ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΣΦΑΛΕΙΑ ΣΥΣΤΗΜΑΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση IDS με χρήση του εργαλείου Zeek

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ

Μπούρας Αντώνιος [ΑΕΜ: 58287]
antobour1@ee.duth.gr
Σιδέρης Πέτρος [ΑΕΜ: 58341]
petrside@ee.duth.gr
Ταχτατζής Αναστάσιος [ΑΕΜ: 58439]
anastach2@ee.duth.gr

Υπεύθυνος καθηγητής:
Αναγνωστόπουλος Μάριος

13 Ιανουαρίου, 2025

Περιεχόμενα

1 Εισαγωγή **2**
1.1 Περιγραφή του προβλήματος 2
1.2 Zeek 2

2 Υλοποίηση **3**
2.1 Αρχικοποίηση 3
2.2 Επιθέσεις 4
2.3 Κανόνες 5
2.3.1 Ανίχνευση Path Traversal 5
2.3.2 Ανίχνευση Code Injection 5
2.3.3 Έλεγχος μεγάλου Payload (buffer overflow) 5

3 Στιγμιότυπα Οθόνης **7**

4 Οπτικοποίηση **9**

1 Εισαγωγή

Το σύνολο της υλοποίησης βρίσκεται στον παρακάτω σύνδεσμο:

https://github.com/petersid2022/ids_project

1.1 Περιγραφή του προβλήματος

Η συγκεκριμένη εξαμηνιαία εργασία επικεντρώνεται στην ανάπτυξη Συστήματος Ανίχνευσης Επίθεσης (Intrusion Detection System - IDS) με χρήση του ανοιχτού κώδικα λογισμικού ανάλυσης δικτυακής κίνησης, **Zeek**.

Η εφαρμογή του συστήματος γίνεται σε τοπικά αποθηκευμένη ιστοσελίδα τύπου ιστολογίου (blog), η οποία αποτελείται από 3 διαδραστικά μέρη (τίτλος, κείμενο, δημοσίευση) όπου ο χρήστης μπορεί να επέμβει μόνο στον τίτλο και το κείμενο. Οι πληροφορίες αποθηκεύονται σε βάση δεδομένων και εμφανίζονται στο κάτω μέρος της ιστοσελίδας.

Επιγραμματικά, για τον έλεγχο λειτουργικότητας της υλοποίησης, πραγματοποιούνται 5 χειροκίνητες επιθέσεις (**XSS, SQL Injection, NGINX Path Traversal, DDoS**) και αξιοποιούνται 3 αυτόματα εργα-λεία (**NMap, hping3, oha**) αντίστοιχα για τη διενέργεια επιθέσεων. Σκοπός της εργασίας είναι αξιοποίηση του **Zeek** για τον έγκαιρο εντοπισμό τους με χρήση κανόνων εντός αυτού.

1.2 Zeek

Το **Zeek** είναι υλοποίηση ανοιχτού κώδικα που προσφέρει αναλυτικές πληροφορίες για την δραστηριότητα που μπορεί να συμβαίνει σε ένα δίκτυο. Η λειτουργία του διαφέρει από τους παραδοσιακούς τρόπους πρόληψης επιθέσεων (**Firewalls** και **Intrusion Prevention Systems**) καθώς λειτουργεί ήπια (passive) καταγράφοντας σε πραγματικό χρόνο τη δικτυακή κίνηση.

Το **Zeek** θεμελιακά είναι πλατφόρμα σεναρίων (**scripting platform**), συνεπώς καθίσταται επί της αρχής εξατομικευμένη λύση που μπορεί να εφαρμοστεί σε πολλά συστήματα. Οι δυνατότητες του στην καταγραφή γεγονότων στο δίκτυο κυμαίνονται από λεπτομερή καταγραφή και εξαγωγή τους σε αρχείο έως εφαρμογή χειριστών (**handlers**) για συγκεκριμένα γεγονότα (**events**) που κατόπιν αυτών μπορεί να τρέξει υποδειγμένο κώδικα.

