
            "source": "CloudSubnet6",
            "target": "NetworkRouter3"
        },
        {
            "source": "NetworkManager3",
            "target": "CloudSubnet9"
        },
        {
            "source": "CloudSubnet9",
            "target": "NetworkRouter3"
        },
        {
            "source": "CloudSubnet9",
            "target": "CloudNetwork97"
        },
        {
            "source": "CloudSubnet9",
            "target": "NetworkRouter3"
        },
        {
            "source": "NetworkManager3",
```

```
            "target": "CloudSubnet11"
        },
        {
            "source": "CloudSubnet11",
            "target": "CloudNetwork99"
        },
        {
            "source": "NetworkManager3",
            "target": "CloudSubnet10"
        },
        {
            "source": "CloudSubnet10",
            "target": "NetworkRouter3"
        },
        {
            "source": "CloudSubnet10",
            "target": "CloudNetwork98"
        },
        {
            "source": "CloudSubnet10",
            "target": "NetworkRouter3"
        },
        {
            "source": "NetworkManager3",
            "target": "CloudSubnet7"
        },
        {
            "source": "CloudSubnet7",
            "target": "CloudNetwork94"
        },
        {
            "source": "NetworkManager3",
            "target": "CloudSubnet8"
        },
        {
            "source": "CloudSubnet8",
            "target": "NetworkRouter2"
        },
        {
            "source": "CloudSubnet8",
            "target": "CloudNetwork95"
        },
        {
            "source": "CloudSubnet8",
            "target": "NetworkRouter2"
        },
        {
            "source": "NetworkManager3",
```

```
                "target": "CloudSubnet2"
            },
            {
                "source": "CloudSubnet2",
                "target": "CloudNetwork89"
            },
            {
                "source": "NetworkManager3",
                "target": "CloudSubnet1"
            },
            {
                "source": "CloudSubnet1",
                "target": "CloudNetwork90"
            }
        ],
        "kinds": {
            "NetworkRouter": true,
            "CloudSubnet": true,
            "Vm": true,
            "NetworkManager": true,
            "FloatingIp": true,
            "CloudNetwork": true,
            "NetworkPort": true,
            "CloudTenant": true,
            "SecurityGroup": true,
            "LoadBalancer": true,
            "Tag": true,
            "AvailabilityZone": true
        },
        "filter_properties": null,
        "icons": {
            "NetworkRouter": {
                "type": "glyph",
                "class": "pficon pficon-route"
            },
            "CloudSubnet": {
                "type": "glyph",
                "class": "pficon pficon-network"
            },
            "Vm": {
                "type": "glyph",
                "class": "pficon pficon-virtual-machine"
            },
            "FloatingIp": {
                "type": "glyph",
                "class": "ff ff-floating-ip"
            },
            "CloudNetwork": {
```

```
                "type": "glyph",
                "class": "ff ff-cloud-network"
            },
            "NetworkPort": {
                "type": "glyph",
                "class": "ff ff-network-port"
            },
            "CloudTenant": {
                "type": "glyph",
                "class": "pficon pficon-cloud-tenant"
            },
            "SecurityGroup": {
                "type": "glyph",
                "class": "pficon pficon-cloud-security"
            },
            "LoadBalancer": {
                "type": "glyph",
                "class": "ff ff-load-balancer"
            },
            "Tag": {
                "type": "glyph",
                "class": "fa fa-tag"
            },
            "AvailabilityZone": {
                "type": "glyph",
                "class": "pficon pficon-zone"
            },
            "Redhat": {
                "type": "image",
                "icon": "/assets/svg/vendor-redhat_network-
6ef2b0c582ea1f2b1279306a1e1c72f55b4f561a0a524816aa1b0ce9325ee065.svg"
            }
        },
        "settings": null
    }
}
    '''
```

api/scriptService.py

```
import logging
import re
import subprocess
from subprocess import STDOUT, PIPE, TimeoutExpired
import pathlib
from sys import stdout
from typing import List
```

```
from flask import Blueprint
from flask import request
from flask import current_app
from flask import json
from flask import make_response
from datetime import datetime
from model.script.script import Script
from model.script.scriptVariable import ScriptVariable
from model.script.scriptScriptVariable import ScriptScriptVariable
from model.script.scriptExecutor import ScriptExecutor
from jsonModel.script.scriptSchema import scripts_schema
from requestHelper import handle_request_error

scripts_api = Blueprint('scripts_api', __name__)


def printRequestData(request):
    current_app.logger.info("request.json : " + str(request.json))
    # multi_dict = request.args
    # for key in multi_dict:
    #     current_app.logger.info(str(multi_dict.get(key)) + " : " +
str(multi_dict.getlist(key)))
    return


def runNmap(request):
    current_app.logger.info("run nmap")
    data = request.json
    ipRangeStart = data.get("ipRangeStart")
    ipRangeEnd = data.get("ipRangeEnd")
    current_app.logger.info("ipRangeStart : " + str(ipRangeStart))
    current_app.logger.info("ipRangeEnd : " + str(ipRangeEnd))
    result = subprocess.run(['nmap', '0.0.0.0'],
                            capture_output=True, text=True)
    current_app.logger.info("process return code : " + str(result.returncode))
    current_app.logger.info("process output : " + str(result.stdout))
    current_app.logger.info("process error output : " + str(result.stderr))
    if result.returncode == 0:
        return result.stdout
    else:
        return result.stderr


def runScript(data):
    current_app.logger.info("runScript")
    scriptId = data.get("scriptId")
    current_app.logger.info("scriptId : " + str(scriptId))
    variableValues = data.get("variableValues")
    current_app.logger.info("variableValues : " + str(variableValues))
```

```
    if not scriptId:
        handle_request_error("Missing required scriptId", 400)
    script = getScript(scriptId)
    if not script:
        handle_request_error("Invalid script id", 400)
    current_app.logger.info("script : " + str(script))
    scriptVariableValues = getScriptVariableValues(
        variableValues, script.variables)
    current_app.logger.info("scriptVariableValues : " +
                            str(scriptVariableValues))
    return runScriptFile(script, scriptVariableValues)


def getScript(id):
    return Script.query.get(id)


def getAllScripts():
    return Script.query.all()


def getList(offset, limit):
    return Script.query.offset(offset).limit(limit)


def getScriptExecutor(id):
    return ScriptExecutor.query.get(id)


def getScriptVariables(scriptId):
    list = ScriptScriptVariable.query.filter(
        ScriptScriptVariable.scriptId == scriptId).all()
    varIds = [var.variableId for var in list]
    return ScriptVariable.query.filter(ScriptVariable.id.in_(varIds)).all()


def getScriptVariableValues(variableValues, script_ScriptVariables:
List[ScriptScriptVariable]):
    current_app.logger.info("variableValues  " + str(variableValues))
    params = []
    if script_ScriptVariables:
        script_ScriptVariables.sort(key=lambda x: x.orderNo)
        for ss_var in script_ScriptVariables:
            var = ss_var.variable
            variableValue = variableValues.get(var.id)
            current_app.logger.info(
                "param  " + str(var.id) + " : " + str(variableValue))
            # TODO validate value
            if (not variableValue):
```

```
                if (ss_var.required):
                    current_app.logger.info(
                        "Missing required variable value for " +
str(var.name))
                    handle_request_error(
                        "Missing required variable value for " + var.name,
400)
            else:
                current_app.logger.info(
                    "var.type.regexPattern  " + str(var.type.regexPattern))
                acceptedPattern = re.compile(var.type.regexPattern)
                current_app.logger.info(
                    "acceptedPattern " + str(acceptedPattern))
                if (not acceptedPattern.fullmatch(variableValue)):
                    current_app.logger.info(
                        "Invalid variable value for " + str(var.name))
                    handle_request_error(
                        "Invalid variable value for " + var.name, 400)
            params.append(variableValue)
    return params


def runScriptFile(scriptFull, scriptVariableValues):
    current_app.logger.info("runScriptFile scriptFull")
    current_app.logger.info(
        "runParams-bef : " + str([scriptFull.executor.cmd,
scriptFull.location]))
    currentFullPath = str(pathlib.Path().resolve())
    current_app.logger.info("currentFullPath : " + str(currentFullPath))
    scriptFullPath = currentFullPath + "/" + scriptFull.location
    current_app.logger.info("scriptFullPath : " + str(scriptFullPath))
    ouputFileName = str(currentFullPath + "/scripts/output/" + scriptFull.id +
                        "-" + datetime.now().strftime("%Y%m%d%H%M%S%f") + "-"
+ "out.txt")
    current_app.logger.info("ouputFileName : " + str(ouputFileName))
    #runParams = [scriptExecutor.cmd, "-r", str(scriptFullPath) +"/"+
script.location, "-o", ouputFileName]
    runParams = [scriptFull.executor.cmd]
    executorArgs = scriptFull.executor.args
    current_app.logger.info("executorArgs : " + str(executorArgs))
    if (executorArgs):
        for execArg in executorArgs:
            runParams.append(execArg.arg)
    runParams.append(scriptFullPath)
    if scriptVariableValues:
        runParams.extend(scriptVariableValues)
    current_app.logger.info("runParams : " + str(runParams))

    with open(ouputFileName, "x") as out:
```

```
        result = subprocess.run(runParams, text=True,
                                timeout=120, stdout=out, stderr=stdout)
    current_app.logger.info(
        "\n\nprocess return code : " + str(result.returncode))
    current_app.logger.info("process output : " + str(result.stdout))
    current_app.logger.info("process error output : " + str(result.stderr))
    if result.returncode == 0:
        return result.stdout
    else:
        return result.stderr


scriptExecutor = {
    'nmap': runNmap
}


@scripts_api.route('/api/scripts', methods=['GET', 'POST', 'OPTIONS'])
def run():
    current_app.logger.info("run script")
    printRequestData(request)
    if request.method == "OPTIONS":  # CORS preflight
        return processOptions(request)
    elif request.method == "POST":  # The actual request following the
preflight
        return processPost(request)
    elif request.method == "GET":
        return processGet(request)
    else:
        return


def processOptions(request):
    current_app.logger.info("script options")
    response = make_response()
    response.headers.add("Access-Control-Allow-Origin", "*")
    response.headers.add('Access-Control-Allow-Headers', "*")
    response.headers.add('Access-Control-Allow-Methods', "*")
    current_app.logger.info("response : " + str(response))
    return response


def processPost(request):
    current_app.logger.info("post script")
    executorResponse = runScript(request.json)

    resultText = "{ \"result\" : \"" + str(executorResponse) + "\" }"
    current_app.logger.info("response : " + resultText)
    response = make_response(json.dumps(resultText))
```

```
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response


def processGet(request):
    current_app.logger.info("get scripts")
    list = getAllScripts()
    current_app.logger.info("script list : " + str(list))
    current_app.logger.info("json script list " +
                            str(json.dumps(scripts_schema.dump(list))))
    response = make_response(json.dumps(scripts_schema.dump(list)))
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response
```

api/scriptTriggerConditionOperator.py

```
from enum import Enum, unique

@unique
class ScriptTriggerConditionOperator(Enum):
    EQUAL = '='
    NOT_EQUAL = '!-'
    GREATER_THAN = '>'
    GREATER_THAN_OR_EQUAL = '>='
    LESS_THAN = '<'
    LESS_THAN_OR_EQUAL = '<='
```

api/scriptTriggerConditionPropertyId.py

```
from enum import Enum, unique

@unique
class ScriptTriggerConditionPropertyId(Enum):
    SURICATA_RULE_ID = 1
    SURICATA_SOURCE_IP = 2
    SURICATA_DESTINATION_IP = 3
    SURICATA_DESTINATION_PORT = 4
    SURICATA_HOST_NAME = 5
    WAZUH_RULE_ID = 6
    WAZUH_RULE_LEVEL = 7
    WAZUH_RULE_AGENT_ID = 8
    WAZUH_RULE_AGENT_HOST_NAME = 9
    WAZUH_RULE_AGENT_IP = 10
```

api/scriptTriggerEvent.py

```
from enum import Enum, unique
```

```
@unique
class ScriptTriggerEvent(Enum):
    WAZUH_ALERT = 1
    SURICATA_ALERT = 2
```

api/scriptTriggerEventStatus.py

```
from enum import Enum, unique

@unique
class ScriptTriggerEventStatus(Enum):
    PENDING = 1
    PROCESSED = 2
```

api/scriptTriggerService.py

```
import logging
import re
import subprocess
from subprocess import STDOUT, PIPE, TimeoutExpired
import pathlib
from sys import stdout
from flask import Blueprint
from flask import request
from flask import current_app
from flask import json
from flask import make_response
from datetime import datetime
from appDatabase import app_db_session
from model.ctrl.suricataAlert import SuricataAlert
from model.ctrl.wazuhAlert import WazuhAlert
from model.script import scriptTrigger
from model.script.ctrlTriggerEvent import CtrlTriggerEvent
from model.script.script import Script
from model.script.scriptTrigger import ScriptTrigger
from model.script.scriptVariable import ScriptVariable
from model.script.scriptScriptVariable import ScriptScriptVariable
from model.script.scriptExecutor import ScriptExecutor
from jsonModel.script.scriptSchema import scripts_schema
from requestHelper import handle_request_error
from scriptService import runScriptFile
from scriptTriggerConditionOperator import ScriptTriggerConditionOperator
from scriptTriggerConditionPropertyId import ScriptTriggerConditionPropertyId
from scriptTriggerEvent import ScriptTriggerEvent
from scriptTriggerEventStatus import ScriptTriggerEventStatus
from scriptTriggerStatus import ScriptTriggerStatus
from scriptTriggerVariableValueType import ScriptTriggerVariableValueType

script_trigger_api = Blueprint('script_trigger_api', __name__)
```

```
def getScriptTrigger(id):
    return ScriptTrigger.query.get(id)


def getList(offset, limit):
    return ScriptTrigger.query.offset(offset).limit(limit)


def getAllRunningForEventType(eventType):
    return ScriptTrigger.query.filter(ScriptTrigger.status ==
ScriptTriggerStatus.RUNNING.name, ScriptTrigger.triggerValue ==
eventType).all()


def baseConditionOk(operator, conditionValue, actualValue):
    if (ScriptTriggerConditionOperator.EQUAL.name == operator):
        return actualValue == conditionValue
    elif (ScriptTriggerConditionOperator.NOT_EQUAL.name == operator):
        return actualValue != conditionValue
    elif (ScriptTriggerConditionOperator.GREATER_THAN.name == operator):
        return actualValue > conditionValue
    elif (ScriptTriggerConditionOperator.GREATER_THAN_OR_EQUAL.name ==
operator):
        return actualValue >= conditionValue
    elif (ScriptTriggerConditionOperator.LESS_THAN.name == operator):
        return actualValue < conditionValue
    elif (ScriptTriggerConditionOperator.LESS_THAN_OR_EQUAL.name == operator):
        return actualValue <= conditionValue
    else:
        return False


def alertRuleIdactualValueGetter(alert):
    return alert.rule.id


def alertAgentIdValueGetter(alert):
    return alert.agent_id


def alertAgentHostnameValueGetter(alert):
    return alert.agent_hostname


def alertAgentIpValueGetter(alert):
    return alert.agent_ip
```

```
def wazuhAlertRuleLevelValueGetter(alert):
    return alert.rule.level


def alertSourceIpValueGetter(alert):
    return alert.src_ip


def alertDestinationIpValueGetter(alert):
    return alert.dest_ip


def alertDestinationPortValueGetter(alert):
    return alert.dest_port


def alertHostnameValueGetter(alert):
    return alert.hostname


conditionActualValueGetterByPropertyId = {
    ScriptTriggerConditionPropertyId.SURICATA_RULE_ID.name:
alertRuleIdactualValueGetter,
    ScriptTriggerConditionPropertyId.SURICATA_SOURCE_IP.name:
alertSourceIpValueGetter,
    ScriptTriggerConditionPropertyId.SURICATA_DESTINATION_IP.name:
alertDestinationIpValueGetter,
    ScriptTriggerConditionPropertyId.SURICATA_DESTINATION_PORT.name:
alertDestinationPortValueGetter,
    ScriptTriggerConditionPropertyId.SURICATA_HOST_NAME.name:
alertHostnameValueGetter,
    ScriptTriggerConditionPropertyId.WAZUH_RULE_ID.name:
alertRuleIdactualValueGetter,
    ScriptTriggerConditionPropertyId.WAZUH_RULE_LEVEL.name:
wazuhAlertRuleLevelValueGetter,
    ScriptTriggerConditionPropertyId.WAZUH_RULE_AGENT_ID.name:
alertAgentIdValueGetter,
    ScriptTriggerConditionPropertyId.WAZUH_RULE_AGENT_HOST_NAME.name:
alertAgentHostnameValueGetter,
    ScriptTriggerConditionPropertyId.WAZUH_RULE_AGENT_IP.name:
alertAgentIpValueGetter
}


def actualConditionValueGetter(conditionProperty, alert):
    return conditionActualValueGetterByPropertyId[conditionProperty](alert)


def conditionsOk(conditions, alert):
    if (not cond):
```

```
        return True

    for cond in conditions:
        condOk = baseConditionOk(
            cond.operator, cond.value,
actualConditionValueGetter(cond.property, alert))
        if (not condOk):
            return False
    return True


def runEventTrigger(scriptTrigger):
    current_app.logger.info("request.json : " + str(request.json))
    # multi_dict = request.args
    # for key in multi_dict:
    #     current_app.logger.info(str(multi_dict.get(key)) + " : " +
str(multi_dict.getlist(key)))
    return


def createEvent(event: CtrlTriggerEvent):
    app_db_session.add(event)
    app_db_session.commit()
    return event


def deleteEvent(event: CtrlTriggerEvent):
    CtrlTriggerEvent.query.filter(CtrlTriggerEvent.id == event.id).delete()
    app_db_session.commit()


def updateEvent(event: CtrlTriggerEvent, status: ScriptTriggerEventStatus,
triggered: list):
    event.status = status.name
    if (triggered):
        string_triggered = [str(int) for int in triggered]
        event.triggered = ",".join(string_triggered)
    app_db_session.commit()
    return event


def getWazuhAlert(id):
    WazuhAlert.query.get(id)


def getSuricataAlert(id):
    SuricataAlert.query.get(id)


alertGetterByEventType = {
```

```
    ScriptTriggerEvent.SURICATA_ALERT.name: getSuricataAlert,
    ScriptTriggerEvent.WAZUH_ALERT.name: getWazuhAlert
}


def getAlert(event: CtrlTriggerEvent):
    return alertGetterByEventType[event.eventType](event.eventId)

def getScriptVariableValues(triggerVariableValues, alert):
    params = []
    if triggerVariableValues:
        triggerVariableValues.sort(key=lambda x : x.variable.order)
        for tVar in triggerVariableValues:
            paramVal = tVar.value
            if (ScriptTriggerVariableValueType.DYNAMIC.name ==
tVar.valueType):
                paramVal = actualConditionValueGetter(tVar.value, alert)
            params.append(paramVal)
    return params

def processEvent(event: CtrlTriggerEvent):
    runningTriggers = getAllRunningForEventType(event.eventType)
    if (not runningTriggers):
        updateEvent(event, ScriptTriggerEventStatus.PROCESSED, None)
        return

    alert = getAlert(event)
    triggered = []
    for trigger in runningTriggers:
        condsOk = conditionsOk(trigger.conditions, alert)
        if (not condsOk):
            continue
        runScriptFile(trigger.script,
getScriptVariableValues(trigger.scriptVariables, alert))
        triggered.append(trigger.script.id)
    updateEvent(event, ScriptTriggerEventStatus.PROCESSED, triggered)


@script_trigger_api.route('/api/scriptTrigger', methods=['GET', 'POST',
'OPTIONS'])
def run():
    current_app.logger.info("run script")
    current_app.logger.info("request.json : " + str(request.json))
    if request.method == "OPTIONS":  # CORS preflight
        return processOptions(request)
    elif request.method == "POST":  # The actual request following the
preflight
        return processPost(request)
    elif request.method == "GET":
```

```
        return processGet(request)
    else:
        return


def processOptions(request):
    current_app.logger.info("script options")
    response = make_response()
    response.headers.add("Access-Control-Allow-Origin", "*")
    response.headers.add('Access-Control-Allow-Headers', "*")
    response.headers.add('Access-Control-Allow-Methods', "*")
    current_app.logger.info("response : " + str(response))
    return response


def processPost(request):
    current_app.logger.info("post script")
    # executorResponse = runScript(request)