2 Υλοποίηση

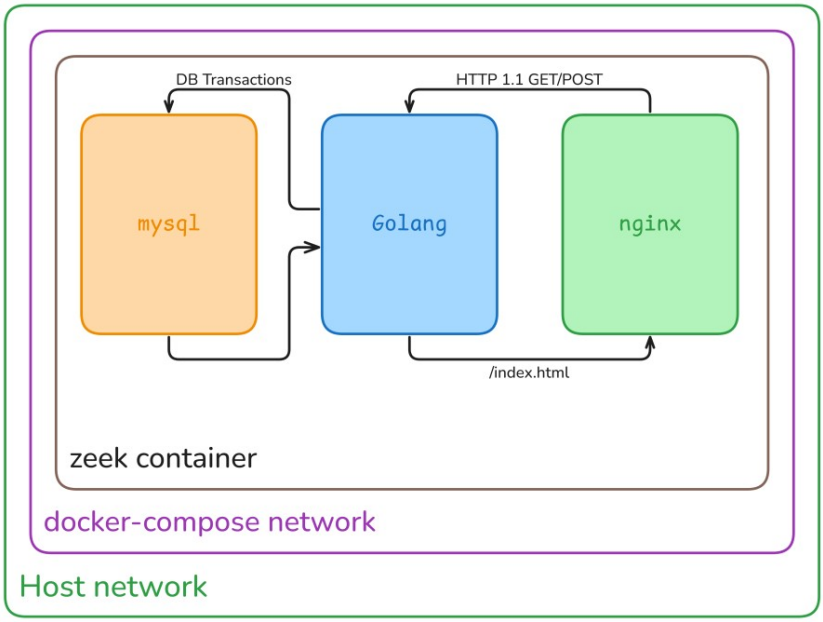
2.1 Αρχικοποίηση

Η υλοποίηση του συστήματος ανίχνευσης εισβολών (IDS) βασίζεται στη δημιουργία μιας ολοκληρωμένης υποδομής, η οποία περιλαμβάνει client-server αρχιτεκτονική, τη χρήση του εργαλείου **Zeek** για την ανίχνευση εισβολών, καθώς και τη διαχείριση δεδομένων και τη δοκιμή επιθέσεων μέσω αυτοματοποιημένων και χειροκίνητων σεναρίων. Όλα τα μέρη της υποδομής αναπτύχθηκαν με γνώμονα τη χρήση **containerization**, ώστε να εξασφαλιστεί η ευκολία εγκατάστασης, διαχείρισης και παραμετροποίησης.

Η γενική τοπολογία περιλαμβάνει τρεις βασικούς πυλώνες: τον client, τον server και το **Zeek-based** IDS. Ο client αποτελεί το frontend της υποδομής μας και είναι υπεύθυνος για την αλληλεπίδραση με τον χρήστη. Περιλαμβάνει στατικά αρχεία **HTML**, **CSS** και **JavaScript**, τα οποία φιλοξενούνται μέσω ενός container που λειτουργεί με **Nginx**. Αυτή η πλευρά του συστήματος επιτρέπει στον χρήστη να στέλνει αιτήματα και να προσομοιώνει διαφορετικά σενάρια επιθέσεων, όπως SQL Injection ή **Cross-Site Scripting** (**XSS**).

Ο server υλοποιήθηκε με τη χρήση της γλώσσας **Go**, η οποία προσφέρει υψηλή απόδοση και ασφάλεια στη διαχείριση αιτημάτων. Λειτουργεί ως ο μεσάζων ανάμεσα στον client και τα υπόλοιπα υποσυστήματα, επεξεργάζεται τα εισερχόμενα δεδομένα και αποθηκεύει τις πληροφορίες σε μια βάση δεδομένων **MySQL**. Το αρχείο **setup.sql** περιλαμβάνει τις αρχικές ρυθμίσεις της βάσης, ενώ ο κώδικας του server (main.go) έχει σχεδιαστεί για να δέχεται τόσο έγκυρα όσο και κακόβουλα αιτήματα, διευκολύνοντας τη δοκιμή των κανόνων του **Zeek**.

Στον πυρήνα του συστήματος βρίσκεται το **Zeek**, το οποίο έχει παραμετροποιηθεί να λειτουργεί σε περιβάλλον **Docker**. Το **Zeek** είναι υπεύθυνο για την παρακολούθηση της δικτυακής κίνησης που διέρχεται από τον server. Οι κανόνες ανίχνευσης, που είναι ορισμένοι στο αρχείο **alerts.zeek**, σχεδιάστηκαν για να εντοπίζουν συγκεκριμένα μοτίβα επιθέσεων, όπως **DDoS**, **SQL Injection** και **Path Traversal**. Το εργαλείο είναι επίσης ρυθμισμένο να εξάγει δεδομένα και alerts, τα οποία καταγράφονται στη βάση δεδομένων για περαιτέρω ανάλυση.



Εικόνα 2.1.1 Γενική επισκόπηση της προτεινόμενης αρχιτεκτονικής

Όλα τα παραπάνω υποσυστήματα είναι συνδεδεμένα μέσω ενός **docker-compose** αρχείου, το οποίο ρυθμίζει την επικοινωνία ανάμεσα στα containers. Το **Zeek** λειτουργεί ως το κεντρικό IDS, ενώ ο server και ο client είναι υπεύθυνοι για τη διαχείριση και την εκτέλεση αιτημάτων. Η παρακολούθηση των δικτυακών αιτημάτων και η ανταπόκριση του συστήματος γίνεται σε πραγματικό χρόνο, επιτρέποντας την ανάλυση της απόδοσης των κανόνων ανίχνευσης και την αξιολόγηση της λειτουργικότητας

του συστήματος.

Με αυτήν την υποδομή, καταφέραμε να δημιουργήσουμε ένα ολοκληρωμένο περιβάλλον, το οποίο είναι επεκτάσιμο και ευέλικτο, τόσο για τη δοκιμή νέων κανόνων ανίχνευσης όσο και για την προσομοίωση διαφορετικών σεναρίων εισβολής.

2.2 Επιθέσεις

Στο πλαίσιο της υλοποίησης του Intrusion Detection System (IDS) με το εργαλείο **Zeek**, δοκιμάστηκαν διάφορες τεχνικές επιθέσεων για την αξιολόγηση της αποτελεσματικότητάς του. Οι επιθέσεις χωρίζονται σε δύο κατηγορίες: χειροκίνητες επιθέσεις και αυτοματοποιημένες επιθέσεις, οι οποίες αποσκοπούν στον εντοπισμό και την ανίχνευση κακόβουλων ενεργειών από το IDS.

Αρχικά, πραγματοποιήθηκαν χειροκίνητες επιθέσεις με στόχο την εκμετάλλευση αδυναμιών σε εφαρμογές και συστήματα. Μια από τις πιο συνηθισμένες επιθέσεις ήταν το **Cross-Site Scripting** (**XSS**), το οποίο περιλαμβάνει την εισαγωγή κακόβουλου **JavaScript** κώδικα σε ιστοσελίδες, με σκοπό την εκτέλεσή του στον browser του χρήστη. Στην προκειμένη περίπτωση, χρησιμοποιήθηκε ο παρακάτω κώδικας για να προκαλέσει την εμφάνιση του περιεχομένου του cookie του χρήστη:

```
<img src="" onerror="alert(document.cookie);" />
```

Ένα ακόμα παράδειγμα επίθεσης ήταν η **SQL Injection**, η οποία εκμεταλλεύεται ευπάθειες στις **SQL** ερωτήσεις μιας εφαρμογής για την εκτέλεση κακόβουλων εντολών στη βάση δεδομένων. Για παράδειγμα, χρησιμοποιήθηκε η εξής εντολή:

```
INSERT INTO Posts (title, content) VALUES ('test1', 'test2'); DROP TABLE Posts; --
```

Η επίθεση αυτή αποσκοπούσε στην εκτέλεση ενός κακόβουλου **SQL statement** για τη διαγραφή δεδομένων από τη βάση. Παράλληλα, πραγματοποιήθηκε και επίθεση **Path Traversal**, η οποία στόχευε στην πρόσβαση σε ευαίσθητα αρχεία του συστήματος μέσω κακόβουλων αιτημάτων, όπως φαίνεται στο παρακάτω παράδειγμα:

```
curl http://localhost/api/etc/passwd
```

Αυτού του είδους οι επιθέσεις χρησιμοποιούνται για την υποκλοπή κρίσιμων συστημικών αρχείων.

Επιπλέον, εξετάστηκε η επίθεση **Buffer Overflow**, στην οποία αποστέλλονται δεδομένα μεγαλύτερα του αναμενόμενου μεγέθους για να προκαλέσουν υπερχείλιση μνήμης. Η συγκεκριμένη επίθεση εκτελέστηκε με την αποστολή μεγάλου αριθμού χαρακτήρων μέσω **HTTP αιτήματος**:

```
curl -H "Content-Type: application/json" -d '{"title":"hello world", "content":"'$
(python -c 'print("A"*5050)' | sed 's/"/\\"/g')'"}' -X POST http://localhost:1234
```

Τέλος, πραγματοποιήθηκε και μια επίθεση **Distributed Denial of Service** (**DDoS**), η οποία είχε ως στόχο τη φόρτωση του συστήματος με μεγάλο όγκο αιτημάτων, μέσω του **script ./ddos.sh**.

Από την άλλη πλευρά, χρησιμοποιήθηκαν και αυτοματοποιημένα εργαλεία για την εκτέλεση επιθέσεων μεγάλης κλίμακας. Ένα από τα πρώτα εργαλεία ήταν το **hping3**, το οποίο χρησιμοποιεί **TCP SYN** πακέτα για την εκτέλεση **DDoS** επιθέσεων. Ένα παράδειγμα της εντολής που χρησιμοποιήθηκε είναι η εξής:

```
sudo hping3 -i u40 -S -p 1234 -c 1000000 172.21.0.3
```