    # resultText = "{ \"result\" : \"" + str(executorResponse) + "\" }"
    # current_app.logger.info("response : " + resultText)
    # response = make_response(json.dumps(resultText))
    response = make_response()
    response.headers.add('Access-Control-Allow-Origin', '*')
    current_app.logger.info("response : " + str(response))
    return response


def processGet(request):
    current_app.logger.info("get scripts")
    # list = getAllScripts()
    # current_app.logger.info("script list : " + str(list))
    # current_app.logger.info("json script list " +
str(json.dumps(scripts_schema.dump(list))))
    # response = make_response(json.dumps(scripts_schema.dump(list)))
    response = make_response()
    response.headers.add('Access-Control-Allow-Origin', '*')
    current_app.logger.info("response : " + str(response))
    return response
```

api/scriptTriggerStatus.py

```
from enum import Enum, unique

@unique
class ScriptTriggerStatus(Enum):
    # For TIME triggered
    PENDING = 1
```

```
    EXECUTED = 2

    # For EVENT triggered
    RUNNING = 3
    STOPPED = 4
```

api/scriptTriggerType.py

```
from enum import Enum, unique

@unique
class ScriptTriggerType(Enum):
    EVENT = 1
    TIME = 2
```

api/scriptTriggerVariableValueType.py

```
from enum import Enum, unique

@unique
class ScriptTriggerVariableValueType(Enum):
    ACTUAL = 1,
    # value must be one of ScriptTriggerConditionProperty.id
    DYNAMIC = 2
```

api/model/ctrl/ctrlEvent.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.wazuhRule import WazuhRule


class CtrlEvent(App_db_base):
    __tablename__ = 'CtrlEvent'
    id = Column(Integer, primary_key=True)
    scenarios_id = Column(String(45), ForeignKey('scenarios.id'))
    teams_id = Column(String(45), ForeignKey('teams.id'))
    users_id = Column(String(45), ForeignKey('users.id'))
    wazuh_alert_id = Column(Integer, ForeignKey('wazuh_alert.id'))
    # rule = relationship('WazuhRule', uselist=False)

    def __init__(self, id, scenarios_id=None, teams_id=None, users_id=None,
wazuh_alert_id=None):
        self.id = id
        self.scenarios_id = scenarios_id
        self.teams_id = teams_id
        self.users_id = users_id
```

```
        self.wazuh_alert_id = wazuh_alert_id

    def __repr__(self):
        repr = f'<CtrlEvent id={self.id}, scenarios_id={self.scenarios_id}, '
\
            f'users_id={self.users_id}, wazuh_alert_id={self.wazuh_alert_id}>'
        return repr
```

api/model/ctrl/report.py

```
from sqlalchemy import Column, Integer, String
from database import Base


class Report(Base):
    __tablename__ = 'Report'
    id = Column(Integer, primary_key=True)
    name = Column(String(255))
    description = Column(String(1000))
    isoCreatedAt = Column(String(19))

    def __init__(self, id=None, name=None, description=None,
isoCreatedAt=None):
        self.id = id
        self.name = name
        self.description = description
        self.isoCreatedAt = isoCreatedAt

    def __repr__(self):
        repr = f'<Report id={self.id}, name={self.name}, ' \
            f'description={self.description},
isoCreatedAt={self.isoCreatedAt}>'
        return repr
```

api/model/ctrl/scenario.py

```
from sqlalchemy import Column, Integer, String
from database import Base


class Scenario(Base):
    __tablename__ = 'scenarios'
    id = Column(Integer, primary_key=True)
    scenario_id = Column(String(45))
    title = Column(String(255))
    focus_area = Column(String(45))
    level = Column(String(45))
```

```
    description = Column(String(1000))
    link = Column(String(1000))
    status = Column(String(45))

    def __init__(self, scenario_id=None, title=None, focus_area=None,
level=None, description=None, link=None, status=None):
        self.scenario_id = scenario_id
        self.title = title
        self.focus_area = focus_area
        self.level = level
        self.description = description
        self.link = link
        self.status = status

    def __repr__(self):
        repr = f'<Scenarios id={self.id}, scenario_id={self.scenario_id}, ' \
            f'title={self.title}, focus_area={self.focus_area}, ' \
            f'level={self.level}, description={self.description} ' \
            f'link={self.link}, status={self.status}>'
        return repr
```

api/model/ctrl/scenarioRules.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.scenario import Scenario
from model.ctrl.suricataRule import SuricataRule
from model.ctrl.wazuhRule import WazuhRule


class ScenarioRules(App_db_base):
    __tablename__ = 'scenarios_rules'
    id = Column(Integer, primary_key=True)
    ids_hids = Column(String(45))
    points = Column(Integer)
    scenarios_id = Column(Integer, ForeignKey('scenarios.id'))
    suricata_rules_id = Column(Integer, ForeignKey('suricata_rules.id'))
    wazuh_rules_id = Column(Integer, ForeignKey('wazuh_rules.id'))
    suricata_rules = relationship('SuricataRule', uselist=False)
    wazuh_rules = relationship('WazuhRule', uselist=False)

    def __init__(self, ids_hids=None, points=None, scenarios_id=None,
suricata_rules_id=None, wazuh_rules_id=None):
        self.ids_hids = ids_hids
        self.points = points
        self.scenarios_id = scenarios_id
```

```
        self.suricata_rules_id = suricata_rules_id
        self.wazuh_rules_id = wazuh_rules_id

    def __repr__(self):
        repr = f'<ScenarioRules id={self.id}, ids_hids={self.ids_hids}, ' \
            f'points={self.points}, scenarios_id={self.scenarios_id}, ' \
            f'suricata_rules_id={self.suricata_rules_id},
wazuh_rules_id={self.wazuh_rules_id}, ' \
            f'suricata_rules={self.suricata_rules},
wazuh_rules={self.wazuh_rules}>'
        return repr
```

api/model/ctrl/scenarioTeam.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database import Base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.team import Team
from model.ctrl.scenario import Scenario


class ScenarioTeam(Base):
    __tablename__ = 'scenarios_teams'
    id = Column(Integer, primary_key=True)
    teams_id = Column(Integer, ForeignKey('teams.id'))
    scenarios_id = Column(Integer, ForeignKey('scenarios.id'))
    # team = relationship('Team', uselist=False)
    # scenario = relationship('Scenario', uselist=False)

    def __init__(self, teams_id=None, scenarios_id=None):
        self.teams_id = teams_id
        self.scenarios_id = scenarios_id

    def __repr__(self):
        repr = f'<ScenarioTeam id={self.id}, teams_id={self.teams_id}, ' \
            f'scenarios_id={self.scenarios_id}>'
        return repr
```

api/model/ctrl/scenarioUser.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database import Base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.scenario import Scenario
from model.ctrl.user import User
```

```
class ScenarioUser(Base):
    __tablename__ = 'scenarios_users'
    id = Column(Integer, primary_key=True)
    users_id = Column(Integer, ForeignKey('users.id'))
    scenarios_id = Column(Integer, ForeignKey('scenarios.id'))
    user = relationship('users', uselist=False)
    scenario = relationship('scenarios', uselist=False)

    def __init__(self, users_id=None, scenarios_id=None):
        self.users_id = users_id
        self.scenarios_id = scenarios_id

    def __repr__(self):
        repr = f'<ScenarioUser id={self.id}, users_id={self.users_id}, ' \
            f'scenarios_id={self.scenarios_id}, user={self.user}, ' \
            f'scenario={self.scenario}>'
        return repr
```

api/model/ctrl/suricataAlert.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.suricataRule import SuricataRule


class SuricataAlert(App_db_base):
    __tablename__ = 'suricata_alert'
    id = Column(Integer, primary_key=True)
    timestamp = Column(Integer)
    src_ip = Column(String(45))
    dest_ip = Column(String(45))
    dest_port = Column(Integer)
    hostname = Column(String(45))
    suricata_rules_id = Column(Integer, ForeignKey('suricata_rules.id'))
    rule = relationship('SuricataRule', uselist=False)

    def __init__(self, timestamp=None, src_ip=None, dest_ip=None,
dest_port=None, hostname=None, suricata_rules_id=None):
        self.timestamp = timestamp
        self.src_ip = src_ip
        self.dest_ip = dest_ip
        self.dest_port = dest_port
        self.hostname = hostname
        self.suricata_rules_id = suricata_rules_id
```

```
    def __repr__(self):
        repr = f'<ScenariosRules id={self.id}, timestamp={self.timestamp}, ' \
            f'src_ip={self.src_ip}, dest_ip={self.dest_ip}, ' \
            f'dest_port={self.dest_port}, hostname={self.hostname}, ' \
            f'suricata_rules_id={self.suricata_rules_id}, rule={self.rule}>'
        return repr
```

api/model/ctrl/suricataRule.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base


class SuricataRule(App_db_base):
    __tablename__ = 'suricata_rules'
    id = Column(Integer, primary_key=True)
    signature_id = Column(Integer)
    signature = Column(String(255))
    severity = Column(String(45))

    def __init__(self, id, signature_id=None, signature=None, severity=None):
        self.id = id
        self.signature_id = signature_id
        self.signature = signature
        self.severity = severity

    def __repr__(self):
        repr = f'<SuricataRule id={self.id}, signature_id={self.signature_id},
' \
            f'signature={self.signature}, severity={self.severity}>'
        return repr
```

api/model/ctrl/team.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database import Base


class Team(Base):
    __tablename__ = 'teams'
    id = Column(Integer, primary_key=True)
    team_id = Column(String(45))
    name = Column(String(45))
```

```
    def __init__(self, team_id=None, name=None):
        self.team_id = team_id
        self.name = name

    def __repr__(self):
        repr = f'<Team id={self.id}, team_id={self.team_id}, ' \
            f'name={self.name}>'
        return repr
```

api/model/ctrl/teamUser.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database import Base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.team import Team
from model.ctrl.user import User


class TeamUser(Base):
    __tablename__ = 'teams_users'
    id = Column(Integer, primary_key=True)
    teams_id = Column(Integer, ForeignKey('teams.id'))
    users_id = Column(Integer, ForeignKey('users.id'))
    # team = relationship('teams', uselist=False)
    # user = relationship('users', uselist=False)

    def __init__(self, users_id=None, scenarios_id=None):
        self.users_id = users_id
        self.scenarios_id = scenarios_id

    def __repr__(self):
        repr = f'<TeamUser id={self.id}, teams_id={self.teams_id}, ' \
            f'users_id={self.users_id}>'
        return repr
```

api/model/ctrl/user.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database import Base


class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    user_id = Column(String(45))
```

```
    user_ip = Column(String(45))
    hostname = Column(String(45))
    username = Column(String(45))
    email = Column(String(45))

    def __init__(self, user_id=None, user_ip=None, hostname=None,
username=None, email=None):
        self.user_id = user_id
        self.user_ip = user_ip
        self.hostname = hostname
        self.username = username
        self.email = email

    def __repr__(self):
        repr = f'<User id={self.id}, user_id={self.user_id}, ' \
            f'user_ip={self.user_ip}, hostname={self.hostname}, ' \
            f'username={self.username}, email={self.email}>'
        return repr
```

api/model/ctrl/wazuhAlert.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.ctrl.wazuhRule import WazuhRule


class WazuhAlert(App_db_base):
    __tablename__ = 'wazuh_alert'
    id = Column(Integer, primary_key=True)
    timestamp = Column(Integer)
    agent_id = Column(Integer)
    agent_hostname = Column(String(255))
    agent_ip = Column(String(45))
    wazuh_rules_id = Column(Integer, ForeignKey('wazuh_rules.id'))
    rule = relationship('WazuhRule', uselist=False)

    def __init__(self, id, timestamp=None, agent_id=None, agent_hostname=None,
agent_ip=None, wazuh_rules_id=None):
        self.id = id
        self.timestamp = timestamp
        self.agent_id = agent_id
        self.agent_hostname = agent_hostname
        self.agent_ip = agent_ip
        self.wazuh_rules_id = wazuh_rules_id

    def __repr__(self):
```

```
        repr = f'<WazuhAlert id={self.id}, timestamp={self.timestamp}, ' \
            f'agent_id={self.agent_id}, agent_hostname={self.agent_hostname},
' \
            f'agent_ip={self.agent_ip}, wazuh_rules_id={self.wazuh_rules_id},
' \
            f'rule={self.rule}>'
        return repr
```

api/model/ctrl/wazuhRule.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base


class WazuhRule(App_db_base):
    __tablename__ = 'wazuh_rules'
    id = Column(Integer, primary_key=True)
    rule_id = Column(Integer)
    level = Column(String(45))
    description = Column(String(255))

    def __init__(self, id, rule_id=None, level=None, description=None):
        self.id = id
        self.rule_id = rule_id
        self.level = level
        self.description = description

    def __repr__(self):
        repr = f'<ScenariosRules id={self.id}, rule_id={self.rule_id}, ' \
            f'level={self.level}, description={self.description}>'
        return repr
```

api/model/script/ctrlTriggerEvent.py

```
from sqlalchemy import Column, Integer, String
from appDatabase import App_db_base


class CtrlTriggerEvent(App_db_base):
    __tablename__ = 'CtrlTriggerEvent'
    id = Column(Integer, primary_key=True)
    # ScriptTriggerEvent.name
    eventType = Column(String(100))
    eventId = Column(Integer)
    status = Column(String(50))
    triggered = Column(String(255))
```

```
    # Event not persistent. Will load later based on type
    event = None

    def __init__(self, eventType, eventId, status, triggered=None,
event=None):
        self.eventType = eventType
        self.eventId = eventId
        self.status = status
        self.triggered = triggered
        self.event = event

    def __repr__(self):
        repr = f'<CtrlEvent id={self.id}, eventType={self.eventType}, ' \
            f'eventId={self.eventId}, status={self.status}, '\
            f'triggered={self.triggered}, event={self.event}>'
        return repr
```

api/model/script/operatingSystem.py

```
from sqlalchemy import Column, String
from appDatabase import App_db_base


class OperatingSystem(App_db_base):
    __tablename__ = 'OperatingSystem'
    id = Column(String(45), primary_key=True)
    name = Column(String(100), unique=True)

    def __init__(self, name=None):
        self.name = name

    def __repr__(self):
        return f'<OperatingSystem id={self.id}, name={self.name}>'
```

api/model/script/script.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.script.scriptExecutor import ScriptExecutor
from model.script.scriptCategory import ScriptCategory
from model.script.scriptScriptVariable import ScriptScriptVariable


class Script(App_db_base):
    __tablename__ = 'Script'
```

```
    id = Column(String(50), primary_key=True)
    name = Column(String(50), unique=True)
    description = Column(String(1000))
    location = Column(String(1000))
    executorId = Column(String(45), ForeignKey('ScriptExecutor.id'))
    categoryId = Column(String(45), ForeignKey('ScriptCategory.id'))
    executor = relationship('ScriptExecutor', uselist=False)
    category = relationship('ScriptCategory', uselist=False)
    variables = relationship('ScriptScriptVariable')

    def __init__(self, name=None, description=None, location=None,
executorId=None, categoryId=None):
        self.name = name
        self.description = description
        self.location = location
        self.executorId = executorId
        self.categoryId = categoryId

    def __repr__(self):
        repr = f'<Script id={self.id}, name={self.name}, ' \
            f'description={self.description}, location={self.location}, ' \
            f'executorId={self.executorId}, executor={self.executor}, ' \
            f'categoryId={self.categoryId}, category={self.category}, ' \
            f'variables={self.variables}>'
        return repr
```

api/model/script/scriptCategory.py

```
from sqlalchemy import Column, String
from appDatabase import App_db_base


class ScriptCategory(App_db_base):
    __tablename__ = 'ScriptCategory'
    id = Column(String(45), primary_key=True)
    name = Column(String(100), unique=True)

    def __init__(self, name=None):
        self.name = name

    def __repr__(self):
        return f'<ScriptCategory id={self.id}, name={self.name}>'
```

api/model/script/scriptExecutor.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
```

```
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.script.operatingSystem import OperatingSystem
from model.script.scriptExecutorArg import ScriptExecutorArg


class ScriptExecutor(App_db_base):
    __tablename__ = 'ScriptExecutor'
    id = Column(String(45), primary_key=True)
    name = Column(String(100), unique=True)
    cmd = Column(String(1000))
    hostOSId = Column(String(45), ForeignKey('OperatingSystem.id'))
    hostOS = relationship('OperatingSystem')
    args = relationship('ScriptExecutorArg')

    def __init__(self, name=None, cmd=None, hostOSId=None):
        self.name = name
        self.cmd = cmd
        self.hostOSId = hostOSId

    def __repr__(self):
        repr = f'<ScriptExecutor id={self.id}, name={self.name}, ' \
            f'cmd={self.cmd}, hostOSId={self.hostOSId}, ' \
            f'hostOS={self.hostOS}, args={self.args}>'
        return repr
```

api/model/script/scriptExecutorArg.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base


class ScriptExecutorArg(App_db_base):
    __tablename__ = 'ScriptExecutorArg'
    id = Column(String(50), primary_key=True)
    executorId = Column(String(45), ForeignKey('ScriptExecutor.id'))
    arg = Column(String(10))
    value = Column(String(100))

    def __init__(self, executorId=None, arg=None, value=None):
        self.executorId = executorId
        self.arg = arg
        self.value = value

    def __repr__(self):
        return f'<ScriptExecutorArg id={self.id},
executorId={self.executorId}, arg={self.arg}, value={self.value}>'
```

api/model/script/scriptScriptVariable.py

```
from sqlalchemy import Column, Integer, String, Boolean, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.script.scriptVariable import ScriptVariable


class ScriptScriptVariable(App_db_base):
    __tablename__ = 'Script_ScriptVariable'
    scriptId = Column(String(50), ForeignKey('Script.id'), primary_key=True)
    variableId = Column(String(50), ForeignKey(
        'ScriptVariable.id'), primary_key=True)
    orderNo = Column(Integer)
    required = Column(Boolean)
    variable = relationship('ScriptVariable', uselist=False)

    def __init__(self, scriptId=None, variableId=None, orderNo=None,
required=None):
        self.scriptId = scriptId
        self.variableId = variableId
        self.orderNo = orderNo
        self.required = required

    def __repr__(self):
        repr = f'<Script_ScriptVariable scriptId={self.scriptId},
variableId={self.variableId}, ' \
            f'orderNo={self.orderNo} variable={self.variable},
required={self.required}>'
        return repr
```

api/model/script/scriptTrigger.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.script.script import Script
from model.script.scriptTriggerCondition import ScriptTriggerCondition
from model.script.scriptTriggerVariable import ScriptTriggerVariable


class ScriptTrigger(App_db_base):
    __tablename__ = 'ScriptTrigger'
    id = Column(String(50), primary_key=True)
```

```
    name = Column(String(50), unique=True)
    description = Column(String(1000))
    #EVENT or TIME
    type = Column(String(20))
    # event for EVENT, datetime for TIME
    triggerValue = Column(String(50))
    scriptId = Column(String(50), ForeignKey('Script.id'))
    status = Column(String(50))
    conditions = relationship('ScriptTriggerCondition')
    script = relationship('Script', uselist=False)
    scriptVariables = relationship('ScriptTriggerVariable')

    def __init__(self, name=None, description=None, type=None,
triggerValue=None, scriptId=None, status=None):
        self.name = name
        self.description = description
        self.type = type
        self.triggerValue = triggerValue
        self.scriptId = scriptId
        self.status = status

    def __repr__(self):
        return f'<ScriptTrigger id={self.id}, name={self.name}, ' \
            f'description={self.description}, type={self.type}, ' \
            f'triggerValue={self.triggerValue}, scriptId={self.scriptId}, ' \
            f'status={self.status}, conditions={self.conditions}, ' \
            f'script={self.script}, scriptVariables={self.scriptVariables}>'
```

api/model/script/scriptTriggerCondition.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.script.scriptTriggerConditionProperty import
ScriptTriggerConditionProperty


class ScriptTriggerCondition(App_db_base):
    __tablename__ = 'ScriptTriggerCondition'
    id = Column(String(50), primary_key=True)
    triggerId = Column(String(100), ForeignKey('ScriptTrigger.id'))
    propertyId = Column(String(100), ForeignKey(
        'ScriptTriggerConditionProperty.id'))
    operator = Column(String(30))
    value = Column(String(100))
    property = relationship('ScriptTriggerConditionProperty')
```