Ένα άλλο εργαλείο που χρησιμοποιήθηκε ήταν το **Apache Bench** (**ab**), το οποίο χρησιμοποιείται για την αποστολή μεγάλου αριθμού **HTTP** αιτημάτων σε έναν server, προκειμένου να εξεταστεί η αντοχή του συστήματος υπό φορτίο. Η εντολή για την εκτέλεση της επίθεσης είναι:

```
ab -n 100000 -c 100 http://localhost:1234/
```

Ένα ακόμα εργαλείο ήταν το **Oha**, το οποίο χρησιμοποιεί παρόμοια λογική με το **Apache Bench**, αλλά είναι σχεδιασμένο για εξαιρετικά υψηλές επιδόσεις. Η εντολή που χρησιμοποιήθηκε για την εκτέλεση του εργαλείου είναι:

```
oha -z 2m -c 1000 http://localhost:1234
```

Τέλος, χρησιμοποιήθηκε το εργαλείο **Nmap** για την ανίχνευση ανοιχτών θυρών και την εκτίμηση της ασφάλειας του συστήματος μέσω σάρωσης υπηρεσιών και συσκευών.

Οι παραπάνω επιθέσεις καλύπτουν ένα ευρύ φάσμα κακόβουλων ενεργειών, οι οποίες αναμένεται να ανιχνευθούν και να καταγραφούν από το IDS. Συμπερασματικά, είναι φανερό ότι το **Zeek** ήταν

αποτελεσματικό στην ανίχνευση των περισσότερων από αυτές τις επιθέσεις. Ωστόσο, παρατηρήθηκαν περιοχές που απαιτούν περαιτέρω βελτιώσεις, όπως η παραμετροποίηση των κανόνων για την καλύτερη αναγνώριση επιθέσεων με υψηλή συχνότητα, όπως οι **DDoS** επιθέσεις.

2.3 Κανόνες

2.3.1 Ανίχνευση Path Traversal

```
if (path_traversal_keywords in original_URI) {
    print fmt("[ALERT] Potential Path traversal detected.
    Host: %s:%d, URI: %s", c$id$orig_h, c$id$orig_p, original_URI);
}
```

Ο πρώτος κανόνας εξετάζει αν το **URI** (Uniform Resource Identifier) που περιλαμβάνεται στο αίτημα περιέχει λέξεις-κλειδιά που σχετίζονται με τεχνικές path traversal, όπως `../`, `..
, /etc/passwd` ή άλλες παρόμοιες ακολουθίες. Αν ανιχνευτεί τέτοια λέξη-κλειδί, εμφανίζεται ειδοποίηση που περιλαμβάνει τη διεύθυνση IP του αιτούντος, την θύρα προέλευσης και το **URI** που θεωρείται ύποπτο.

```
[ALERT] Potential Path traversal detected.
```

περιλαμβάνοντας τη διεύθυνση IP (`c$id$orig_h`), την θύρα (`c$id$orig_p`), και το **URI**.

Ο σκοπός αυτού του κανόνα είναι να εντοπίζει προσπάθειες παράκαμψης του προβλεπόμενου εύρους πρόσβασης σε αρχεία ή καταλόγους του διακομιστή. Αυτές οι επιθέσεις χρησιμοποιούνται συνήθως για την απόκτηση πρόσβασης σε ευαίσθητα δεδομένα ή αρχεία διαμόρφωσης που δεν θα έπρεπε να είναι προσβάσιμα από εξωτερικούς χρήστες.

2.3.2 Ανίχνευση Code Injection

```
if (code_injection_keywords in full_body) {
    print fmt("[ALERT] Potential Code Injection detected.
    Host: %s:%d, Body: %s", c$id$orig_h, c$id$orig_p, full_body);
}
```

Ο δεύτερος κανόνας ελέγχει το σώμα του αιτήματος (**full body**) για λέξεις-κλειδιά που υποδηλώνουν πιθανή **έγχυση κώδικα** (**Code Injection**), όπως `eval(), system(), ;` ή άλλες κακόβουλες εντολές. Αν τέτοια λέξη βρεθεί, εμφανίζεται ειδοποίηση που περιλαμβάνει τη διεύθυνση IP, την θύρα, και το πλήρες σώμα του αιτήματος.

```
[ALERT] Potential Code Injection detected.
```

περιλαμβάνοντας τη διεύθυνση IP (`c$id$orig_h`), την θύρα (`c$id$orig_p`), και το πλήρες σώμα του αιτήματος (`full_body`). Ο σκοπός αυτού του κανόνα είναι η αποτροπή επιθέσεων στις οποίες οι εισβολείς προσπαθούν να εκτελέσουν αυθαίρετο κώδικα στον διακομιστή, όπως εκτέλεση **Shell Commands**, **SQL Injection**, ή άλλες παρόμοιες επιθέσεις. Η ανίχνευση τέτοιων μοτίβων μπορεί να βοηθήσει στην πρόληψη ζημιών όπως η κλοπή δεδομένων ή η αλλοίωση λειτουργιών του συστήματος.

2.3.3 Έλεγχος μεγάλου Payload (buffer overflow)

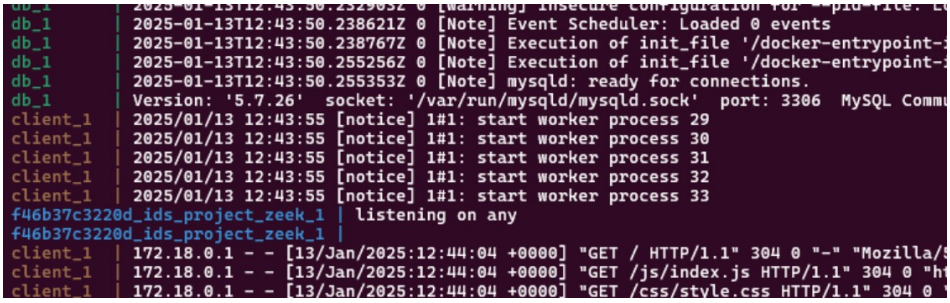
```
if (|full_body| > payload_threshold) {
    print fmt("[ALERT] Payload exceeds threshold.
    Host: %s:%d, Length: %s", c$id$orig_h, c$id$orig_p, |full_body|);
}
```

Ο τρίτος κανόνας ελέγχει το μέγεθος του σώματος του αιτήματος (payload) και το συγκρίνει με ένα **προκαθορισμένο όριο** (**payload threshold**). Αν το μέγεθος υπερβαίνει αυτό το όριο, ενεργοποιείται ειδοποίηση που περιλαμβάνει τη διεύθυνση IP, την θύρα, και το μήκος του σώματος.

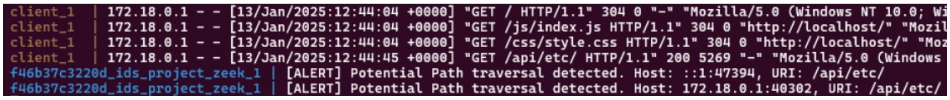
```
[ALERT] Payload exceeds threshold.
```

περιλαμβάνοντας τη διεύθυνση IP, την θύρα, και το μέγεθος του σώματος. Ο σκοπός αυτού του κανόνα είναι να εντοπίσει περιπτώσεις όπου οι εισβολείς στέλνουν υπερβολικά μεγάλα δεδομένα σε μια προσπάθεια να προκαλέσουν ζημιά, όπως μέσω επιθέσεων **Buffer Overflow** ή κατάχρησης πόρων του διακομιστή.

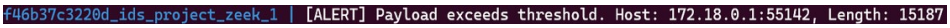
3 Στιγμιότυπα Οθόνης



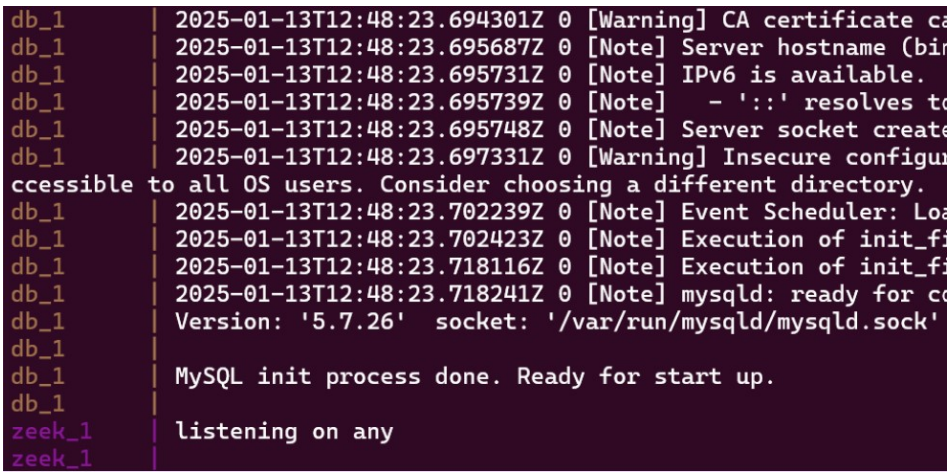
Εικόνα 3.1 Κανονική λειτουργία των `containers`