```
    def __init__(self, triggerId=None, propertyId=None, operator=None,
value=None):
        self.triggerId = triggerId
        self.propertyId = propertyId
        self.operator = operator
        self.value = value

    def __repr__(self):
        return f'<ScriptTriggerCondition id={self.id},
triggerId={self.triggerId}, ' \
            f'propertyId={self.propertyId}, operator={self.operator}, ' \
            f'value={self.value}, property={self.property}>'
```

api/model/script/scriptTriggerConditionProperty.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
from model.script.scriptVariableType import ScriptVariableType


class ScriptTriggerConditionProperty(App_db_base):
    __tablename__ = 'ScriptTriggerConditionProperty'
    id = Column(String(50), primary_key=True)
    typeId = Column(String(45), ForeignKey('ScriptVariableType.id'))
    type = relationship('ScriptVariableType')

    def __init__(self, typeId=None):
        self.typeId = typeId

    def __repr__(self):
        return f'<ScriptTriggerConditionProperty id={self.id},
typeId={self.typeId}, type={self.type}>'
```

api/model/script/scriptTriggerVariable.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
from model.script.scriptVariable import ScriptVariable


class ScriptTriggerVariable(App_db_base):
    __tablename__ = 'ScriptTriggerVariable'
    triggerId = Column(String(100), ForeignKey(
        'ScriptTrigger.id'), primary_key=True)
    variableId = Column(String(100), ForeignKey(
```

```
        'ScriptVariable.id'), primary_key=True)
    # DYNAMIC or ACTUAL
    valueType = Column(String(30))
    value = Column(String(100))
    variable = relationship('ScriptVariable')

    def __init__(self, triggerId=None, variableId=None, valueType=None,
value=None):
        self.triggerId = triggerId
        self.variableId = variableId
        self.valueType = valueType
        self.value = value

    def __repr__(self):
        return f'<ScriptTriggerVariable triggerId={self.triggerId}, ' \
            f'variableId={self.variableId}, valueType={self.valueType}, ' \
            f'variable={self.variable}, value={self.value}>'
```

api/model/script/scriptVariable.py

```
from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.orm import relationship
from appDatabase import App_db_base
# Need following import for sqlalchemy to load in correct order
from model.script.scriptVariableType import ScriptVariableType


class ScriptVariable(App_db_base):
    __tablename__ = 'ScriptVariable'
    id = Column(String(50), primary_key=True)
    name = Column(String(50), unique=True)
    typeId = Column(String(1000), ForeignKey('ScriptVariableType.id'))
    type = relationship('ScriptVariableType')

    def __init__(self, name=None, typeId=None):
        self.name = name
        self.typeId = typeId

    def __repr__(self):
        repr = f'<ScriptVariable id={self.id}, name={self.name}, ' \
            f'typeId={self.typeId}, type={self.type}>'
        return repr
```

api/model/script/scriptVariableType.py

```
from sqlalchemy import Column, String
from appDatabase import App_db_base
```

```
class ScriptVariableType(App_db_base):
    __tablename__ = 'ScriptVariableType'
    id = Column(String(50), primary_key=True)
    name = Column(String(50), unique=True)
    regexPattern = Column(String(200))

    def __init__(self, name=None, regexPattern=None):
        self.name = name
        self.regexPattern = regexPattern

    def __repr__(self):
        repr = f'<ScriptVariableType id={self.id}, name={self.name}, ' \
            f'regexPattern={self.regexPattern}>'
        return repr
```

api/mode/calendarEvent.py

```
from sqlalchemy import Column, String
from sqlalchemy.sql.sqltypes import Boolean
from database import Base

class CalendarEvent(Base):
    __tablename__ = 'CalendarEvent'
    id = Column(String(50), primary_key=True)
    name = Column(String(50))
    isoDateFrom = Column(String(19))
    isoDateTo = Column(String(19))
    allDay = Column(Boolean)

    def __init__(self, name=None, isoDateFrom=None, isoDateTo=None,
allDay=None):
        self.name = name
        self.isoDateFrom = isoDateFrom
        self.isoDateTo = isoDateTo
        self.allDay = allDay

    def __repr__(self):
        return '<CalendarEvent %r>' % (self.name)
```

api/jsonModel/calendarEventSchema.py

```
from marshmallow_sqlalchemy import SQLAlchemyAutoSchema
from model.calendarEvent import CalendarEvent

class CalendarEventSchema(SQLAlchemyAutoSchema):
```

```
    class Meta:
        model = CalendarEvent

calendar_event_schema = CalendarEventSchema()
calendar_events_schema = CalendarEventSchema(many=True)
```

api/jsonModel/network.py

```
class Network():
    def __init__(self, nodes=None, edges=None):
        self.nodes = nodes
        self.edges = edges

    def __repr__(self):
        return f'<Network nodes={self.nodes}, edges={self.edges}>'
```

api/jsonModel/reportSchema.py

```
from marshmallow_sqlalchemy import SQLAlchemyAutoSchema
from model.ctrl.report import Report

class ReportSchema(SQLAlchemyAutoSchema):
    class Meta:
        model = Report

report_schema = ReportSchema()
reports_schema = ReportSchema(many=True)
```

api/jsonModel/scenarioSchema.py

```
from marshmallow_sqlalchemy import SQLAlchemyAutoSchema
from model.ctrl.scenario import Scenario

class ScenarioSchema(SQLAlchemyAutoSchema):
    class Meta:
        model = Scenario

scenario_schema = ScenarioSchema()
scenarios_schema = ScenarioSchema(many=True)
```

api/jsonModel/teamSchema.py

```
from marshmallow_sqlalchemy import SQLAlchemyAutoSchema
from model.ctrl.scenario import Scenario
from model.ctrl.team import Team
```

```
class TeamSchema(SQLAlchemyAutoSchema):
    class Meta:
        model = Team

team_schema = TeamSchema()
teams_schema = TeamSchema(many=True)
```

api/jsonModel/userSchema.py

```
from marshmallow_sqlalchemy import SQLAlchemyAutoSchema
from model.ctrl.scenario import Scenario
from model.ctrl.user import User

class UserSchema(SQLAlchemyAutoSchema):
    class Meta:
        model = User

user_schema = UserSchema()
users_schema = UserSchema(many=True)
```

api/test/testScenarioService.py

```
import sys
import unittest
from flask import current_app
import logging
import app
from model.ctrl.suricataAlert import SuricataAlert
from model.ctrl.suricataRule import SuricataRule
from model.ctrl.team import Team
from model.ctrl.user import User
from model.ctrl.wazuhAlert import WazuhAlert
from model.ctrl.wazuhRule import WazuhRule
from model.script.ctrlTriggerEvent import CtrlTriggerEvent
from scriptTriggerConditionOperator import ScriptTriggerConditionOperator
from scriptTriggerConditionPropertyId import ScriptTriggerConditionPropertyId
from scriptTriggerEvent import ScriptTriggerEvent
from scriptTriggerEventStatus import ScriptTriggerEventStatus
import scenarioService
from appDatabase import app_db_session
from database import db_session


class TestScenarioService(unittest.TestCase):

    logging.basicConfig(stream=sys.stderr, level=logging.DEBUG)
```

```
    def setUp(self):
        self.app_context = app.create_app().app_context()
        self.app_context.push()
        logging.debug("\n\nRunning " + self._testMethodName)

    def tearDown(self):
        self.app_context.pop()

    def testGetTeamsByScenarioId_whenNoTeams_thenReturnEmptyList(self):
        result = scenarioService.getTeamsByScenarioId(1000)
        self.assertEqual([], result)

    def testGetTeamsByScenarioId_whenExist_thenReturnTeams(self):
        result = scenarioService.getTeamsByScenarioId(1)
        logging.debug("result " + str(result))
        self.assertEqual(3, len(result))
        self.assertIsInstance(result[0], Team)
        self.assertEqual(1, result[0].id)
        self.assertEqual('tid_1', result[0].team_id)
        self.assertEqual('The Avengers', result[0].name)

    def
testGetUsersByScenarioIdAndTeamId_whenNoUsers_thenReturnEmptyList(self):
        result = scenarioService.getUsersByScenarioIdAndTeamId(1000, 1)
        self.assertEqual([], result)

    def testGetUsersByScenarioIdAndTeamId_whenExist_thenReturnUsers(self):
        result = scenarioService.getUsersByScenarioIdAndTeamId(1, 1)
        self.assertEqual(4, len(result))
        self.assertIsInstance(result[0], User)
        self.assertEqual(1, result[0].id)
        self.assertEqual('uid_1', result[0].user_id)
        self.assertEqual('georgia', result[0].username)
        # following does not work, because defered loading doesn't load
        # fields initially, but loads them if they are referenced later
        # so printing or reading the value will make it load
        # self.assertEqual(None, result[0].email)

    def testGetEventCountPerHour_whenScenarioIdAndTeamId(self):
        result = scenarioService.getEventCountPerHour(1, 1, None)
        self.assertEqual(3, len(result))
        self.assertEqual('2022-05-06T00', result[0]['time'])
        self.assertEqual(1, result[0]['eventsNo'])
        self.assertEqual('2022-05-06T22', result[1]['time'])
        self.assertEqual(2, result[1]['eventsNo'])
        self.assertEqual('2022-05-06T23', result[2]['time'])
        self.assertEqual(17, result[2]['eventsNo'])
```

```
    def testGetEventCountPerHour_whenScenarioIdAndTeamIdAndUserId(self):
        result = scenarioService.getEventCountPerHour(1, 1, 1)
        self.assertEqual(2, len(result))
        self.assertEqual('2022-05-06T00', result[0]['time'])
        self.assertEqual(1, result[0]['eventsNo'])
        self.assertEqual('2022-05-06T23', result[1]['time'])
        self.assertEqual(7, result[1]['eventsNo'])
        logging.debug("result : " + str(result))

    def suite():
        suite = unittest.TestSuite()
        suite.addTest(TestScenarioService(
            'testGetTeamsByScenarioId'))
        suite.addTest(TestScenarioService(
            'testGetUsersByScenarioIdAndTeamId'))
        return suite

    if __name__ == '__main__':
        runner = unittest.TextTestRunner()
        runner.run(suite())
    else:
        logging.info("__name__ : " + str(__name__))
```

api/test/testScriptService.py

```
import re
import sys
import unittest
from flask import current_app
import logging
import app
from model.ctrl.suricataAlert import SuricataAlert
from model.ctrl.suricataRule import SuricataRule
from model.ctrl.wazuhAlert import WazuhAlert
from model.ctrl.wazuhRule import WazuhRule
from model.script.ctrlTriggerEvent import CtrlTriggerEvent
from scriptTriggerConditionOperator import ScriptTriggerConditionOperator
from scriptTriggerConditionPropertyId import ScriptTriggerConditionPropertyId
from scriptTriggerEvent import ScriptTriggerEvent
from scriptTriggerEventStatus import ScriptTriggerEventStatus
import scriptService
from appDatabase import app_db_session
from werkzeug import exceptions


class TestScriptService(unittest.TestCase):
```

```
    logging.basicConfig(stream=sys.stderr, level=logging.DEBUG)

    def setUp(self):
        self.app_context = app.create_app().app_context()
        self.app_context.push()
        logging.debug("\n\nRunning " + self._testMethodName)

    def tearDown(self):
        self.app_context.pop()

    def testRunScript_whenScriptWithVariables(self):
        data = {
            "scriptId" : 2,
            "variableValues": {
                "ip": "192.168.2.2"
            }
        }
        result = scriptService.runScript(data)

    def testRunScript_whenScriptVariableInvalid(self):
        data = {
            "scriptId" : 2,
            "variableValues": {
                "ip": "192.168.2.2="
            }
        }
        with self.assertRaises(exceptions.HTTPException) as http_error:
            scriptService.runScript(data)
            self.assertEqual(http_error.exception.code, 400)

    def suite():
        suite = unittest.TestSuite()
        suite.addTest(TestScriptService(
            'testRunScript_whenScriptWithVariables'))
        suite.addTest(TestScriptService(
            'testRunScript_whenScriptVariableInvalid'))
        return suite

    if __name__ == '__main__':
        runner = unittest.TextTestRunner()
        runner.run(suite())
    else:
        logging.info("__name__ : " + str(__name__))
```

api/test/testSCriptTriggerService.py

```
import sys
```

```
import unittest
from flask import current_app
import logging
import app
from model.ctrl.suricataAlert import SuricataAlert
from model.ctrl.suricataRule import SuricataRule
from model.ctrl.wazuhAlert import WazuhAlert
from model.ctrl.wazuhRule import WazuhRule
from model.script.ctrlTriggerEvent import CtrlTriggerEvent
from scriptTriggerConditionOperator import ScriptTriggerConditionOperator
from scriptTriggerConditionPropertyId import ScriptTriggerConditionPropertyId
from scriptTriggerEvent import ScriptTriggerEvent
from scriptTriggerEventStatus import ScriptTriggerEventStatus
import scriptTriggerService
from appDatabase import app_db_session


class TestScriptTriggerService(unittest.TestCase):

    logging.basicConfig(stream=sys.stderr, level=logging.DEBUG)

    def setUp(self):
        self.app_context = app.create_app().app_context()
        self.app_context.push()
        logging.debug("\n\nRunning " + self._testMethodName)

    def tearDown(self):
        self.app_context.pop()

    def testGetScriptTrigger_whenNotExist(self):
        result = scriptTriggerService.getScriptTrigger('0')
        self.assertEqual(None, result)

    def testGetScriptTrigger_whenTimeTriggered(self):
        result = scriptTriggerService.getScriptTrigger(1)
        logging.debug("result : " + str(result))
        self.assertEqual('1', result.id)
        self.assertEqual('1', result.scriptId)
        self.assertEqual([], result.conditions)
        self.assertEqual('1', result.script.id)
        self.assertEqual([], result.script.variables)
        self.assertEqual([], result.scriptVariables)

    def testGetScriptTrigger_whenEventTriggered(self):
        result = scriptTriggerService.getScriptTrigger(2)
        logging.debug("result : " + str(result))
        self.assertEqual('2', result.id)
        self.assertEqual('2', result.scriptId)
        self.assertEqual(3, len(result.conditions))
```

```
        result.conditions.sort(key=lambda x: x.id)

        i = 0
        for cond in result.conditions:
            i += 1
            self.assertEqual('2_' + str(i), cond.id)

        self.assertEqual('2', result.script.id)
        self.assertEqual(1, len(result.script.variables))
        self.assertEqual(len(result.script.variables),
                         len(result.scriptVariables))
        varVal = result.scriptVariables[0]
        self.assertEqual('2', varVal.triggerId)
        self.assertEqual('ip', varVal.variableId)
        self.assertEqual('DYNAMIC', varVal.valueType)
        self.assertEqual('WAZUH_RULE_AGENT_IP', varVal.value)

    def testActualConditionValueGetter_whenPropertySuricataRuleId(self):
        rule = SuricataRule(1234, 1, 'signature', 'severity')
        alert = SuricataAlert(12345, '127.0.0.1',
                              '127.0.0.2', 1, 'hostname', rule.id)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.SURICATA_RULE_ID.name, alert)
        self.assertEqual(rule.id, result)

    def testActualConditionValueGetter_whenPropertySuricataSourceIp(self):
        rule = SuricataRule(1234, 1, 'signature', 'severity')
        alert = SuricataAlert(12345, '127.0.0.1',
                              '127.0.0.2', 1, 'hostname', rule.id)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.SURICATA_SOURCE_IP.name, alert)
        self.assertEqual(alert.src_ip, result)

    def
testActualConditionValueGetter_whenPropertySuricataDestinationIp(self):
        rule = SuricataRule(1234, 1, 'signature', 'severity')
        alert = SuricataAlert(12345, '127.0.0.1',
                              '127.0.0.2', 1, 'hostname', rule.id)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.SURICATA_DESTINATION_IP.name,
alert)
        self.assertEqual(alert.dest_ip, result)

    def
testActualConditionValueGetter_whenPropertySuricataDestinationPort(self):
```

```
        rule = SuricataRule(1234, 1, 'signature', 'severity')
        alert = SuricataAlert(12345, '127.0.0.1',
                              '127.0.0.2', 1, 'hostname', rule.id)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.SURICATA_DESTINATION_PORT.name,
alert)
        self.assertEqual(alert.dest_port, result)

    def testActualConditionValueGetter_whenPropertySuricataHostName(self):
        rule = SuricataRule(1234, 1, 'signature', 'severity')
        alert = SuricataAlert(12345, '127.0.0.1',
                              '127.0.0.2', 1, 'hostname', rule.id)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.SURICATA_HOST_NAME.name, alert)
        self.assertEqual(alert.hostname, result)

    def testActualConditionValueGetter_whenPropertyWazuhRuleId(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.WAZUH_RULE_ID.name, alert)
        self.assertEqual(rule.id, result)

    def testActualConditionValueGetter_whenPropertyWazuhRuleLevel(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.WAZUH_RULE_LEVEL.name, alert)
        self.assertEqual(rule.level, result)

    def testActualConditionValueGetter_whenPropertyWazuhRuleAgentId(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.WAZUH_RULE_AGENT_ID.name, alert)
        self.assertEqual(alert.agent_id, result)

    def testActualConditionValueGetter_whenPropertyWazuhAgentHostName(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
```

```
                            'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.WAZUH_RULE_AGENT_HOST_NAME.name,
alert)
        self.assertEqual(alert.agent_hostname, result)

    def testActualConditionValueGetter_whenPropertyWazuhAgentIp(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                            'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.actualConditionValueGetter(
            ScriptTriggerConditionPropertyId.WAZUH_RULE_AGENT_IP.name, alert)
        self.assertEqual(alert.agent_ip, result)

    def
testBaseConditionOk_whenOperatorEqualAndValueStringAndOk_thenReturnTrue(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                            'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.EQUAL.name, '127.0.0.1',
alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorEqualAndValueIntegerAndOk_thenReturnTrue(self)
:
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                            'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.EQUAL.name, 1234, rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorEqualAndValueStringAndNotOk_thenReturnFalse(se
lf):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                            'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.EQUAL.name, '127.0.0.2',
alert.agent_ip)
```

```
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorEqualAndValueIntegerAndNotOk_thenReturnFalse(s
elf):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.EQUAL.name, 12345, rule.id)
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorNotEqualAndValueStringAndOk_thenReturnTrue(sel
f):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.NOT_EQUAL.name, '127.0.0.3',
alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorNotEqualAndValueIntegerAndOk_thenReturnTrue(se
lf):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.NOT_EQUAL.name, 12345, rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorNotEqualAndValueStringAndNotOk_thenReturnFalse
(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.NOT_EQUAL.name, '127.0.0.1',
alert.agent_ip)
        self.assertEqual(False, result)
```

```
    def
testBaseConditionOk_whenOperatorNotEqualAndValueIntegerAndNotOk_thenReturnFals
e(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.NOT_EQUAL.name, 1234, rule.id)
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorGreaterThanAndValueStringAndOk_thenReturnTrue(
self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, '127.0.0.0',
alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorGreaterThanAndValueIntegerAndOk_thenReturnTrue
(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, 1233, rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorGreaterThanAndValueStringAndNotOk_thenReturnFa
lse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, '127.0.0.2',
alert.agent_ip)
        self.assertEqual(False, result)
```

```
    def
testBaseConditionOk_whenOperatorGreaterThanAndValueIntegerAndNotOk_thenReturnF
alse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, 1234, rule.id)
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorGreaterThanAndValueStringAndOk_thenReturnTrue(
self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, '127.0.0.0',
alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorGreaterThanAndValueIntegerAndOk_thenReturnTrue
(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, 1233, rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorEqualAndValueStringAndNotOk_thenReturnFalse(se
lf):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, '127.0.0.2',
alert.agent_ip)
        self.assertEqual(False, result)
```

```
    def
testBaseConditionOk_whenOperatorGreaterThanAndValueIntegerAndNotOk_thenReturnF
alse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN.name, 1234, rule.id)
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorGreaterThanOrEqualAndValueStringAndOk_thenRetu
rnTrue(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN_OR_EQUAL.name,
'127.0.0.0', alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorGreaterThanOrEqualAndValueIntegerAndOk_thenRet
urnTrue(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN_OR_EQUAL.name, 1234,
rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorGreaterThanOrEqualAndValueStringAndNotOk_thenR
eturnFalse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN_OR_EQUAL.name,
'127.0.0.2', alert.agent_ip)
        self.assertEqual(False, result)
```

```
    def
testBaseConditionOk_whenOperatorGreaterThanOrEqualAndValueIntegerAndNotOk_then
ReturnFalse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.GREATER_THAN_OR_EQUAL.name, 1235,
rule.id)
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorLessThanAndValueStringAndOk_thenReturnTrue(sel
f):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN.name, '127.0.0.2',
alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorLessThanAndValueIntegerAndOk_thenReturnTrue(se
lf):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN.name, 1235, rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorLessThanAndValueStringAndNotOk_thenReturnFalse
(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN.name, '127.0.0.1',
alert.agent_ip)
        self.assertEqual(False, result)
```