Εικόνα 3.2 Potential Path Traversal ALERT

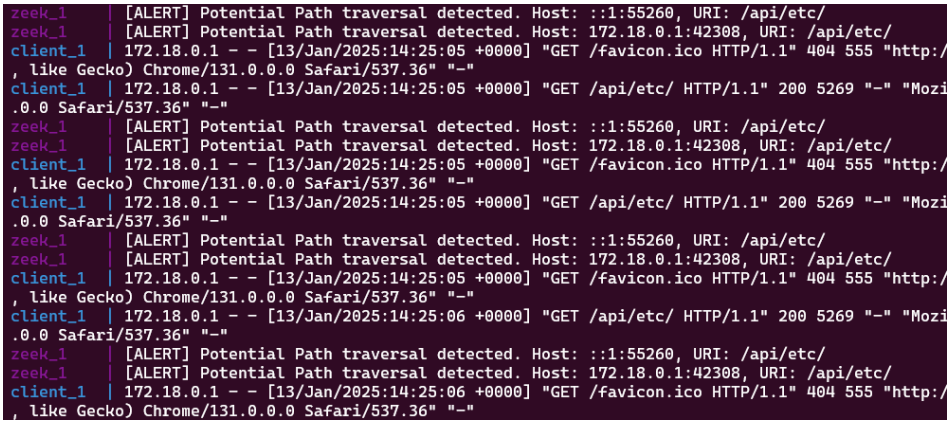


Εικόνα 3.3 Payload Exceeds Threshold

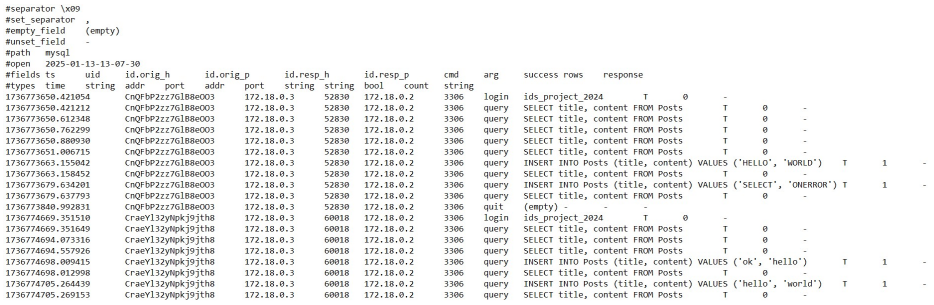


Εικόνα 3.4 Το **Zeek Container** αναμένοντας δικτυακή κίνηση

Εικόνα 3.5 Εντοπισμός νέου post από τον **Server Container**

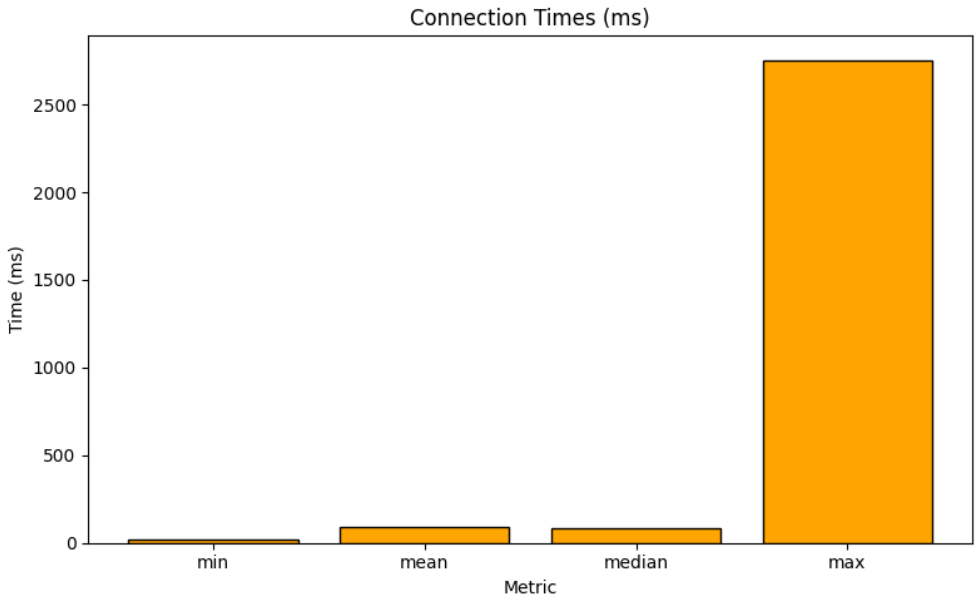