```
    def
testBaseConditionOk_whenOperatorLessThanAndValueIntegerAndNotOk_thenReturnFals
e(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN.name, 1234, rule.id)
        self.assertEqual(False, result)

    def
testBaseConditionOk_whenOperatorLessThanOrEqualAndValueStringAndOk_thenReturnT
rue(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN_OR_EQUAL.name,
'127.0.0.2', alert.agent_ip)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorLessThanOrEqualAndValueIntegerAndOk_thenReturn
True(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN_OR_EQUAL.name, 1234,
rule.id)
        self.assertEqual(True, result)

    def
testBaseConditionOk_whenOperatorLessThanOrEqualAndValueStringAndNotOk_thenRetu
rnFalse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN_OR_EQUAL.name,
'127.0.0.0', alert.agent_ip)
        self.assertEqual(False, result)
```

```
    def
testBaseConditionOk_whenOperatorLessThanOrEqualThanAndValueIntegerAndNotOk_the
nReturnFalse(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        result = scriptTriggerService.baseConditionOk(
            ScriptTriggerConditionOperator.LESS_THAN_OR_EQUAL.name, 1233,
rule.id)
        self.assertEqual(False, result)

    def testCreateEvent(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        event = CtrlTriggerEvent(ScriptTriggerEvent.WAZUH_ALERT.name, 123456,
                         ScriptTriggerEventStatus.PENDING.name, event=alert)

        result = scriptTriggerService.createEvent(event)
        self.assertEqual(event, result)

        storedEvent = CtrlTriggerEvent.query.get(result.id)
        self.assertEqual(result.id, storedEvent.id)
        self.assertEqual(result.eventType, storedEvent.eventType)
        self.assertEqual(result.eventId, storedEvent.eventId)
        self.assertEqual(result.status, storedEvent.status)
        self.assertEqual(result.triggered, storedEvent.triggered)


    def testUpdateEvent_whenTriggeredHasEntries(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        event = CtrlTriggerEvent(ScriptTriggerEvent.WAZUH_ALERT.name, 123456,
                         ScriptTriggerEventStatus.PENDING.name, event=alert)

        initEvent = scriptTriggerService.createEvent(event)
        result = scriptTriggerService.updateEvent(
            initEvent, ScriptTriggerEventStatus.PROCESSED, [1, 5])

        app_db_session.flush()
        storedEvent = CtrlTriggerEvent.query.get(result.id)
        self.assertEqual(result.id, storedEvent.id)
        self.assertEqual(result.eventType, storedEvent.eventType)
        self.assertEqual(result.eventId, storedEvent.eventId)
        self.assertEqual(
```

```
            ScriptTriggerEventStatus.PROCESSED.name, storedEvent.status)
        self.assertEqual('1,5', storedEvent.triggered)
        self.assertEqual(result.event, storedEvent.event)

    def
testUpdateEvent_whenTriggeredEmpty_thenDoNotUpdateTriggeredField(self):
        rule = WazuhRule(1234, 5678, 5, 'description')
        alert = WazuhAlert(12345, 67890, 98765,
                           'agent hostname', '127.0.0.1', 8732647832)
        alert.rule = rule
        event = CtrlTriggerEvent(ScriptTriggerEvent.WAZUH_ALERT.name, 123456,
                         ScriptTriggerEventStatus.PENDING.name, event=alert)

        initEvent = scriptTriggerService.createEvent(event)
        result = scriptTriggerService.updateEvent(
            initEvent, ScriptTriggerEventStatus.PROCESSED, [])

        app_db_session.flush()
        storedEvent = CtrlTriggerEvent.query.get(result.id)
        self.assertEqual(result.id, storedEvent.id)
        self.assertEqual(result.eventType, storedEvent.eventType)
        self.assertEqual(result.eventId, storedEvent.eventId)
        self.assertEqual(
            ScriptTriggerEventStatus.PROCESSED.name, storedEvent.status)
        self.assertEqual(None, storedEvent.triggered)
        self.assertEqual(result.event, storedEvent.event)

    def testDeleteEvent(self):
        event = CtrlTriggerEvent(ScriptTriggerEvent.WAZUH_ALERT.name, 123456,
                         ScriptTriggerEventStatus.PENDING.name)
        event = scriptTriggerService.createEvent(event)
        scriptTriggerService.deleteEvent(event)
        storedEvent = CtrlTriggerEvent.query.get(event.id)
        self.assertEqual(None, storedEvent)

    def suite():
        suite = unittest.TestSuite()
        suite.addTest(TestScriptTriggerService(
            'testGetScriptTrigger_whenNotExist'))
        suite.addTest(TestScriptTriggerService(
            'testGetScriptTrigger_whenTimeTriggered'))
        suite.addTest(TestScriptTriggerService(
            'testGetScriptTrigger_whenEventTriggered'))
        return suite

    if __name__ == '__main__':
        runner = unittest.TextTestRunner()
        runner.run(suite())
```

```
    else:
        logging.info("__name__ : " + str(__name__))
```

A.3 Κώδικας Frontend

src/App.js

```
import React, { useState, useEffect } from 'react';
import logo from './logo.svg';
import { useStyles } from './DrawerStyles.js'
import CssBaseline from '@material-ui/core/CssBaseline';
import MiniDrawer from './Drawer';
import Calendar from './Calendar';
import Campaign from './Campaign';
import IndividualScoring from './individualScore/IndividualScoring';
import ScenarioNetworks from './ScenarioNetworks';
import Reports from './Reports';
import Scenarios from './Scenarios';
import Scoring from './score/Scoring';
import TeamScoring from './teamScore/TeamScoring';
import Timeline from './Timeline';
import ScriptManagement from './script/ScriptManagement'
import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-
dom';
import ScenarioTeams from './ScenarioTeams';

export default function App() {
  const classes = useStyles();

  const routes = [
    {
      path: "/scenarios",
      component: Scenarios
    },
    {
      path: "/scenarios/networks",
      component: ScenarioNetworks
    },
    {
      path: "/scenarios/teams",
      component: ScenarioTeams
    },
    {
      path: "/timeline",
      component: Timeline
```

```
  },
  {
    path: "/scoring",
    component: Scoring
  },
  {
    path: "/scoring/team",
    component: TeamScoring
  },
  {
    path: "/scoring/individual",
    component: IndividualScoring
  },
  {
    path: "/campaign",
    component: Campaign
  },
  {
    path: "/calendar",
    component: Calendar
  },
  {
    path: "/reports",
    component: Reports
  },
  {
    path: "/scripts",
    component: ScriptManagement
  },
];

return (
  <div className={classes.root}>
    <CssBaseline />
    <MiniDrawer />
    <main className={classes.content}>
      <div className={classes.toolbar} />
      <Switch>
        <Route exact path="/scenarios"><Scenarios /></Route>
        {routes.map((route, index) => (
          <Route
            key={index}
            path={route.path}
          >
            {route.component}
          </Route>
        ))}
      </Switch>
```

```
      </main>
    </div>
  );
}
```

src/AttaclsTable.js

```
import React from 'react';
import BaseTable from './BaseTable';

const columns = [
  { id: 'time', label: 'Time', minWidth: 170 },
  { id: 'status', label: 'Attack status', minWidth: 170 },
  { id: 'type', label: 'Attack type', minWidth: 170 }
];

export default function AttacksTable(props) {
  return (
    <BaseTable columns={columns} rows={props.data} />
  );
}
```

src/AttacksTableContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import AttacksTable from './AttacksTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function AttacksTableContainer(props) {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        {props.title}
      </Typography>
```

```
      <AttacksTable data={props.eventsData}/>
    </div>
  );
}
```

src/BaseTable.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Paper from '@material-ui/core/Paper';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';

const useStyles = makeStyles({
  root: {
    width: '100%',
  },
  container: {
    maxHeight: '300px',
  }
});

export default function BaseTable(props) {
  const classes = useStyles();

  return (
    <Paper className={classes.root}>
      <TableContainer className={classes.container}>
        <Table stickyHeader aria-label="sticky table" size="small">
          <TableHead key="header">
            <TableRow className={classes.header}>
              {props.columns.map((column) => (
                <TableCell
                  key={column.id}
                  align={column.align}
                  style={{ minWidth: column.minWidth, backgroundColor: 'gray'
}}
                >
                  {column.label}
                </TableCell>
              ))}
            </TableRow>
          </TableHead>
```

```
          <TableBody>
            {props.rows.map((row) => {
              return (
                <TableRow hover role="checkbox" tabIndex={-1} key={row.code}>
                  {props.columns.map((column) => {
                    const value = row[column.id];
                    return (
                      <TableCell key={column.id} align={column.align}>
                        {column.format && typeof value === 'number' ?
column.format(value) : value}
                      </TableCell>
                    );
                  })}
                </TableRow>
              );
            })}
          </TableBody>
        </Table>
      </TableContainer>
    </Paper>
  );
}
```

src/Calendar.js

```
import React, { useState, useEffect } from 'react';
import FullCalendar, { formatDate } from '@fullcalendar/react';
import dayGridPlugin from '@fullcalendar/daygrid';
import timeGridPlugin from '@fullcalendar/timegrid'
import interactionPlugin from '@fullcalendar/interaction' // needed for
dayClick
import listPlugin from '@fullcalendar/list';

function createEvent(titleValue, startValue, endValue, allDayValue) {
  return {
    title: titleValue, start: startValue, end: endValue, allDay: allDayValue
  };
}

function parseEventsResponse(content) {
  console.log("response : " + JSON.stringify(content));
  let eventList = [];
  content.forEach(r => {
    eventList.push(createEvent(r.name, r.isoDateFrom, r.isoDateTo, r.allDay));
  });
  console.log("eventList : " + JSON.stringify(eventList));
  return eventList;
```

```
}

function transformEvent(eventData) {
  console.log("eventData : " + JSON.stringify(eventData));
  return createEvent(eventData.name, eventData.isoDateFrom,
eventData.isoDateTo, eventData.allDay);
}

export default function Calendar() {
  const calEventSources = [
    {
      url: '/api/calendar',
      method: 'GET',
      failure: function () {
        alert('There was an error while fetching events!');
      },
      color: '#3f51b5',   // a non-ajax option
      textColor: '#fff' // a non-ajax option
    }
  ];

  return (
    <div>
      <FullCalendar
        defaultView="dayGridMonth"
        headerToolbar={{
          left: 'prev,next today',
          center: 'title',
          right: 'dayGridMonth,timeGridWeek,timeGridDay,listWeek'
        }}
        plugins={[dayGridPlugin, timeGridPlugin, interactionPlugin,
listPlugin]}
        eventSources={calEventSources}
        //eventSourceSuccess={ parseEventsResponse }
        eventDataTransform={transformEvent}
      //eventClick = { }
      />
    </div>
  );
}
```

src/Campaign.js

```
import React, { useState, useEffect } from 'react';
import PageTitle from './PageTitle';
import CtrlDropDown from './CtrlDropDown';
import Grid from '@material-ui/core/Grid';
```

```
import { getScenarios, getScenarioTeams } from './ServerCaller';
import SecurityEventsContainer from './SecurityEventsContainer';
import AttacksTableContainer from './AttacksTableContainer';
import ComponentStatusContainer from './ComponentStatusContainer';
import NetworksTableContainer from './NetworksTableContainer';
import { Typography } from '@mui/material';
import RankContainer from './RankContainer';

function scenariosDropDownTextDisplayer(scenario) {
  // return scenario.scenario_id + " - " + scenario.title;
  return scenario.title;
}

function teamsDropDownTextDisplayer(team) {
  // return team.team_id + " - " + team.name;
  return team.name;
}

function createEventData(eventName, startTime, source, destination, riskLevel,
code) {
  return { eventName, startTime, source, destination, riskLevel, code };
}

function createEventRows(data) {
  let rowList = [];
  data.forEach(r => {
    rowList.push(createEventData(r.name, r.startTime, r.source, r.destination,
r.riskLevel, r.code));
  });
  console.log('eventData rowList : ' + JSON.stringify(rowList));
  return rowList;
}

function createClientEventRows() {
  let rowList = [];
  for (let i = 0; i < 10; i++) {
    let eventNo = Math.floor(Math.random() * (10000 - 1 + 1)) + 1;
    rowList.push(createEventData('Event ' + eventNo, new
Date().toISOString().substring(0, 19), 'Source ' + eventNo, 'Destination ' +
eventNo, 'Risk Level ' + eventNo, eventNo));
  }
  console.log('eventClientData rowList : ' + JSON.stringify(rowList));
  return rowList;
}

function createAttackData(time, status, type, code) {
  return { time, status, type, code };
}
```

```
function createClientAttackRows() {
  let rowList = [];
  for (let i = 0; i < 10; i++) {
    let attackNo = Math.floor(Math.random() * (10000 - 1 + 1)) + 1;
    rowList.push(createAttackData(new Date().toISOString().substring(0, 19),
'Attack status ' + attackNo, 'Attack type ' + attackNo, attackNo));
  }
  console.log('attackClientData rowList : ' + JSON.stringify(rowList));
  return rowList;
}

export default function Campaign(props) {
  const [selectedScenario, setSelectedScenario] = useState();
  const [selectedTeam, setSelectedTeam] = useState();
  const [scenarios, setScenarios] = useState([]);
  const [teams, setTeams] = useState([]);
  const [intervalId, setIntervald] = useState(0);
  const [eventsData, setEventsData] = useState([]);
  const [attacksData, setAttacksData] = useState([]);
  const [rank, setRank] = useState(0);

  const title = "Campaign";
  const allOptionValue = "--all--";

  function stopPolling() {
    console.log("stop polling for intervalId " + intervalId);
    clearInterval(intervalId);
  }

  useEffect(() => {
    function updateScenarios(scenarios) {
      setScenarios(scenarios);
      if (scenarios && scenarios.length)
        setSelectedScenario(scenarios[0].id);
    }

    getScenarios(updateScenarios, null);
    return () => stopPolling();
  }, []);

  useEffect(() => {
    console.log("selected scenario changed. will fetch teams for scenarioId :
" + selectedScenario);
    stopPolling();
    setTeams([])
    setSelectedTeam(null);
    setRank(0);
```

```
    function updateTeams(teams) {
      setTeams(teams);
      if (teams && teams.length)
        setSelectedTeam(allOptionValue);
    }

    if (selectedScenario) {
      getScenarioTeams(selectedScenario, updateTeams, null);
    }
    return () => stopPolling();
  }, [selectedScenario]);

  useEffect(() => {
    console.log("selected team changed. will fetch users for scenarioId,
teamId : " + selectedScenario + ", " + selectedTeam);
    stopPolling();

    function startPolling() {
      console.log("starting polling for scenario " + selectedScenario + " and
team " + selectedTeam);
      var prevInterval = intervalId;
      console.log("prevInterval : " + prevInterval);
      var newInterval = setInterval(
        () => {
          console.log("sending for interval : " + prevInterval);
          setEventsData(createClientEventRows());
          setAttacksData(createClientAttackRows());
          setRank(Math.floor(Math.random() * (10 - 1 + 1)) + 1);
        }, 5000
      );
      setIntervald(newInterval);
      console.log("new intervalId : " + intervalId);
    }

    if (selectedScenario && selectedTeam) {
      startPolling();
    }
    return () => stopPolling();
  }, [selectedScenario, selectedTeam]);

  const scenarioChangeHandler = (event) => {
    setSelectedScenario(event.target.value);
  };

  const teamChangeHandler = (event) => {
    setSelectedTeam(event.target.value);
  };
```

```
  return (
    <div>
      <PageTitle titleText={title} />
      <Grid
        container
        direction="row"
        justifyContent="center"
        alignItems="center"
        spacing={3}
        style={{ marginBottom: '2vh' }}
      >
        <Grid item xs>
          <CtrlDropDown title="Scenario" options={scenarios}
textDisplayer={scenariosDropDownTextDisplayer}
changeHandler={scenarioChangeHandler} />
        </Grid>
        <Grid item xs>
          <CtrlDropDown title="Team" options={teams}
textDisplayer={teamsDropDownTextDisplayer} changeHandler={teamChangeHandler}
/>
        </Grid>
      </Grid>

      <RankContainer style={{ marginBottom: '2vh' }} title="Live ranking"
rank={rank} />

      <Grid
        container
        direction="row"
        justifyContent='center'
        alignItems='stretch'
        spacing={2}
      >
        <Grid item xs={12} md={8} lg={10}>
          <SecurityEventsContainer title="Live Security Events"
eventsData={eventsData} />
        </Grid>
        <Grid item xs={12} md={4} lg={2}>
          <NetworksTableContainer />
        </Grid>
        <Grid item xs={12} md={8} lg={10}>
          <AttacksTableContainer title="Live Attacks Log"
eventsData={attacksData} />
        </Grid>
        <Grid item xs={12} md={4} lg={2}>
          <ComponentStatusContainer />
        </Grid>
      </Grid>
```

```
    </div>
  );
}
```

src/ComponentStatusContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import ComponentStatusTable from './ComponentStatusTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function ComponentStatusContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Component status
      </Typography>

      <ComponentStatusTable />
    </div>
  );
}
```

src/ComponentStatusTable.js

```
import React from 'react';
import BaseTable from './BaseTable';

const columns = [
  { id: 'component', label: 'Compopnent', minWidth: 170 },
  { id: 'status', label: 'Status', minWidth: 100 },
];

function createData(component, status, code) {
  return { component, status, code };
}
```

```
const rows = [
  createData('RT1', 'down', 1),
  createData('RT2', 'up', 2),
  createData('RT1', 'down', 3),
  createData('RT2', 'up', 4),
  createData('RT1', 'down', 5),
  createData('RT2', 'up', 6),
  createData('RT1', 'down', 7),
  createData('RT2', 'up', 8),
  createData('RT1', 'down', 9),
  createData('RT2', 'up', 10),
  createData('RT1', 'down', 11),
  createData('RT2', 'up', 12),
  createData('RT1', 'down', 13),
  createData('RT2', 'up', 14),
];


export default function ComponentStatusTable() {
  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/CtrlDropDown.js

```
import React, { useEffect, useState } from 'react';
import InputLabel from '@material-ui/core/InputLabel';
import Select from '@material-ui/core/Select';
import { makeStyles } from '@material-ui/core/styles';

const useStyles = makeStyles((theme) => ({
  fullWidth: {
    width: '100%'
  }
}));

function createOptions(options, textDisplayer) {
  if (!textDisplayer) {
    textDisplayer = (r) => r.id;
  }
  return (
    options.map((r) => (<option key={r.id}
value={r.id}>{textDisplayer(r)}</option>))
  );
};
```

```
export default function CtrlDropDown(props) {
  const classes = useStyles();
  const [selected, setSelected] = useState();

  const handleChange = (event) => {
    setSelected(event.target.value);
    if (props.changeHandler)
      props.changeHandler(event);
  };

  return (
    <div>
      <InputLabel>{props.title}</InputLabel>
      <Select
        native
        value={selected}
        onChange={handleChange}
        label={props.title}
        inputProps={{
          name: props.title,
          id: props.title,
        }}
        className={classes.fullWidth}
      >
        {props.allOptionValue && props.options && props.options.length &&
<option key={props.allOptionValue} value={props.allOptionValue}>All</option>}
        {createOptions(props.options, props.textDisplayer)}
      </Select>
    </div>
  );
}
```

src/CtrlSnackBar.js

```
import { Snackbar } from '@material-ui/core';
import React, { useState, useEffect } from 'react';

export default function CtrlSnackBar(props) {
  const [snackPack, setSnackPack] = React.useState([]);
  const [open, setOpen] = React.useState(false);
  const [messageInfo, setMessageInfo] = React.useState(undefined);
  const [vertical, setVertical] = React.useState('top');
  const [horizontal, setHorizontal] = React.useState('right');

  React.useEffect(() => {
    if (snackPack.length && !messageInfo) {
```

```
      // Set a new snack when we don't have an active one
      setMessageInfo({ ...snackPack[0] });
      setSnackPack((prev) => prev.slice(1));
      setOpen(true);
    } else if (snackPack.length && messageInfo && open) {
      // Close an active snack when a new one is added
      setOpen(false);
    }
  }, [snackPack, messageInfo, open]);

  const handleClick = (message) => () => {
    setSnackPack((prev) => [...prev, { message, key: new Date().getTime() }]);
  };

  const handleClose = (event, reason) => {
    if (reason === 'clickaway') {
      return;
    }
    setOpen(false);
  };

  const handleExited = () => {
    setMessageInfo(undefined);
  };

  return (
    <Snackbar style={{
      borderColor: 'red',
    }}
      anchorOrigin={{ vertical, horizontal }}
      autoHideDuration={6000}
      open={open}
      onClose={handleClose}
      onExited={handleExited}
      message={messageInfo ? messageInfo.message : undefined}
      key={messageInfo ? messageInfo.key : undefined}
    />
  );
}
```

src/Drawer.js

```
import React from 'react';
import clsx from 'clsx';
import { makeStyles, useTheme } from '@material-ui/core/styles';
import Drawer from '@material-ui/core/Drawer';
import AppBar from '@material-ui/core/AppBar';
```

```
import Toolbar from '@material-ui/core/Toolbar';
import List from '@material-ui/core/List';
import CssBaseline from '@material-ui/core/CssBaseline';
import Typography from '@material-ui/core/Typography';
import Divider from '@material-ui/core/Divider';
import IconButton from '@material-ui/core/IconButton';
import MenuIcon from '@mui/icons-material/Menu';
import ChevronLeftIcon from '@mui/icons-material/ChevronLeft';
import ChevronRightIcon from '@mui/icons-material/ChevronRight';
import ListItem from '@material-ui/core/ListItem';
import ListItemIcon from '@material-ui/core/ListItemIcon';
import ListItemText from '@material-ui/core/ListItemText';
import DashboardIcon from '@mui/icons-material/Dashboard';
import ScoreIcon from '@mui/icons-material/Score';
import TimelineIcon from '@mui/icons-material/Timeline';
import FlagIcon from '@mui/icons-material/Flag';
import CalendarTodayIcon from '@mui/icons-material/CalendarToday';
import AssessmentIcon from '@mui/icons-material/Assessment';
import GroupIcon from '@mui/icons-material/Group';
import PersonIcon from '@mui/icons-material/Person';
import RouterIcon from '@mui/icons-material/Router';
import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-
dom';
import Calendar from './Calendar';
import Campaign from './Campaign';
import IndividualScoring from './IndividualScoring';
import ScenarioNetworks from './ScenarioNetworks';
import Reports from './Reports';
import ScenariosTable from './ScenariosTable';
import Scoring from './Scoring';
import TeamScoring from './TeamScoring';
import Timeline from './Timeline';
import { withRouter } from "react-router-dom";
import { useStyles } from './DrawerStyles.js'
import { createBrowserHistory } from "history";
import ScriptManagement from './script/ScriptManagement';
import { Code } from '@mui/icons-material';
import ScenarioTeams from './ScenarioTeams';

function MiniDrawer(props) {
  const history = createBrowserHistory();
  const classes = useStyles();
  const theme = useTheme();
  const [open, setOpen] = React.useState(false);
  const [selectedIndex, setSelectedIndex] = React.useState(0);

  const handleDrawerOpen = () => {
    setOpen(true);
```

```
};

const handleDrawerClose = () => {
  setOpen(false);
};

const handleClick = (index, path) => {
  console.log('clicked : ' + index + '\t' + path);
  setSelectedIndex(index);
  history.push(path);
};

const itemsList = [
  {
    text: "Scenarios",
    icon: <DashboardIcon />,
    path: "/scenarios",
    component: ScenariosTable,
    uiClass: ''
  },
  {
    text: "Networks",
    icon: <RouterIcon />,
    path: "/scenarios/networks",
    component: ScenarioNetworks,
    uiClass: classes.nested
  },
  {
    text: "Teams",
    icon: <GroupIcon />,
    path: "/scenarios/teams",
    component: ScenarioTeams,
    uiClass: classes.nested
  },
  {
    text: "Timeline",
    icon: <TimelineIcon />,
    path: "/timeline",
    component: Timeline,
    uiClass: ''
  },
  {
    text: "Scoring",
    icon: <ScoreIcon />,
    path: "/scoring",
    component: Scoring,
    uiClass: ''
  },
```

```
  {
    text: "Team-based",
    icon: <GroupIcon />,
    path: "/scoring/team",
    component: TeamScoring,
    uiClass: classes.nested
  },
  {
    text: "Individual",
    icon: <PersonIcon />,
    path: "/scoring/individual",
    component: IndividualScoring,
    uiClass: classes.nested
  },
  {
    text: "Scenario Campaign",
    icon: <FlagIcon />,
    path: "/campaign",
    component: Campaign,
    uiClass: ''
  },
  {
    text: "Calendar",
    icon: <CalendarTodayIcon />,
    path: "/calendar",
    component: Calendar,
    uiClass: ''
  },
  {
    text: "Visualisation Reports",
    icon: <AssessmentIcon />,
    path: "/reports",
    component: Reports,
    uiClass: ''
  },
  {
    text: "Script Management",
    icon: <Code />,
    path: "/scripts",
    component: ScriptManagement,
    uiClass: ''
  }
];

return (
  <div>
    <AppBar
      position="fixed"
```

```
        className={clsx(classes.appBar, {
          [classes.appBarShift]: open,
        })}
      >
        <Toolbar>
          <IconButton
            color="inherit"
            aria-label="open drawer"
            onClick={handleDrawerOpen}
            edge="start"
            className={clsx(classes.menuButton, {
              [classes.hide]: open,
            })}
          >
            <MenuIcon />
          </IconButton>
          <Typography variant="h6" noWrap>
            CTRL Admin Console
          </Typography>
        </Toolbar>
      </AppBar>
      <Drawer
        variant="permanent"
        className={clsx(classes.drawer, {
          [classes.drawerOpen]: open,
          [classes.drawerClose]: !open,
        })}
        classes={{
          paper: clsx({
            [classes.drawerOpen]: open,
            [classes.drawerClose]: !open,
          }),
        }}
      >
        <div className={classes.toolbar}>
          <IconButton onClick={handleDrawerClose}>
            {theme.direction === 'rtl' ? <ChevronRightIcon /> :
<ChevronLeftIcon />}
          </IconButton>
        </div>
        <Divider />
        <List>
          {itemsList.map((item, index) => {
            const { text, icon, path, uiClass } = item;
            return (
              <ListItem button key={text} onClick={() =>
this.handleClick(index, path)} className={uiClass} selected={selectedIndex ===
index}>
```

```
              {icon && <ListItemIcon>{icon}</ListItemIcon>}
              <ListItemText primary={text} />
            </ListItem>
          );
        })}
      </List>
    </Drawer>
  </div>
 );
}

export default withRouter(MiniDrawer);
```

src/DrawerStyles.js

```
import { makeStyles } from '@material-ui/core/styles';

const drawerWidth = 240;

const useStyles = makeStyles((theme) => ({
  root: {
    display: 'flex',
  },
  appBar: {
    zIndex: theme.zIndex.drawer + 1,
    transition: theme.transitions.create(['width', 'margin'], {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
  },
  appBarShift: {
    marginLeft: drawerWidth,
    width: `calc(100% - ${drawerWidth}px)`,
    transition: theme.transitions.create(['width', 'margin'], {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.enteringScreen,
    }),
  },
  menuButton: {
    marginRight: 36,
  },
  hide: {
    display: 'none',
  },
  drawer: {
    width: drawerWidth,
    flexShrink: 0,
```

```
    whiteSpace: 'nowrap',
  },
  drawerOpen: {
    width: drawerWidth,
    transition: theme.transitions.create('width', {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.enteringScreen,
    }),
  },
  drawerClose: {
    transition: theme.transitions.create('width', {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
    overflowX: 'hidden',
    width: theme.spacing(7) + 1,
    [theme.breakpoints.up('sm')]: {
      width: theme.spacing(9) + 1,
    },
  },
  toolbar: {
    display: 'flex',
    alignItems: 'center',
    justifyContent: 'flex-end',
    padding: theme.spacing(0, 1),
    // necessary for content to be below app bar
    ...theme.mixins.toolbar,
  },
  nested: {
    paddingLeft: theme.spacing(4),
  },
  content: {
    flexGrow: 1,
    padding: theme.spacing(3),
  },
}));

export { useStyles };
```

src/EventsBasedOnNumberChart.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import {
  ResponsiveContainer, LineChart, Line, XAxis, Tooltip
} from 'recharts';
import Typography from '@material-ui/core/Typography';
```

```
const data = [
  {
    time: 0, eventsNo: 10,
  },
  {
    time: 1, eventsNo: 300,
  },
  {
    time: 2, eventsNo: 200,
  },
  {
    time: 3, eventsNo: 278,
  },
  {
    time: 4, eventsNo: 189,
  },
  {
    time: 5, eventsNo: 239,
  },
  {
    time: 6, eventsNo: 349,
  },
  {
    time: 7, eventsNo: 45,
  },
];

const useStyles = makeStyles((theme) => ({
  outer: {
    marginTop: '15vh'
  }
}));

export default function EventsBasedOnNumberChart(props) {
  //const [data, setData] = useState([]);
  const classes = useStyles();

  return (
    <div className={classes.outer}>
      <Typography variant="overline" component="div" align="center">
      Events t --&gt; e (based on number of events)
      </Typography>
      <ResponsiveContainer width={'100%'} height={300}>
        <LineChart
          data={props.data}
        //margin={{
        //  top: 30, right: 5, left: 5, bottom: 5,
```

```
        //}}
        >
          <XAxis dataKey="time" />
          <Tooltip />
          <Line type="monotone" dataKey="eventsNo" name="Events No"
stroke="#8884d8" activeDot={{ r: 8 }} />
        </LineChart>
      </ResponsiveContainer>
    </div>
  );
}
```

src/NetworkGraph.js

```
import React, { useState, useEffect } from 'react';
import Graph from "graphology";
import { SigmaContainer, useLoadGraph } from "@react-sigma/core";
import "@react-sigma/core/lib/react-sigma.min.css";
import { useLayoutCircular } from "@react-sigma/layout-circular";
import RouterIcon from '@mui/icons-material/Router';
import CloudIcon from '@mui/icons-material/Cloud';
import Cloud from '@mui/icons-material/Cloud';

const items = [
  {
    "itemId": "NetworkManager5",
    "name": "vmengine Network Manager",
    "kind": "NetworkManager",
    "model": "ManageIQ::Providers::Redhat::NetworkManager",
    "miq_id": 5,
    "key": "NetworkManager5",
    "status": "Valid",
    "display_kind": "Redhat"
  },
  {
    "itemId": "CloudTenant2",
    "name": "tenant",
    "kind": "CloudTenant",
    "model": "ManageIQ::Providers::Openstack::CloudManager::CloudTenant",
    "miq_id": 2,
    "key": "CloudTenant2",
    "status": "OK",
    "display_kind": "CloudTenant"
  },
  {
    "itemId": "NetworkManager3",
    "name": "threatrealm1.sec.ouc.ac.cy Network Manager",
```

```
    "kind": "NetworkManager",
    "model": "ManageIQ::Providers::Redhat::NetworkManager",
    "miq_id": 3,
    "key": "NetworkManager3",
    "status": "Valid",
    "display_kind": "Redhat"
  },
  {
    "itemId": "CloudTenant1",
    "name": "tenant",
    "kind": "CloudTenant",
    "model": "ManageIQ::Providers::Openstack::CloudManager::CloudTenant",
    "miq_id": 1,
    "key": "CloudTenant1",
    "status": "OK",
    "display_kind": "CloudTenant"
  }
];

const relations = [
  {
    "source": "NetworkManager5",
    "target": "CloudTenant2"
  },
  {
    "source": "NetworkManager5",
    "target": "CloudTenant1"
  },
  {
    "source": "NetworkManager3",
    "target": "CloudTenant1"
  },
  {
    "source": "NetworkManager3",
    "target": "NetworkManager5"
  }
];

const kinds = {
  "NetworkRouter": true,
  "CloudSubnet": true,
  "Vm": true,
  "NetworkManager": true,
  "FloatingIp": true,
  "CloudNetwork": true,
  "NetworkPort": true,
  "CloudTenant": Cloud,
  "SecurityGroup": true,
```

```
  "LoadBalancer": true,
  "Tag": true,
  "AvailabilityZone": true
};

const imagePerkind = {
  "NetworkRouter": "./images/router.png",
  "CloudSubnet": "./images/cloudSubnet.png",
  "Vm": "./images/vm.png",
  "NetworkManager": "./images/networkManager.png",
  "FloatingIp": "./images/location.png",
  "CloudNetwork": false,
  "NetworkPort": false,
  "CloudTenant": false,
  "SecurityGroup": "./images/secureCloud.png",
  "LoadBalancer": "./images/loadBalancer.png",
  "Tag": "./images/tag.png",
  "AvailabilityZone": false
};

export const LoadGraph = (props) => {
  const loadGraph = useLoadGraph();
  const { positions, assign } = useLayoutCircular();

  //imagePerKind[node.kind]

  useEffect(() => {
    console.log("nodes : " + JSON.stringify(props.nodes));
    console.log("typeof nodes : " + (typeof props.nodes));
    const graph = new Graph();
    props.nodes.forEach(node => {
      graph.addNode(node.key, { x: Math.random(), y: Math.random(), size: 10,
label: node.itemId })
    });
    props.edges.forEach(edge => {
      graph.addEdge(edge.source, edge.target)
    });
    loadGraph(graph);
    assign();
  }, [assign, loadGraph, props.nodes, props.relations]);

  return null;

  // circular.assign(graph);
  //   const settings = forceAtlas2.inferSettings(graph);
  //   forceAtlas2.assign(graph, { settings, iterations: 600 });
};
```

```
export default function NetworkGraph() {
  return (
    <div>
      {/* <kubernetesTopologyGraph items="items" relations="relations"
kinds="kinds">
      </kubernetesTopologyGraph> */}
      <SigmaContainer style={{ height: "70vh", width: "100%" }}>
        <LoadGraph nodes={items} edges={relations} />
      </SigmaContainer>
    </div>
  );
}
```

src/NetworksTable.js

```
import React from 'react';
import BaseTable from './BaseTable';

const columns = [
  { id: 'name', label: 'Name', minWidth: 170 }
];

function createData(name, code) {
  return { name, code };
}

const rows = [
  createData('Network 1', 1),
  createData('Network 2', 2),
  createData('Network 3', 3),
  createData('Network 4', 4),
  createData('Network 5', 5),
  createData('Network 6', 6)
];


export default function ComponentStatusTable() {
  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/NetworksTableContainer.js

```
import React from 'react';
```

```
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import NetworksTable from './NetworksTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function NetworksTableContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Networks
      </Typography>

      <NetworksTable />
    </div>
  );
}
```

src/PageTitle.js

```
import React from 'react';

export default function PageTitle(props) {
  return (
    <h1>{props.titleText}</h1>
  );
}
```

src/RankContainer.js

```
import { Paper, Typography } from '@mui/material';
import React, { useState, useEffect } from 'react';

export default function RankContainer(props) {

  return (
    <Paper elevation={3} style={{padding:'20px 16px'}}>
      <div variant='h5'>{props.title} : {props.rank}</div>
```

```
    </Paper>
  );
}
```

src/Reports.js

```
import React from 'react';
import ReportTable from './ReportTable';
import PageTitle from './PageTitle';

export default function Scenarios() {
  return (
    <div>
      <PageTitle titleText="Reports" />
      <ReportTable />
    </div>
  );
}
```

src/ReportTable.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { DataGrid } from '@mui/x-data-grid';
import { Download } from '@mui/icons-material';
import IconButton from '@mui/material/IconButton';
import { getReports, sendDownloadReportRequest } from './ServerCaller';
import CtrlSnackBar from './CtrlSnackBar';

function createData(idValue, nameValue, descriptionValue, createdAtValue) {
  return {
    id: idValue, name: nameValue,
    description: descriptionValue, createdAt: createdAtValue,
    download: idValue
  };
}

function createRows(data) {
  let rowList = [];
  data.forEach(r => {
    rowList.push(createData(r.id, r.name, r.description, r.isoCreatedAt));
  });
  console.log('rowList : ' + JSON.stringify(rowList));
  return rowList;
}
```

```
function downloadReport(reportId, handleRunError) {
  console.log("report to download : " + JSON.stringify(reportId));
    sendDownloadReportRequest(reportId, null, handleRunError);
}

function createDownloadButton(reportId, handleRunError) {
  return (
    <IconButton color="primary" onClick={(event) => { downloadReport(reportId,
handleRunError); }}>
      <Download />
    </IconButton>
  );
}

function createColumns(handleRunError) {
  let cols = [
    { field: 'name', headerName: 'Name', flex: 1, sortable: true },
    { field: 'description', headerName: 'Description', flex: 1, sortable:
false },
    { field: 'createdAt', headerName: 'Created At', flex: 1, sortable: false
},
    {
      field: 'download', headerName: 'Download', flex: 1, sortable: false,
      type: 'actions',
      renderCell: (reportId) => createDownloadButton(reportId, handleRunError)
    },
  ];

  return cols;
}

export default function ReportTable() {
  const [snackBarOpen, setSnackBarOpen] = useState({
    open: false,
    msg: undefined
  });

  const [tableState, setTableState] = useState({
    loading: false,
    rows: [],
  });

  const handleRunError = (serverErrorResponse) => {
    let message = serverErrorResponse.response.data;
    // TODO setSnackBarOpen({ open: true, msg: message });
  }

  const columns = createColumns(handleRunError);
```

```
  useEffect(() => {
    let mounted = true;
    setTableState({ loading: true });

    function updateTableState(reports) {
      if (mounted) {
        setTableState({ loading: false, rows: createRows(reports) });
      }
    }

    getReports(updateTableState, null);

    return function cleanup() {
      mounted = false;
    }
  }, [setTableState]);

  if (tableState.loading) {
    return 'Loading...';
  }
  return (
    <div style={{ width: '100%' }}>
      <CtrlSnackBar />
      <DataGrid rows={tableState.rows} columns={columns} autoPageSize={true}
        autoHeight={true} showColumnRightBorder={true}
showCellRightBorder={true}
        initialState={{ pinnedColumns: { right: ['download'] } }} />
    </div>
  );
}
```

src/ScenarioNetworks.js

```
import React, { useState, useEffect } from 'react';
import CtrlDropDown from './CtrlDropDown';
import NetworkGraph from './NetworkGraph';
import { getScenarios } from './ServerCaller';
import PageTitle from './PageTitle';
// import kubernetesTopologyGraph from 'kubernetesUI';

function scenariosDropDownTextDisplayer(scenario) {
  return scenario.title;
}

export default function ScenarioNetworks() {
  const [scenarios, setScenarios] = useState([]);
```

```
  useEffect(() => {
    function updateScenarios(scenarios) {
      setScenarios(scenarios);
    }

    getScenarios(updateScenarios, null);
  }, [setScenarios]);

  return (
    <div>
      <PageTitle titleText='Scenario Networks' />
      <CtrlDropDown title="Scenario" options={scenarios}
textDisplayer={scenariosDropDownTextDisplayer} />
      <NetworkGraph></NetworkGraph>
    </div>
  );
}
```

src/Scenarios.js

```
import React from 'react';
import ScenariosTable2 from './ScenariosTable2';
import PageTitle from './PageTitle';

export default function Scenarios() {
  return (
    <div>
      <PageTitle titleText="Running Scenarios" />
      <ScenariosTable2 />
    </div>
  );
}
```

src/ScenariosTable2.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { DataGrid } from '@mui/x-data-grid';
import Link from '@material-ui/core/Link';
import axios from 'axios'
import Box from '@material-ui/core/Box';
```