Εικόνα 3.6 Εντοπισμός πολλαπλών επιθέσεων από τον **Server Container**

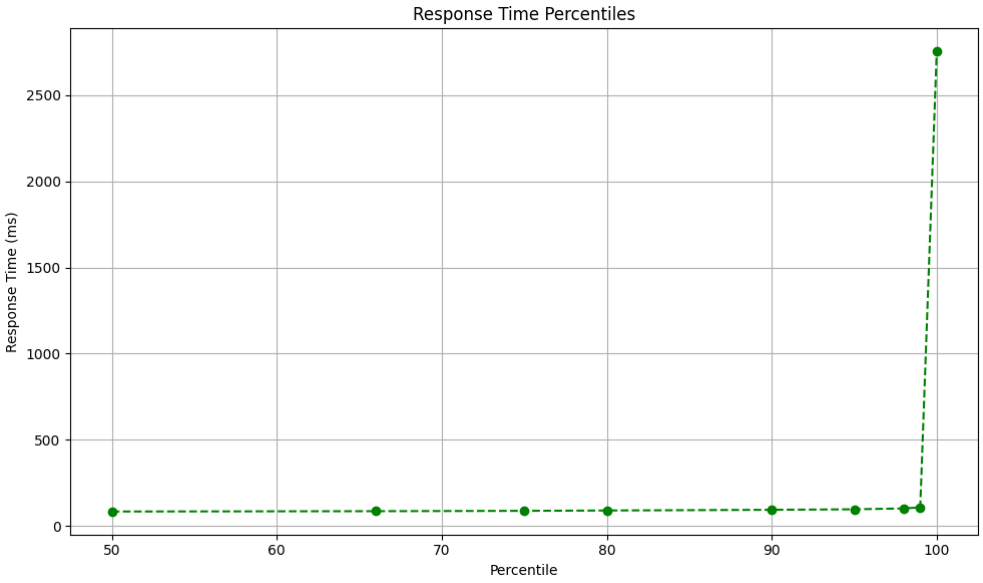


Εικόνα 3.7 mysql.log

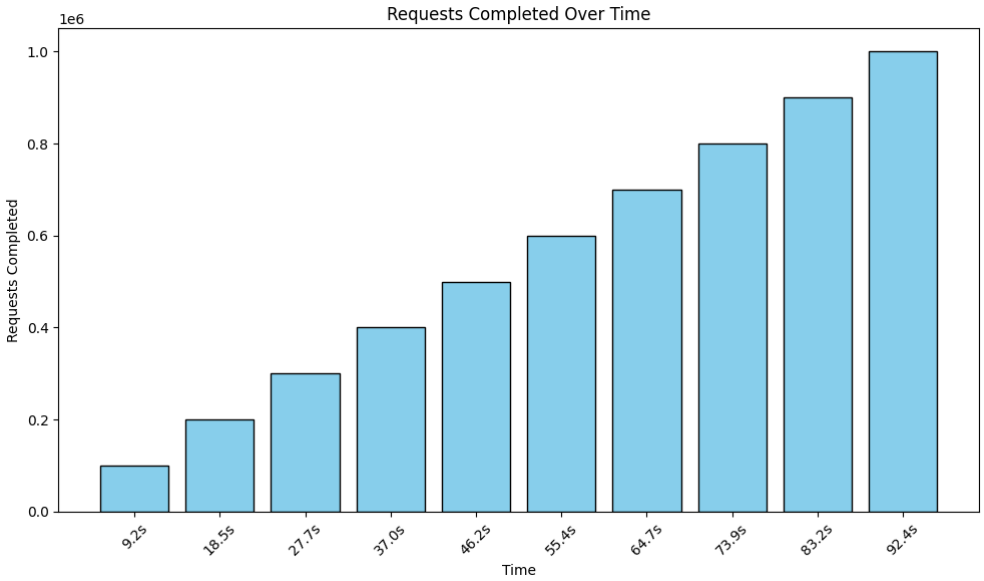
4 Οπτικοποίηση



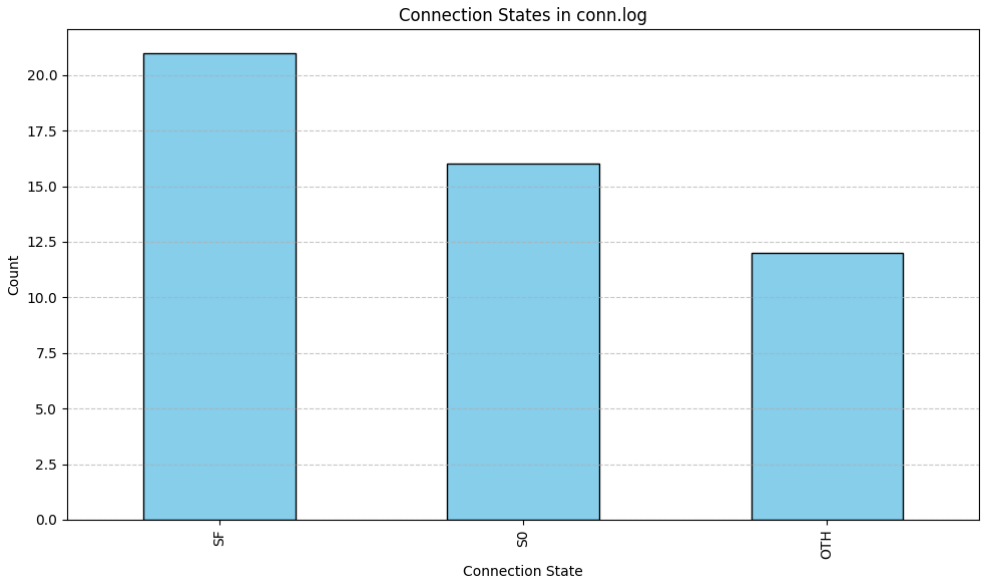
Εικόνα 4.1 Χρονικές καθυστερήσεις συνδέσεων (ms) από με Apache Bench