```
function createDescriptionCellContent(descriptionValue, linkValue) {
  if (linkValue) {
    return (
      <div style={{ lineHeight: "initial" }}>
          <Box component="span" textOverflow="ellipsis">
            {descriptionValue}
          </Box>
          <Link style={{ display: "block" }} href={linkValue} target="_blank"
rel="noopener">read more</Link>
      </div>
    );
  } else {
    return (
      <div>
        {descriptionValue}
      </div>
    );
  }
}

function createData(nameValue, idValue, descriptionValue, linkValue) {
  return {
    name: nameValue, id: idValue, description:
      { descriptionText: descriptionValue, linkText: linkValue }
  };
}

function createRows(data) {
  let rowList = [];
  data.forEach(r => {
    rowList.push(createData(r.title, r.scenario_id, r.description, r.link));
  });
  console.log('rowList : ' + JSON.stringify(rowList));
  return rowList;
}

const columns = [
  { field: 'name', headerName: 'Name', flex: 1, sortable: true },
  { field: 'id', headerName: 'Id', flex: 1, sortable: true },
  {
    field: 'description', headerName: 'Description', flex: 1, sortable: false,
    renderCell: (params) =>
createDescriptionCellContent(params.value.descriptionText,
params.value.linkText)
  },
];

export default function ScenariosTable2() {
```

```
  const [tableState, setTableState] = useState({
    loading: false,
    rows: [],
  });

  useEffect(() => {
    let mounted = true;
    setTableState({ loading: true });
    const apiUrl = '/api/scenarios';
    axios.get(apiUrl).then((result) => {
      console.log('scenarios call result : ' + JSON.stringify(result));
      if (mounted) {
        const scenarios = result.data;
        setTableState({ loading: false, rows: createRows(scenarios) });
      }
    })
      .catch((error) => {
        if (error.response) {
          // The request was made and the server responded with a status code
          // that falls out of the range of 2xx
          console.log(error.response.data);
          console.log(error.response.status);
          console.log(error.response.headers);
        } else if (error.request) {
          // The request was made but no response was received
          // `error.request` is an instance of XMLHttpRequest in the browser
and an instance of
          // http.ClientRequest in node.js
          console.log(error.request);
        } else {
          // Something happened in setting up the request that triggered an
Error
          console.log('Error', error.message);
        }
        console.log(error.config);
      });

    return function cleanup() {
      mounted = false;
    }
  }, [setTableState]);

  if (tableState.loading) {
    return 'Loading...';
  }
  return (
    <div style={{ width: '100%' }}>
```

```
      <DataGrid rows={tableState.rows} columns={columns} autoPageSize={true}
autoHeight={true} showColumnRightBorder={true} showCellRightBorder={true} />
    </div>
  );
}
```

src/ScenarioTeams.js

```
import React, { useEffect, useState } from 'react';
import PageTitle from './PageTitle';
import CtrlDropDown from './CtrlDropDown';
import Grid from '@material-ui/core/Grid';
import EventsBasedOnNumberChart from './EventsBasedOnNumberChart';
import { getScenarios, getScenarioTeams, getScenarioTeamUsers,
getEventsPerHour } from './ServerCaller';

function scenariosDropDownTextDisplayer(scenario) {
  // return scenario.scenario_id + " - " + scenario.title;
  return scenario.title;
}

function teamsDropDownTextDisplayer(team) {
  // return team.team_id + " - " + team.name;
  return team.name;
}

function usersDropDownTextDisplayer(user) {
  // return user.user_id + " - " + user.username;
  return user.username;
}

export default function ScenarioTeams(props) {
  const [selectedScenario, setSelectedScenario] = useState();
  const [selectedTeam, setSelectedTeam] = useState();
  const [selectedUser, setSelectedUser] = useState();
  const [scenarios, setScenarios] = useState([]);
  const [teams, setTeams] = useState([]);
  const [users, setUsers] = useState([]);
  const [chartData, setChartData] = useState([]);

  const title = "Teams participating in scenario";
  const allOptionValue = "--all--";

  useEffect(() => {
    function updateScenarios(scenarios) {
      setScenarios(scenarios);
      if (scenarios && scenarios.length)
```

```
        setSelectedScenario(scenarios[0].id);
    }

    getScenarios(updateScenarios, null);
  }, []);

  useEffect(() => {
    console.log("selected scenario changed. will fetch teams for scenarioId :
" + selectedScenario);
    setTeams([])
    setSelectedTeam(null);
    setChartData([]);
    function updateTeams(teams) {
      setTeams(teams);
      if (teams && teams.length)
        setSelectedTeam(allOptionValue);
    }

    if (selectedScenario)
      getScenarioTeams(selectedScenario, updateTeams, null);
  }, [selectedScenario]);

  useEffect(() => {
    console.log("selected team changed. will fetch users for scenarioId,
teamId : " + selectedScenario + ", " + selectedTeam);
    setUsers([])
    setSelectedUser(null);
    setChartData([]);
    function updateUsers(users) {
      setUsers(users);
      if (users && users.length)
        setSelectedUser(users[0].id);
    }
    function updateChartData(data) {
      setChartData(data);
    }

    if (selectedScenario && selectedTeam) {
      getScenarioTeamUsers(selectedScenario, selectedTeam, updateUsers, null);
      getEventsPerHour(selectedScenario, selectedTeam, null, updateChartData,
null);
    }
  }, [selectedScenario, selectedTeam]);

  useEffect(() => {
    console.log("selected user changed. will fetch chart data : " +
selectedScenario + ", " + selectedTeam);
    setChartData([]);
```

```
    function updateChartData(data) {
      setChartData(data);
    }

    if (selectedScenario && selectedTeam && selectedUser) {
      let userId = selectedUser;
      if (allOptionValue === userId) { userId = null; }
      getEventsPerHour(selectedScenario, selectedTeam, userId,
updateChartData, null);
    }
  }, [selectedScenario, selectedTeam, selectedUser]);

  const scenarioChangeHandler = (event) => {
    setSelectedScenario(event.target.value);
  };

  const teamChangeHandler = (event) => {
    setSelectedTeam(event.target.value);
  };

  const userChangeHandler = (event) => {
    setSelectedUser(event.target.value);
  };

  return (
    <div style={props.style}>
      <PageTitle titleText={title} />
      <Grid
        container
        direction="row"
        justifyContent="center"
        alignItems="center"
        spacing={3}
      >
        <Grid item xs>
          <CtrlDropDown title="Scenario" options={scenarios}
textDisplayer={scenariosDropDownTextDisplayer}
changeHandler={scenarioChangeHandler} />
        </Grid>
        <Grid item xs>
          <CtrlDropDown title="Team" options={teams}
textDisplayer={teamsDropDownTextDisplayer} changeHandler={teamChangeHandler}
/>
        </Grid>
        <Grid item xs>
          <CtrlDropDown title="User" options={users}
textDisplayer={usersDropDownTextDisplayer} changeHandler={userChangeHandler}
allOptionValue={allOptionValue} />
```

```
        </Grid>
      </Grid>
      <EventsBasedOnNumberChart data={chartData} />
    </div>
  );
}
```

src/ScenarioTeamUserSelection.js

```
import React, { useEffect, useState } from 'react';
import CtrlDropDown from './CtrlDropDown';
import Grid from '@material-ui/core/Grid';
import { getScenarios, getScenarioTeams, getScenarioTeamUsers } from
'./ServerCaller';

function scenariosDropDownTextDisplayer(scenario) {
  // return scenario.scenario_id + " - " + scenario.title;
  return scenario.title;
}

function teamsDropDownTextDisplayer(team) {
  // return team.team_id + " - " + team.name;
  return team.name;
}

function usersDropDownTextDisplayer(user) {
  // return user.user_id + " - " + user.username;
  return user.username;
}

export default function ScenarioTeamUserSelection(props) {
  const allOptionValue = "--all--";

  const [scenarios, setScenarios] = useState([]);
  const [teams, setTeams] = useState([]);
  const [users, setUsers] = useState([]);

  const [selectedScenario, setSelectedScenario] = useState();
  const [selectedTeam, setSelectedTeam] = useState();
  const [selectedUser, setSelectedUser] = useState();

  useEffect(() => {
    function updateScenarios(scenarios) {
      setScenarios(scenarios);
      if (scenarios && scenarios.length)
        setSelectedScenario(scenarios[0].id);
    }
```

```
    getScenarios(updateScenarios, null);
  }, []);

  useEffect(() => {
    console.log("selected scenario changed. will fetch teams for scenarioId :
" + selectedScenario);
    setTeams([])
    setSelectedTeam(null);

    function updateTeams(teams) {
      setTeams(teams);
      if (teams && teams.length)
        setSelectedTeam(allOptionValue);
    }

    if (selectedScenario)
      getScenarioTeams(selectedScenario, updateTeams, null);
  }, [selectedScenario]);

  useEffect(() => {
    console.log("selected team changed. will fetch users for scenarioId,
teamId : " + selectedScenario + ", " + selectedTeam);
    setUsers([])
    setSelectedUser(null);

    function updateUsers(users) {
      setUsers(users);
      if (users && users.length)
        setSelectedUser(users[0].id);
    }

    if (selectedScenario && selectedTeam) {
      getScenarioTeamUsers(selectedScenario, selectedTeam, updateUsers, null);
    }
  }, [selectedScenario, selectedTeam]);

  const scenarioChangeHandler = (event) => {
    setSelectedScenario(event.target.value);
  };

  const teamChangeHandler = (event) => {
    setSelectedTeam(event.target.value);
  };

  const userChangeHandler = (event) => {
    setSelectedUser(event.target.value);
  };
```

```
  return (
    <div style={props.style}>
      <Grid
        container
        direction="row"
        justifyContent="center"
        alignItems="center"
        spacing={3}
      >
        <Grid item xs>
          <CtrlDropDown title="Scenario" options={scenarios}
textDisplayer={scenariosDropDownTextDisplayer}
changeHandler={scenarioChangeHandler} />
        </Grid>
        <Grid item xs>
          <CtrlDropDown title="Team" options={teams}
textDisplayer={teamsDropDownTextDisplayer} changeHandler={teamChangeHandler}
/>
        </Grid>
        <Grid item xs>
          <CtrlDropDown title="User" options={users}
textDisplayer={usersDropDownTextDisplayer} changeHandler={userChangeHandler}
allOptionValue={allOptionValue} />
        </Grid>
      </Grid>
    </div>
  );
}
```

src/SecurityEventsContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import SecurityEventsTable from './SecurityEventsTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function SecurityEventsContainer(props) {
  const classes = useStyles();
```

```
  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        {props.title}
      </Typography>

      <SecurityEventsTable data={props.eventsData}/>
    </div>
  );
}
```

src/SecurityEventsTable.js

```
import React from 'react';
import BaseTable from './BaseTable';

const columns = [
  { id: 'eventName', label: 'Event name', minWidth: 170 },
  { id: 'startTime', label: 'Start time', minWidth: 170 },
  {
    id: 'source',
    label: 'Source',
    minWidth: 170,
  },
  {
    id: 'destination',
    label: 'Destination',
    minWidth: 170,
  },
  {
    id: 'riskLevel',
    label: 'Risk level',
    minWidth: 170,
  },
];

export default function SecurityEventsTable(props) {
  return (
    <BaseTable columns={columns} rows={props.data} />
  );
}
```

src/ServerCaller.js

```
import React, { useState, useEffect } from 'react';
```

```
import axios from 'axios'

export function getScenarios(successHandler, errorHandler) {
  console.log("Sending get scenarios request..");

  const apiUrl = '/api/scenarios';
  axios.get(apiUrl).then((result) => {
    console.log('scenarios call result : ' + JSON.stringify(result));
    successHandler(result.data)
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    });
}

export function getScenarioTeams(scenarioIdValue, successHandler,
errorHandler) {
  console.log("Sending get scenario teams request..");

  const apiUrl = '/api/teams';
  axios.get(apiUrl, { params: { scenarioId: scenarioIdValue } }).then((result)
=> {
    console.log('scenario teams call result : ' + JSON.stringify(result));
    successHandler(result.data)
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    });
}

export function getScenarioTeamUsers(scenarioIdValue, teamIdValue,
successHandler, errorHandler) {
  console.log("Sending get scenario team users request..");

  const apiUrl = '/api/users';
  axios.get(apiUrl, { params: { scenarioId: scenarioIdValue, teamId:
teamIdValue } }).then((result) => {
    console.log('scenario team users call result : ' +
JSON.stringify(result));
    successHandler(result.data)
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
```

```
        errorHandler(error)
    });
}

export function getEventsPerHour(scenarioIdValue, teamIdValue, userIdValue,
successHandler, errorHandler) {
  console.log("Sending get events per hour request..");

  const apiUrl = '/api/event';
  axios.get(apiUrl, { params: { scenarioId: scenarioIdValue, teamId:
teamIdValue, userId: userIdValue } }).then((result) => {
    console.log('get events per hour call result : ' +
JSON.stringify(result));
    successHandler(result.data)
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    });
}

export function getScripts(successHandler, errorHandler) {
  console.log("Sending get scripts request..");

  const apiUrl = '/api/scripts';
  axios.get(apiUrl).then((result) => {
    console.log('scripts call result : ' + JSON.stringify(result));
    successHandler(result.data)
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    });
}


export function sendRunScriptRequest(scriptId, variableValues, successHandler,
errorHandler) {
  console.log("sendRunScriptRequest");
  console.log(scriptId);
  console.log(variableValues);

  const apiUrl = '/api/scripts';
  axios.post(apiUrl, {
    "scriptId": scriptId,
    "variableValues": variableValues
  }).then((result) => {
```

```
    console.log('scripts call result : ' + JSON.stringify(result));
    successHandler(result.data);
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    })
}

export function handleResponseError(error) {
  if (error.response) {
    // The request was made and the server responded with a status code
    // that falls out of the range of 2xx
    console.log(error.response.data);
    console.log(error.response.status);
    console.log(error.response.headers);
  } else if (error.request) {
    // The request was made but no response was received
    // `error.request` is an instance of XMLHttpRequest in the browser and an
instance of
    // http.ClientRequest in node.js
    console.log(error.request);
  } else {
    // Something happened in setting up the request that triggered an Error
    console.log('Error', error.message);
  }
  console.log(error.config);
}

export function getReports(successHandler, errorHandler) {
  console.log("Sending get reports request..");

  const apiUrl = '/api/report';
  axios.get(apiUrl).then((result) => {
    console.log('scripts call result : ' + JSON.stringify(result));
    successHandler(result.data)
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    });
}

export function sendDownloadReportRequest(reportId, successHandler,
errorHandler) {
  console.log("sendDownloadReportRequest");
```

```
  console.log(reportId);

  const apiUrl = '/api/report';
 /*  axios.get(apiUrl, {
    "reportId": reportId
  }).then((result) => {
    console.log('report download call result : ' + JSON.stringify(result));
    successHandler(result.data);
  })
    .catch((error) => {
      handleResponseError(error)
      if (errorHandler)
        errorHandler(error)
    }) */
}
```

src/Skeleton.js

```
import React from 'react';
import clsx from 'clsx';
import { useTheme } from '@material-ui/core/styles';
import Drawer from '@material-ui/core/Drawer';
import AppBar from '@material-ui/core/AppBar';
import Toolbar from '@material-ui/core/Toolbar';
import List from '@material-ui/core/List';
import CssBaseline from '@material-ui/core/CssBaseline';
import Typography from '@material-ui/core/Typography';
import Divider from '@material-ui/core/Divider';
import IconButton from '@material-ui/core/IconButton';
import MenuIcon from '@mui/icons-material/Menu';
import ChevronLeftIcon from '@mui/icons-material/ChevronLeft';
import ChevronRightIcon from '@mui/icons-material/ChevronRight';
import ListItem from '@material-ui/core/ListItem';
import ListItemIcon from '@material-ui/core/ListItemIcon';
import ListItemText from '@material-ui/core/ListItemText';
import DashboardIcon from '@mui/icons-material/Dashboard';
import ScoreIcon from '@mui/icons-material/Score';
import TimelineIcon from '@mui/icons-material/Timeline';
import FlagIcon from '@mui/icons-material/Flag';
import CalendarTodayIcon from '@mui/icons-material/CalendarToday';
import AssessmentIcon from '@mui/icons-material/Assessment';
import GroupIcon from '@mui/icons-material/Group';
import PersonIcon from '@mui/icons-material/Person';
import RouterIcon from '@mui/icons-material/Router';
import { Switch, Route } from 'react-router-dom';
import Calendar from './Calendar';
import Campaign from './Campaign';
```

```
import IndividualScoring from './individualScore/IndividualScoring';
import ScenarioNetworks from './ScenarioNetworks';
import Reports from './Reports';
import Scenarios from './Scenarios';
import Scoring from './score/Scoring';
import TeamScoring from './teamScore/TeamScoring';
import Timeline from './Timeline';
import { withRouter } from "react-router-dom";
import { useStyles } from './DrawerStyles.js'
import ScriptManagement from './script/ScriptManagement';
import { Code } from '@mui/icons-material';
import ScenarioTeams from './ScenarioTeams';

function MiniDrawer(props) {
  const { history } = props;
  const classes = useStyles();
  const theme = useTheme();
  const [open, setOpen] = React.useState(false);

  const handleDrawerOpen = () => {
    setOpen(true);
  };

  const handleDrawerClose = () => {
    setOpen(false);
  };

  const itemsList = [
    {
      text: "Scenarios",
      icon: <DashboardIcon />,
      path: "/scenarios",
      component: <Scenarios />,
      exact: true,
      onClick: () => history.push("/scenarios"),
      uiClass: ''
    },
    {
      text: "Networks",
      icon: <RouterIcon />,
      path: "/scenarios/networks",
      component: <ScenarioNetworks />,
      exact: false,
      onClick: () => history.push("/scenarios/networks"),
      uiClass: classes.nested
    },
    {
      text: "Teams",
```

```
      icon: <GroupIcon />,
      path: "/scenarios/teams",
      component: <ScenarioTeams />,
      exact: false,
      onClick: () => history.push("/scenarios/teams"),
      uiClass: classes.nested
    },
    {
      text: "Timeline",
      icon: <TimelineIcon />,
      path: "/timeline",
      component: <Timeline />,
      exact: false,
      onClick: () => history.push("/timeline"),
      uiClass: ''
    },
    {
      text: "Scoring",
      icon: <ScoreIcon />,
      path: "/scoring",
      component: <Scoring />,
      exact: true,
      onClick: () => history.push("/scoring"),
      uiClass: ''
    },
    {
      text: "Team-based",
      icon: <GroupIcon />,
      path: "/scoring/team",
      component: <TeamScoring />,
      exact: false,
      onClick: () => history.push("/scoring/team"),
      uiClass: classes.nested
    },
    {
      text: "Individual",
      icon: <PersonIcon />,
      path: "/scoring/individual",
      exact: false,
      component: <IndividualScoring />,
      onClick: () => history.push("/scoring/individual"),
      uiClass: classes.nested
    },
    {
      text: "Scenario Campaign",
      icon: <FlagIcon />,
      path: "/campaign",
      component: <Campaign />,
```

```
    exact: false,
    onClick: () => history.push("/campaign"),
    uiClass: ''
  },
  {
    text: "Calendar",
    icon: <CalendarTodayIcon />,
    path: "/calendar",
    component: <Calendar />,
    exact: false,
    onClick: () => history.push("/calendar"),
    uiClass: ''
  },
  {
    text: "Visualisation Reports",
    icon: <AssessmentIcon />,
    path: "/reports",
    component: <Reports />,
    exact: false,
    onClick: () => history.push("/reports"),
    uiClass: ''
  },
  {
    text: "Script Management",
    icon: <Code />,
    path: "/scripts",
    component: <ScriptManagement />,
    exact: false,
    onClick: () => history.push("/scripts"),
    uiClass: ''
  }
];

return (
  <div className={classes.root}>
    <CssBaseline />
    <AppBar
      position="fixed"
      className={clsx(classes.appBar, {
        [classes.appBarShift]: open,
      })}
    >
      <Toolbar>
        <IconButton
          color="inherit"
          aria-label="open drawer"
          onClick={handleDrawerOpen}
          edge="start"
```

```
            className={clsx(classes.menuButton, {
              [classes.hide]: open,
            })}
          >
            <MenuIcon />
          </IconButton>
          <Typography variant="h6" noWrap>
            CTRL Admin Console
          </Typography>
        </Toolbar>
      </AppBar>
      <Drawer
        variant="permanent"
        className={clsx(classes.drawer, {
          [classes.drawerOpen]: open,
          [classes.drawerClose]: !open,
        })}
        classes={{
          paper: clsx({
            [classes.drawerOpen]: open,
            [classes.drawerClose]: !open,
          }),
        }}
      >
        <div className={classes.toolbar}>
          <IconButton onClick={handleDrawerClose}>
            {theme.direction === 'rtl' ? <ChevronRightIcon /> :
<ChevronLeftIcon />}
          </IconButton>
        </div>
        <Divider />
        <List>
          {itemsList.map((item, index) => {
            const { text, icon, onClick, uiClass } = item;
            return (
              <ListItem button key={text} onClick={onClick}
className={uiClass}>
                {icon && <ListItemIcon>{icon}</ListItemIcon>}
                <ListItemText primary={text} />
              </ListItem>
            );
          })}
        </List>
      </Drawer>
      <main className={classes.content}>
        <div className={classes.toolbar} />
        <Switch>
          {itemsList.map((route, index) => (
```

```
            <Route
              key={index}
              path={route.path}
              exact={route.exact}
            >
              {route.component}
            </Route>
          ))}
        </Switch>
      </main>
    </div>
  );
}

export default withRouter(MiniDrawer);
```

src/Timeline.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import TimelineChart from './TimelineChart';
import SecurityEventsContainer from './SecurityEventsContainer';
import TopEventsContainer from './TopEventsContainer';
import ComponentStatusContainer from './ComponentStatusContainer';

const useStyles = makeStyles((theme) => ({
  container: {
    display: 'grid',
    columnGap: '10px',
    gridColumnGap: '10px',
    rowGap: '10px',
    gridRowGap: '10px',
    justifyItems: 'stretch',
    alignItems: 'stretch',
    gridTemplateAreas:
      ` 'timeline timeline topEvents'
       'securityEvents securityEvents componentStatus'`,
  },
  timeline: {
    gridArea: 'timeline'
  },
  topEvents: {
    gridArea: 'topEvents'
  },
  securityEvents: {
    gridArea: 'securityEvents'
  },
```