Εικόνα 4.2 Χρονικές καθυστερήσεις συνδέσεων (ms) από με Apache Bench (percentile)



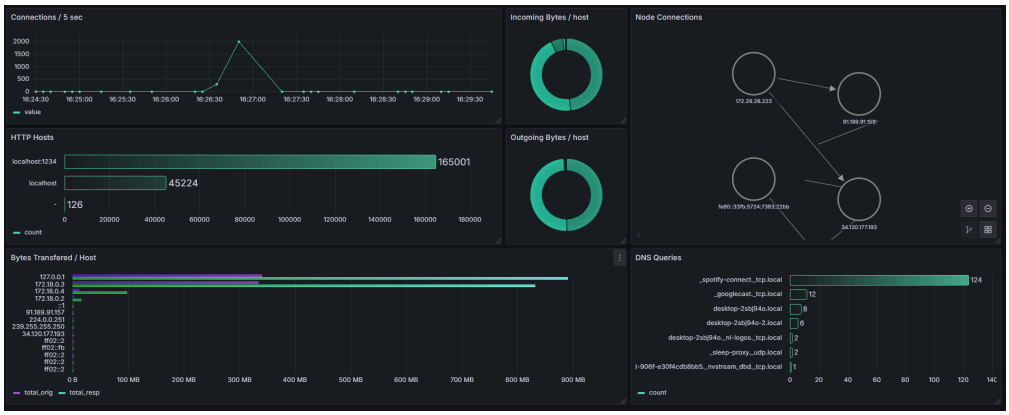
Εικόνα 4.3 Ολοκληρωμένα αιτήματα με την πάροδο του χρόνου



Εικόνα 4.4 Καταστάσεις σύνδεσης στο αρχείο conn.log: SF (Επιτυχής Σύνδεση), S0 (Προσπάθεια Σύνδεσης, Χωρίς Απάντηση), OTH (Άλλη Κατάσταση)

Προκειμένου να οπτικοποιηθούν τα alerts που έχει παράγει το Zeek, χρησιμοποιήθηκαν τα logs που προέκυψαν. Στη συνέχεια, χρησιμοποιήθηκαν οι βιβλιοθήκες pandas και matplotlib. Επιπρόσθετα, χρησιμοποιώντας οδηγό από το Github μετατράπηκαν τα log files της **Zeek** σε μία **sqlite3** βάση δεδομένων όπου μετεπειτα φορτώθηκε σε ένα υπάρχον dashboard σε ένα self hosted Grafana instance.

Εικόνα 4.5 Το UI του **Grafana**



Εικόνα 4.6 Το **Grafana** μετά από κάποια ώρα λειτουργίας

Αναφορές

[1] Go Documentation. Avoiding sql injection risk. https://go.dev/doc/database/sql-injection. [Ανακτήθηκε 13 Ιανουαρίου 2025].

[2] Go Documentation. Go-mysql-driver. https://pkg.go.dev/github.com/go-sql-driver/mysql#section-readme. [Ανακτήθηκε 13 Ιανουαρίου 2025].

[3] hackertarget. Zeek to sqlite. https://github.com/hackertarget/pcap-did-what/blob/master/zeek-docker/zeek-to-sqlite.py. [Ανακτήθηκε 13 Ιανουαρίου 2025].

[4] The University of Texas at San Antonio. Zeek intrusion detection series, lab 1: Introduction to the capabilities of zeek. https://research.cec.sc.edu/files/cyberinfra/files/Zeek_Lab_Series.pdf. [Ανακτήθηκε 13 Ιανουαρίου 2025].

[5] C. Sanders and J. Smith. *Applied Network Security Monitoring*. Elsevier Inc., 2014.

[6] Zeek. Zeek documentation. https://docs.zeek.org/en/master/. [Ανακτήθηκε 13 Ιανουαρίου 2025].

[7] Συλλογικός Τόμος. *Ασφάλεια Πληροφοριών και Συστημάτων στον Κυβερνοχώρο*. ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ, 1 edition, 2021.