```
  componentStatus: {
    gridArea: 'componentStatus'
  },
}));

function createData(eventName, startTime, source, destination, riskLevel,
code) {
  return { eventName, startTime, source, destination, riskLevel, code };
}

const rows = [
  createData('Event 1', '2020-01-01T16:50:43', 'Source 1', 'Destination 1',
'Risk Level 1', 1),
  createData('Event 2', '2020-01-01T16:51:43', 'Source 2', 'Destination 2',
'Risk Level 2', 2),
  createData('Event 3', '2020-01-01T16:52:43', 'Source 3', 'Destination 3',
'Risk Level 3', 3),
  createData('Event 4', '2020-01-01T16:53:43', 'Source 4', 'Destination 4',
'Risk Level 4', 4),
  createData('Event 5', '2020-01-01T16:54:43', 'Source 5', 'Destination 5',
'Risk Level 5', 5),
  createData('Event 6', '2020-01-01T16:55:43', 'Source 6', 'Destination 6',
'Risk Level 1', 6),
  createData('Event 7', '2020-01-01T16:56:43', 'Source 7', 'Destination 7',
'Risk Level 2', 7),
  createData('Event 8', '2020-01-01T16:57:43', 'Source 8', 'Destination 8',
'Risk Level 3', 8),
  createData('Event 9', '2020-01-01T16:58:43', 'Source 9', 'Destination 9',
'Risk Level 4', 9),
  createData('Event 10', '2020-01-01T16:59:43', 'Source 10', 'Destination 10',
'Risk Level 5', 10),
  createData('Event 11', '2020-01-01T17:00:43', 'Source 11', 'Destination 11',
'Risk Level 1', 11),
  createData('Event 12', '2020-01-01T17:01:43', 'Source 12', 'Destination 12',
'Risk Level 2', 12),
  createData('Event 13', '2020-01-01T17:02:43', 'Source 13', 'Destination 13',
'Risk Level 3', 13),
  createData('Event 14', '2020-01-01T17:03:43', 'Source 14', 'Destination 14',
'Risk Level 4', 14),
];

export default function Timeline() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <div className={classes.timeline}>
        <TimelineChart />
```

```
      </div>
      <div className={classes.topEvents}>
        <TopEventsContainer />
      </div>
      <div className={classes.securityEvents}>
        <SecurityEventsContainer title="Important Security Events"
eventsData={rows}/>
      </div>
      <div className={classes.componentStatus}>
        <ComponentStatusContainer />
      </div>
    </div>
  );
}
```

src/TimelineChart.js

```
import React, { useState } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import { ScatterChart, Scatter, Tooltip, XAxis, YAxis, ZAxis,
ResponsiveContainer } from 'recharts';

const data = [
  {
    time: 0, eventsNo: 10, index: 1,
  },
  {
    time: 1, eventsNo: 300, index: 1,
  },
  {
    time: 2, eventsNo: 200, index: 1,
  },
  {
    time: 3, eventsNo: 278, index: 1,
  },
  {
    time: 4, eventsNo: 189, index: 1,
  },
  {
    time: 5, eventsNo: 239, index: 1,
  },
  {
    time: 6, eventsNo: 349, index: 1,
  },
  {
    time: 7, eventsNo: 45, index: 1,
```

```
  },
];

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

function renderTooltip(active, payload) {
  if (active && payload && payload.length) {
    const data = payload[0] && payload[0].payload;

    return (
      <div
        style={{
          backgroundColor: '#fff',
          border: '1px solid #999',
          margin: 0,
          padding: 10,
        }}
      >
        <p>{data.hour}</p>
        <p>
          <span>value: </span>
          {data.value}
        </p>
      </div>
    );
  }

  return null;
};

function parseDomain() {
  return [
    0,
    Math.max(
      Math.max.apply(
        null,
        data.map((entry) => entry.value),
      ),
      Math.max.apply(
        null,
        data.map((entry) => entry.value),
      ),
    ),
```

```
  ]
};

export default function TimelineChart(props) {
  const classes = useStyles();
  const { active, payload } = props;
  // const [data, setData] = useState([]);

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Timeline View
      </Typography>
      <ResponsiveContainer width='100%' height={300}>
        <ScatterChart
          margin={{
            top: 30,
            right: 20,
            bottom: 0,
            left: 0,
          }}
        >
          <XAxis
            type="number"
            dataKey="time"
            name="time"
            interval={0}
            tickLine={{ transform: 'translate(0, -6)' }}
          />
          <YAxis
            type="number"
            dataKey="index"
            height={10}
            width={80}
            tick={false}
            tickLine={false}
            axisLine={false}
            domain={[0,2]}
          />
          <ZAxis type="number" dataKey="eventsNo" domain={parseDomain()}
range={[10, 300]}/>
          <Tooltip cursor={{ strokeDasharray: '3 3' }} wrapperStyle={{ zIndex:
100 }} content={renderTooltip(active, payload)} />
          <Scatter data={data} fill="#8884d8" />
        </ScatterChart>
      </ResponsiveContainer>
    </div>
  );
```

```
}
```

src/TopEventsContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import TopEventsTable from './TopEventsTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function TopEventsContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Top events by event count
      </Typography>

      <TopEventsTable />
    </div>
  );
}
```

src/TopEventsTable.js

```
import React from 'react';
import BaseTable from './BaseTable';

const columns = [
  { id: 'event', label: 'Event', minWidth: 170 },
  { id: 'sessions', label: 'Sessions', minWidth: 100 },
];

function createData(event, sessions, code) {
  return { event, sessions, code };
}

const rows = [
```

```
  createData('Application activity', '40', 1),
  createData('Packet tagging', '50', 2),
  createData('Invalid tags', '60', 3),
  createData('Application activity', '40', 4),
  createData('Packet tagging', '50', 5),
  createData('Invalid tags', '60', 6),
  createData('Application activity', '40', 7),
  createData('Packet tagging', '50', 8),
  createData('Invalid tags', '60', 9),
  createData('Application activity', '40', 10),
  createData('Packet tagging', '50', 11),
  createData('Invalid tags', '60', 12),
  createData('Application activity', '40', 13),
  createData('Packet tagging', '50', 14),
  createData('Invalid tags', '60', 15),
];

export default function TopEventsTable() {

  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/individualScore/individualScoreChart.js

```
import React from 'react';
import {
   ResponsiveContainer, BarChart, Bar, XAxis, YAxis, Tooltip
} from 'recharts';

const data = [
  {
    time: '2020-01-01T10:00:00', score: 4000
  },
  {
    time: '2020-01-01T11:00:00', score: 3000
  },
  {
    time: '2020-01-01T12:00:00', score: 2000
  },
  {
    time: '2020-01-01T13:00:00', score: 2780
  },
  {
    time: '2020-01-01T14:00:00', score: 1890
  },
```

```
  {
    time: '2020-01-01T15:00:00', score: 2390
  },
  {
    time: '2020-01-01T16:00:00', score: 3490
  },
];

export default function IndividualScoreChart() {
    return (
    <ResponsiveContainer width={'100%'} height={300}>
      <BarChart
        data={data}
      >
        <XAxis dataKey="time" />
        <YAxis />
        <Tooltip />
        <Bar dataKey="score" name="Score" fill="#8884d8" />
      </BarChart>
       </ResponsiveContainer>
    );
}
```

src/individualScore/individualScoreContainer.js

```
import React from 'react';
import Typography from '@material-ui/core/Typography';
import IndividualScoreChart from './IndividualScoreChart';

export default function IndividualScoreContainer(props) {
  return (
    <div style={props.style}>
      <Typography variant="overline" component="div" align="center">
        Scores awarded
      </Typography>

      <IndividualScoreChart />
    </div>
  );
}
```

src/individualScore/individualScorePerEventContainer.js

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import IndividualScorePerEventTable from './IndividualScorePerEventTable';
```

```
const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function IndividualScorePerEventContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Score Per Event
      </Typography>

      <IndividualScorePerEventTable />
    </div>
  );
}
```

src/individualScore/individualScorePerEventTable.js

```
import React from 'react';
import BaseTable from '../BaseTable';

const columns = [
  { id: 'eventName', label: 'Event name', minWidth: 170 },
  { id: 'startTime', label: 'Start time', minWidth: 170 },
  {
    id: 'source',
    label: 'Source',
    minWidth: 170,
  },
  {
    id: 'destination',
    label: 'Destination',
    minWidth: 170,
  },
  {
    id: 'riskLevel',
    label: 'Risk level',
    minWidth: 170,
  },
  { id: 'user', label: 'User', minWidth: 170 },
  { id: 'score', label: 'Score', minWidth: 170 },
```

```
];

function createData(eventName, startTime, source, destination, riskLevel,
user, score, code) {
  return { eventName, startTime, source, destination, riskLevel, user, score,
code };
}

const rows = [
  createData('Event 1', '2020-01-01T16:50:43', 'Source 1', 'Destination 1',
'Risk Level 1', 'User A', '150', 1),
  createData('Event 2', '2020-01-01T16:51:43', 'Source 2', 'Destination 2',
'Risk Level 2', 'User B', '786', 2),
  createData('Event 3', '2020-01-01T16:52:43', 'Source 3', 'Destination 3',
'Risk Level 3', 'User C', '234', 3),
  createData('Event 4', '2020-01-01T16:53:43', 'Source 4', 'Destination 4',
'Risk Level 4', 'User D', '79834', 4),
  createData('Event 5', '2020-01-01T16:54:43', 'Source 5', 'Destination 5',
'Risk Level 5', 'User A', '1325', 5),
  createData('Event 6', '2020-01-01T16:55:43', 'Source 6', 'Destination 6',
'Risk Level 1', 'User B', '685', 6),
  createData('Event 7', '2020-01-01T16:56:43', 'Source 7', 'Destination 7',
'Risk Level 2', 'User C', '435', 7),
  createData('Event 8', '2020-01-01T16:57:43', 'Source 8', 'Destination 8',
'Risk Level 3', 'User D', '24785', 8),
  createData('Event 9', '2020-01-01T16:58:43', 'Source 9', 'Destination 9',
'Risk Level 4', 'User A', '234', 9),
  createData('Event 10', '2020-01-01T16:59:43', 'Source 10', 'Destination 10',
'Risk Level 5', 'User B', '8354', 10),
  createData('Event 11', '2020-01-01T17:00:43', 'Source 11', 'Destination 11',
'Risk Level 1', 'User C', '54696', 11),
  createData('Event 12', '2020-01-01T17:01:43', 'Source 12', 'Destination 12',
'Risk Level 2', 'User D', '1342', 12),
  createData('Event 13', '2020-01-01T17:02:43', 'Source 13', 'Destination 13',
'Risk Level 3', 'User A', '3150', 13),
  createData('Event 14', '2020-01-01T17:03:43', 'Source 14', 'Destination 14',
'Risk Level 4', 'User B', '15320', 14),
];

export default function UserScorePerEventTable() {
  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/individualScore/individualScoring.js

```
import React from 'react';
import IndividualScorePerEventTable from './IndividualScorePerEventTable';
import { Grid } from '@mui/material';
import ScenarioTeamUserSelection from '../ScenarioTeamUserSelection';
import RankChartsContainer from './RankChartContainer';
import IndividualScoreContainer from './IndividualScoreContainer';

export default function IndividualScoring(props) {
  return (
    <div style={props.style}>
      <ScenarioTeamUserSelection style={{ marginBottom: '2vh' }} />
      <Grid container
        direction="row"
        justifyContent="center"
        alignItems="stretch"
        spacing={2}
      >
        <Grid item xs={12} md={10}>
          <IndividualScoreContainer />
        </Grid>
        <Grid item xs={12} md={2}>
          <RankChartsContainer />
        </Grid>
        <Grid item xs={12}>
          <IndividualScorePerEventTable />
        </Grid>
      </Grid>
    </div>
  );
}
```

src/individualScore/RankChart.js

```
import React from 'react';
import {
  ResponsiveContainer, PieChart, Pie, Label, Legend
} from 'recharts';

const data = [
  {
    rank: 1, score: 400
  },
  {
    rank: 2, score: 300
  },
  {
    rank: 3, score: 200
```

```
  }
];

export default function RankChart() {
  return (
    <ResponsiveContainer width={'100%'} height={300}>
      <PieChart
      >
        <Pie data={data}
          dataKey="score"
          nameKey="rank"
          legendType='circle'
          fill="#82ca9d"
        >
          <Legend iconType='circle'></Legend>
        </Pie>
      </PieChart>
    </ResponsiveContainer>
  );
}
```

src/individualScore/RankChartContainer.js

```
import React from 'react';
import Typography from '@material-ui/core/Typography';
import RankChart from './RankChart';

export default function RankChartsContainer(props) {
  return (
    <div style={props.style}>
      <Typography variant="overline" component="div" align="center">
        Overall rank
      </Typography>

      <RankChart />
    </div>
  );
}
```

src/score/Scoring.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import TeamScoreChart from './TeamScoreChart';
import TeamScorePerEventContainer from './TeamScorePerEventContainer';
import TopTeamScoresContainer from './TopTeamScoresContainer';
```

```
const useStyles = makeStyles((theme) => ({
  container: {
    display: 'grid',
    columnGap: '10px',
    gridColumnGap: '10px',
    rowGap: '10px',
    gridRowGap: '10px',
    justifyItems: 'stretch',
    alignItems: 'stretch',
    gridTemplateAreas:
     ` 'scoresAwarded scoresAwarded topScores'
       'scoresPerEvent scoresPerEvent scoresPerEvent'`,
  },
  scoresAwarded: {
    gridArea: 'scoresAwarded'
  },
  topScores: {
    gridArea: 'topScores'
  },
  scoresPerEvent: {
    gridArea: 'scoresPerEvent'
  },
}));

export default function Scoring() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <div className={classes.scoresAwarded}>
        <TeamScoreChart />
      </div>
      <div className={classes.topScores}>
        <TopTeamScoresContainer />
      </div>
      <div className={classes.scoresPerEvent}>
        <TeamScorePerEventContainer />
      </div>
    </div>
  );
}
```

src/score/TeamScoreChart.js

```
import React from 'react';
import {
   ResponsiveContainer, BarChart, Bar, XAxis, YAxis, Tooltip
```

```
} from 'recharts';

const data = [
  {
    name: 'Team A', score: 4000
  },
  {
    name: 'Team B', score: 3000
  },
  {
    name: 'Team C', score: 2000
  },
  {
    name: 'Team D', score: 2780
  },
  {
    name: 'Team E', score: 1890
  },
  {
    name: 'Team F', score: 2390
  },
  {
    name: 'Team G', score: 3490
  },
];

export default function TeamScoreChart() {
    return (
    <ResponsiveContainer width={'100%'} height={300}>
      <BarChart
        data={data}
      >
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Bar dataKey="score" name="Score" fill="#8884d8" />
      </BarChart>
       </ResponsiveContainer>
    );
}
```

src/score/TeamScorePerEventContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import TeamScorePerEventTable from './TeamScorePerEventTable';
```

```
const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function TeamScorePerEventContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Score Per Event
      </Typography>

      <TeamScorePerEventTable />
    </div>
  );
}
```

src/score/TeamScorePerEventTable.js

```
import React from 'react';
import BaseTable from '../BaseTable';

const columns = [
  { id: 'eventName', label: 'Event name', minWidth: 170 },
  { id: 'startTime', label: 'Start time', minWidth: 170 },
  {
    id: 'source',
    label: 'Source',
    minWidth: 170,
  },
  {
    id: 'destination',
    label: 'Destination',
    minWidth: 170,
  },
  {
    id: 'riskLevel',
    label: 'Risk level',
    minWidth: 170,
  },
  { id: 'team', label: 'Team', minWidth: 170 },
  { id: 'score', label: 'Score', minWidth: 170 },
```

```
];

function createData(eventName, startTime, source, destination, riskLevel,
team, score) {
  return { eventName, startTime, source, destination, riskLevel, team, score
};
}

const rows = [
  createData('Event 1', '2020-01-01T16:50:43', 'Source 1', 'Destination 1',
'Risk Level 1', 'Team A', '150'),
  createData('Event 2', '2020-01-01T16:51:43', 'Source 2', 'Destination 2',
'Risk Level 2', 'Team B', '786'),
  createData('Event 3', '2020-01-01T16:52:43', 'Source 3', 'Destination 3',
'Risk Level 3', 'Team C', '234'),
  createData('Event 4', '2020-01-01T16:53:43', 'Source 4', 'Destination 4',
'Risk Level 4', 'Team D', '79834'),
  createData('Event 5', '2020-01-01T16:54:43', 'Source 5', 'Destination 5',
'Risk Level 5', 'Team A', '1325'),
  createData('Event 6', '2020-01-01T16:55:43', 'Source 6', 'Destination 6',
'Risk Level 1', 'Team B', '685'),
  createData('Event 7', '2020-01-01T16:56:43', 'Source 7', 'Destination 7',
'Risk Level 2', 'Team C', '435'),
  createData('Event 8', '2020-01-01T16:57:43', 'Source 8', 'Destination 8',
'Risk Level 3', 'Team D', '24785'),
  createData('Event 9', '2020-01-01T16:58:43', 'Source 9', 'Destination 9',
'Risk Level 4', 'Team A', '234'),
  createData('Event 10', '2020-01-01T16:59:43', 'Source 10', 'Destination 10',
'Risk Level 5', 'Team B', '8354'),
  createData('Event 11', '2020-01-01T17:00:43', 'Source 11', 'Destination 11',
'Risk Level 1', 'Team C', '54696'),
  createData('Event 12', '2020-01-01T17:01:43', 'Source 12', 'Destination 12',
'Risk Level 2', 'Team D', '1342'),
  createData('Event 13', '2020-01-01T17:02:43', 'Source 13', 'Destination 13',
'Risk Level 3', 'Team A', '3150'),
  createData('Event 14', '2020-01-01T17:03:43', 'Source 14', 'Destination 14',
'Risk Level 4', 'Team B', '15320'),
];

export default function TeamScorePerEventTable() {
  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/score/TopTeamScoresContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import TopTeamScoresTable from './TopTeamScoresTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function TopTeamScoresContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Top team scores
      </Typography>

      <TopTeamScoresTable />
    </div>
  );
}
```

src/score/TopTeamScoresTable.js

```
import React from 'react';
import BaseTable from '../BaseTable';

const columns = [
  { id: 'score', label: 'Score', minWidth: 170 },
  { id: 'team', label: 'Team', minWidth: 100 },
];

function createData(score, team) {
  return { score, team };
}

const rows = [
  createData('1234', 'Team A'),
  createData('3243', 'Team B'),
  createData('634', 'Team C'),
  createData('87354', 'Team D'),
  createData('435', 'Team E'),
  createData('6778', 'Team F'),
```

```
  createData('2421', 'Team G'),
];

export default function TopTeamScoresTable() {

  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/score/TeamScoring.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import UserScoreChart from './UserScoreChart';
import UserScorePerEventContainer from './UserScorePerEventContainer';
import TopUserScoresContainer from './TopUserScoresContainer';

const useStyles = makeStyles((theme) => ({
  container: {
    display: 'grid',
    columnGap: '10px',
    gridColumnGap: '10px',
    rowGap: '10px',
    gridRowGap: '10px',
    justifyItems: 'stretch',
    alignItems: 'stretch',
    gridTemplateAreas:
     `  'scoresAwarded scoresAwarded topScores'
        'scoresPerEvent scoresPerEvent scoresPerEvent'`,
  },
  scoresAwarded: {
    gridArea: 'scoresAwarded'
  },
  topScores: {
    gridArea: 'topScores'
  },
  scoresPerEvent: {
    gridArea: 'scoresPerEvent'
  },
}));

export default function TeamScoring() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
```

```
      <div className={classes.scoresAwarded}>
        <UserScoreChart />
      </div>
      <div className={classes.topScores}>
        <TopUserScoresContainer />
      </div>
      <div className={classes.scoresPerEvent}>
        <UserScorePerEventContainer />
      </div>
    </div>
  );
}
```

src/score/TopUserScoresContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import TopUserScoresTable from './TopUserScoresTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function TopUserScoresContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Top user scores
      </Typography>

      <TopUserScoresTable />
    </div>
  );
}
```

src/score/TopUserScoresTable.js

```
import React from 'react';
import BaseTable from '../BaseTable';
```

```
const columns = [
  { id: 'score', label: 'Score', minWidth: 170 },
  { id: 'user', label: 'User', minWidth: 100 },
];

function createData(score, user) {
  return { score, user };
}

const rows = [
  createData('1234', 'User A'),
  createData('3243', 'User B'),
  createData('634', 'User C'),
  createData('87354', 'User D'),
  createData('435', 'User E'),
  createData('6778', 'User F'),
  createData('2421', 'User G'),
];

export default function TopUserScoresTable() {

  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/score/UserScoreChart.js

```
import React from 'react';
import {
   ResponsiveContainer, BarChart, Bar, XAxis, YAxis, Tooltip
} from 'recharts';

const data = [
  {
    user: 'User A', score: 4000
  },
  {
    user: 'User B', score: 3000
  },
  {
    user: 'User C', score: 2000
  },
  {
    user: 'User D', score: 2780
  },
```

```
  {
    user: 'User E', score: 1890
  },
  {
    user: 'User F', score: 2390
  },
  {
    user: 'User G', score: 3490
  },
];

export default function UserScoreChart() {
    return (
        <ResponsiveContainer width={'100%'} height={300}>
      <BarChart
        data={data}
      >
        <XAxis dataKey="user" />
        <YAxis />
        <Tooltip />
        <Bar dataKey="score" name="Score" fill="#8884d8" />
      </BarChart>
       </ResponsiveContainer>
    );
}
```

src/score/UserScorePerEventContainer.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Typography from '@material-ui/core/Typography';
import UserScorePerEventTable from './UserScorePerEventTable';

const useStyles = makeStyles((theme) => ({
  container: {
    height: '100%',
    width: '100%'
  }
}));

export default function UserScorePerEventContainer() {
  const classes = useStyles();

  return (
    <div className={classes.container}>
      <Typography variant="overline" component="div" align="center">
        Score Per Event
```

```
      </Typography>

      <UserScorePerEventTable />
    </div>
  );
}
```

src/score/UserScorePerEventTable.js

```
import React from 'react';
import BaseTable from '../BaseTable';

const columns = [
  { id: 'eventName', label: 'Event name', minWidth: 170 },
  { id: 'startTime', label: 'Start time', minWidth: 170 },
  {
    id: 'source',
    label: 'Source',
    minWidth: 170,
  },
  {
    id: 'destination',
    label: 'Destination',
    minWidth: 170,
  },
  {
    id: 'riskLevel',
    label: 'Risk level',
    minWidth: 170,
  },
  { id: 'user', label: 'User', minWidth: 170 },
  { id: 'score', label: 'Score', minWidth: 170 },
];

function createData(eventName, startTime, source, destination, riskLevel,
user, score) {
  return { eventName, startTime, source, destination, riskLevel, user, score
};
}

const rows = [
  createData('Event 1', '2020-01-01T16:50:43', 'Source 1', 'Destination 1',
'Risk Level 1', 'User A', '150'),
  createData('Event 2', '2020-01-01T16:51:43', 'Source 2', 'Destination 2',
'Risk Level 2', 'User B', '786'),
  createData('Event 3', '2020-01-01T16:52:43', 'Source 3', 'Destination 3',
'Risk Level 3', 'User C', '234'),
```

```
  createData('Event 4', '2020-01-01T16:53:43', 'Source 4', 'Destination 4',
'Risk Level 4', 'User D', '79834'),
  createData('Event 5', '2020-01-01T16:54:43', 'Source 5', 'Destination 5',
'Risk Level 5', 'User A', '1325'),
  createData('Event 6', '2020-01-01T16:55:43', 'Source 6', 'Destination 6',
'Risk Level 1', 'User B', '685'),
  createData('Event 7', '2020-01-01T16:56:43', 'Source 7', 'Destination 7',
'Risk Level 2', 'User C', '435'),
  createData('Event 8', '2020-01-01T16:57:43', 'Source 8', 'Destination 8',
'Risk Level 3', 'User D', '24785'),
  createData('Event 9', '2020-01-01T16:58:43', 'Source 9', 'Destination 9',
'Risk Level 4', 'User A', '234'),
  createData('Event 10', '2020-01-01T16:59:43', 'Source 10', 'Destination 10',
'Risk Level 5', 'User B', '8354'),
  createData('Event 11', '2020-01-01T17:00:43', 'Source 11', 'Destination 11',
'Risk Level 1', 'User C', '54696'),
  createData('Event 12', '2020-01-01T17:01:43', 'Source 12', 'Destination 12',
'Risk Level 2', 'User D', '1342'),
  createData('Event 13', '2020-01-01T17:02:43', 'Source 13', 'Destination 13',
'Risk Level 3', 'User A', '3150'),
  createData('Event 14', '2020-01-01T17:03:43', 'Source 14', 'Destination 14',
'Risk Level 4', 'User B', '15320'),
];

export default function UserScorePerEventTable() {
  return (
    <BaseTable columns={columns} rows={rows} />
  );
}
```

src/script/ScriptManagement.js

```
import React, { useState, useEffect } from 'react';
import ScriptsTable from './ScriptsTable';
import PageTitle from '../PageTitle';

export default function ScriptManagement() {
  return (
    <div>
      <PageTitle titleText='Scripts' />
      <ScriptsTable />
    </div>
  );
}
```

src/script/ScriptRunForm.js

```
import * as React from 'react';
import TextField from '@mui/material/TextField';

export default function ScriptRunForm(props) {
  return (
    <form>
      {props.variables && props.variables.map(ss_var => (
          <TextField
            autoFocus
            margin="dense"
            key={ss_var.orderNo}
            id={ss_var.variable.id}
            label={ss_var.variable.name}
            fullWidth
            variant="outlined"
            required={ss_var.required}
            onChange={e => props.onValueChange(e.target.id, e.target.value)}
          />
      ))}
    </form>
  );
}
```

src/script/ScriptRunFormDialog.js

```
import * as React from 'react';
import Button from '@mui/material/Button';
import TextField from '@mui/material/TextField';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle from '@mui/material/DialogTitle';
import ScriptRunForm from './ScriptRunForm';

export default function ScriptRunFormDialog(props) {
  const [variableValues, setVariableValues] = React.useState({});

  const updateVariableValues = (values) => {
    console.log("vars values bef : " + JSON.stringify(variableValues));
    setVariableValues(values);
    console.log("vars values after : " + JSON.stringify(variableValues));
  };

  const updateValue = (key, value) => {
    console.log("key : " + key + ", value : " + value);
```

```
    console.log("vars bef : " + JSON.stringify(variableValues));
    variableValues[key] = value;
    console.log("vars after : " + JSON.stringify(variableValues));
  };

  const run = () => {
    console.log("run scriptId : " + props.script.id + ", variableValues : " +
JSON.stringify(variableValues));
    props.onRun(props.script.id, variableValues);
  }

  return (
    <div>
      <Dialog open={props.open} onClose={props.onClose}>
        <DialogTitle>{props.script && props.script.name}</DialogTitle>
        <DialogContent>
          <DialogContentText>
            {props.script && props.script.description}
          </DialogContentText>
          <ScriptRunForm variables={props.script && props.script.variables}
variableValues={variableValues}
            onValueChange={updateValue} />
        </DialogContent>
        <DialogActions>
          <Button onClick={props.onClose}>Cancel</Button>
          <Button onClick={run}>Run</Button>
        </DialogActions>
      </Dialog>
    </div>
  );
}
```

src/script/ScriptsTable.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { DataGrid } from '@mui/x-data-grid';
import axios from 'axios';
import { PlayArrow } from '@mui/icons-material';
import IconButton from '@mui/material/IconButton';
import { getScripts, sendRunScriptRequest } from '../ServerCaller';
import ScriptRunFormDialog from './ScriptRunFormDialog';
import CtrlSnackBar from '../CtrlSnackBar';

function createData(script, idValue, nameValue, descriptionValue,
locationValue,
  executorValue, categoryValue, variableValues) {
```

```
  return {
    id: idValue, name: nameValue,
    description: descriptionValue, location: locationValue,
    executor: executorValue, category: categoryValue,
    variables: variableValues, run: script
  };
}

function createRows(data) {
  let rowList = [];
  data.forEach(r => {
    rowList.push(createData(r, r.id, r.name, r.description, r.location,
      r.executor, r.category, r.variables));
  });
  console.log('rowList : ' + JSON.stringify(rowList));
  return rowList;
}

function runScript(script, updateForm, handleRunError) {
  console.log("script to run : " + JSON.stringify(script));
  let vars = script.variables;
  console.log("variables : " + JSON.stringify(vars));
  if (vars && vars.length) {
    console.log("variables : " + JSON.stringify(vars));
    updateForm(true, script);
  } else {
    updateForm(false, '');
    sendRunScriptRequest(script.id, null, null, handleRunError);
  }
}

function createRunButton(params, updateForm, handleRunError) {
  return (
    <IconButton color="primary" onClick={(event) => { runScript(params.value,
updateForm, handleRunError); }}>
      <PlayArrow />
    </IconButton>
  );
}

function createColumns(updateForm, handleRunError) {
  let cols = [
    { field: 'name', headerName: 'Name', flex: 1, sortable: true },
    { field: 'description', headerName: 'Description', flex: 1, sortable:
false },
    { field: 'location', headerName: 'Location', flex: 1, sortable: true },
    {
      field: 'executor', headerName: 'Executor', flex: 1, sortable: true,
```

```
      valueGetter: ({ value }) => { return value.name }
    },
    {
      field: 'category', headerName: 'Category', flex: 1, sortable: true,
      valueGetter: ({ value }) => { return value.name }
    },
    {
      field: 'run', headerName: 'Run', flex: 1, sortable: false,
      type: 'actions',
      // getActions: (params) => [
      //   <GridActionsCellItem icon={ <PlayArrow /> } onClick={
runScirpt(params) } label="Run" />,
      // ]
      renderCell: (params) => createRunButton(params, updateForm,
handleRunError)
    },
  ];

  return cols;
}

export default function ScriptsTable() {
  const [snackBarOpen, setSnackBarOpen] = useState({
    open: false,
    msg: undefined
  });
  const [formOpen, setFormOpen] = useState({
    open: false,
    selectedScript: null
  });
  const updateForm = (openV, script) => {
    console.log(`in updateForm openV : ${openV}
    script : ${JSON.stringify(script)}`);
    setFormOpen({ open: openV, selectedScript: script })
  }
  const closeForm = () => { console.log("in closeForm"); setFormOpen({ open:
false }) }

  const [tableState, setTableState] = useState({
    loading: false,
    rows: [],
  });

  const handleRunError = (serverErrorResponse) => {
    let message = serverErrorResponse.response.data;
    // TODO setSnackBarOpen({ open: true, msg: message });
  }
```

```
  const columns = createColumns(updateForm, handleRunError);

  useEffect(() => {
    let mounted = true;
    setTableState({ loading: true });

    function updateTableState(scripts) {
      if (mounted) {
        setTableState({ loading: false, rows: createRows(scripts) });
      }
    }

    getScripts(updateTableState, null);

    return function cleanup() {
      mounted = false;
    }
  }, [setTableState]);

  if (tableState.loading) {
    return 'Loading...';
  }
  return (
    <div style={{ width: '100%' }}>
      <CtrlSnackBar />
      <ScriptRunFormDialog open={formOpen.open} onClose={closeForm}
        onRun={sendRunScriptRequest} script={formOpen.selectedScript} />
      <DataGrid rows={tableState.rows} columns={columns} autoPageSize={true}
        autoHeight={true} showColumnRightBorder={true}
showCellRightBorder={true}
        initialState={{ pinnedColumns: { right: ['run'] } }} />
    </div>
  );
}
```

Βιβλιογραφία

[1] ENISA, «ENISA Threat Landscape 2022,» ENISA, 2022.

[2] «Locked Shields,» [Ηλεκτρονικό]. Available: https://ccdcoe.org/exercises/locked-shields/.

[3] «Crossed Swords,» [Ηλεκτρονικό]. Available: https://ccdcoe.org/exercises/crossed-swords/.

[4] «European Cyber Security Challenge,» [Ηλεκτρονικό]. Available: https://ecsc.eu/.

[5] «SANS Cyber Range,» [Ηλεκτρονικό]. Available: https://www.sans.org/cyber-ranges/.

[6] «SANS Holiday Hack Challenge,» [Ηλεκτρονικό]. Available: https://www.sans.org/mlp/holiday-hack-challenge/.

[7] P. Čeleda, J. Čegan, J. Vykopal και D. Tovarňák, «KYPO – A Platform for Cyber Defence Exercises,» 2015.

[8] «Cyberbit,» [Ηλεκτρονικό]. Available: https://www.cyberbit.com/.

[9] G. Potamos, A. Peratikou και S. Stavrou, «Towards a Maritime Cyber Range training environment,» σε *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021.

[10] O. Jacq, P. G. Salazar, K. Parasuraman, J. Kuusijärvi, A. Gkaniatsou, E. Latsa και A. Amditis, «The Cyber-MAR Project: First Results and Perspectives on the Use of Hybrid Cyber Ranges for Port Cyber Risk Assessment,» σε *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021.

[11] A. Alvarez, G. Yuste, J. Villalobos, J. Martínez, M. Chronopoulos, M. Ghering, I. Karapistoli, G. Alexopoulos, C. Lombardo, J. F. Pajo, C. Marcenaro, P. Polvanesi, C. Costa, M. Athanatos, N. Petroulakis, A. Mokkas, I. Tsampoulatidis, G. Lamanna, M. Giribaldi, G. Amponis, M. Zevgara, S. Ouzounidis, M. Bärmann, M. Skarregaard, S. Egenfeldt-Nielsen, A. Miaoudakis, D. Ntegiannis, M. Smyrlis, B. Vidalenc, F. Rebecchi, G. Chollon, A. Pastor, D. López, P. Gouvas, P. Velonias, I. Aliferis, I. Kontakos, A. Mozo, D. Rivera, J. I. Moreno, L. Marcos, S. Vakaruk, A. Karamchandani, A. Angelogianni, I. Politis, C. Xenakis, V. Fragkos και K. Papachristopoulou, *SPIDER 5G Cyber Range,* 2022.

[12] «MITRE ATT&CK,» [Ηλεκτρονικό]. Available: https://attack.mitre.org/.

[13] «CVE,» [Ηλεκτρονικό]. Available: https://cve.mitre.org/.

[14] «theHarvester,» [Ηλεκτρονικό]. Available: https://github.com/laramies/theHarvester.

[15] «SpiderFoot,» [Ηλεκτρονικό]. Available: https://www.spiderfoot.net/documentation/.

[16] «Atomic Red Team,» [Ηλεκτρονικό]. Available: https://github.com/redcanaryco/atomic-red-team.

[17] «Invoke-AtomicRedTeam,» [Ηλεκτρονικό]. Available: https://github.com/redcanaryco/invoke-atomicredteam.

[18] «Nikto,» [Ηλεκτρονικό]. Available: https://cirt.net/Nikto2.

[19] «sqlmap,» [Ηλεκτρονικό]. Available: https://sqlmap.org/.

[20] «Exploit Database,» [Ηλεκτρονικό]. Available: https://www.exploit-db.com/.

[21] «OWASP ZAP,» [Ηλεκτρονικό]. Available: https://www.zaproxy.org/.

[22] «MHDDOS,» [Ηλεκτρονικό]. Available: https://github.com/MatrixTM/MHDDoS.

[23] J. Davis και S. Magrath, «A Survey of Cyber Ranges and Testbeds,» 2013.

[24] M. M. Yamin, B. Katt και V. Gkioulos, «Cyber ranges and security testbeds: Scenarios, functions, tools and architecture,» *Computers & Security,* τόμ. 88, 2020.

[25] J. Vykopal, R. Ošlejšek, P. Celeda, M. Vizváry και D. Tovarnák, «KYPO Cyber Range: Design and Use Cases,» σε *Proceedings of the 12th International Conference on Software Technologies - Volume 1: ICSOFT*, 2017.

[26] J. Mirkovic, T. V. Benzel, T. Faber, R. Braden, J. T. Wroclawski και S. Schwab, «The DETER Project: Advancing the Science of Cyber Security Experimentation and Test,» σε *In Proceedings of the 2010 IEEE International Conference on Technologies for Homeland Security (HST '10)*, Waltham, Massachusetts, 2010.

[27] «Merit Alphaville,» [Ηλεκτρονικό]. Available: https://www.merit.edu/security/training/alphaville/.

[28] E. C. S. O. (ECSO), «Understanding Cyber Ranges: From Hype to Reality,» Brussels, 2020.

[29] A. M. Ugur, K. Bicakci, H. M. Dilek, M. A. Ozbayoglu και İ. E. Tatlı, «Automated Generation of Attack Graphs Using NVD,» σε *In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy (CODASPY '18)*, Tempe, AZ, USA, 2018.

[30] T. Sommestad, *Experimentation on operational cyber security in CRATE,* 2015.

[31] H. Holm και T. Sommestad, «SVED: Scanning, Vulnerabilities, Exploits and Detection,» σε *MILCOM 2016 - 2016 IEEE Military Communications Conference*, 2016.

[32] A. Applebaum, D. Miller, B. Strom, C. Korban και R. Wolf, «Intelligent, Automated Red Team Emulation,» σε *Proceedings of the 32nd Annual Conference on Computer Security Applications*, Los Angeles, California, USA, 2016.

[33] S. Y. Enoch, Z. Huang, C. Y. Moon, D. Lee, M. K. Ahn και D. S. Kim, «HARMer: Cyber-attacks Automation and Evaluation,» *IEEE Access,* τόμ. 8, pp. 129397-129414, 07 2020.

[34] J. Hong και D. S. Kim, «HARMs: Hierarchical attack representation models for network security analysis,» σε *Proceedings of the 10th Australian Information Security Management Conference, AISM 2012*, Joondalup, WA Australia, 2012.

[35] J. Plot, A. Shaffer και G. Singh, «CARTT: Cyber Automated Red Team Tool,» σε *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[36] F. Rebecchi, A. Pastor, A. Mozo, C. Lombardo, R. Bruschi, I. Aliferis, R. Doriguzzi-Corin, P. Gouvas, A. A. Romero, A. Angelogianni, I. Politis και C. Xenakis, «A Digital Twin for the 5G Era: the SPIDER Cyber Range,» σε *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2022.

[37] E. C. Chaskos, *Cyber-security training: A comparative analysis of cyber-ranges and emerging trends,* 2019.

[38] [Ηλεκτρονικό]. Available: https://www.eda.europa.eu/info-hub/press-centre/latest-news/2019/11/07/eda-cyber-ranges-federation-project-showcased-at-demo-exercise-in-finland.

[39] «ECHO,» [Ηλεκτρονικό]. Available: https://echonetwork.eu/.

[40] N. Oikonomou, N. Mengidis, M. Spanopoulos, A. Voulgaridis, M. Merialdo, I. Raisr, K. H. Guardtime, P. de La Vallee, T. Tsikrika, S. Vrochidis και K. Votis, «ECHO Federated Cyber Range: Towards Next-Generation Scalable Cyber Ranges,» σε *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021.

[41] W. W. Royce, «Managing the development of large software systems,» σε *Technical Papers of Western Electronic Show and Convention*, 1970.

[42] K. Schwaber, «SCRUM Development Process,» σε *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*, 1995.

[43] «Azure Boards,» [Ηλεκτρονικό]. Available: https://azure.microsoft.com/en-us/services/devops/boards/.

[44] «Jira,» [Ηλεκτρονικό]. Available: https://www.atlassian.com/software/jira.

[45] «Trello,» [Ηλεκτρονικό]. Available: https://trello.com/en.

[46] «ManageIQ,» [Ηλεκτρονικό]. Available: https://www.manageiq.org/docs/get-started/.

[47] C. Virág, J. Čegan, T. Lieskovan και M. Merialdo, «The Current State of The Art and Future of European Cyber Range Ecosystem,» σε *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021.