Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**(Ελληνικά) Ανάπτυξη επέκτασης Chrome για την ανίχνευση κακόβουλης JavaScript** **(Αγγλικά) Chrome extension development for malicious JavaScript detection**
Ονοματεπώνυμο Φοιτητή	**Δήμητρα Ανδρομάχη Μπαρκούζου**
Πατρώνυμο	**Ευάγγελος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 13068**
Επιβλέπων	**Κωνσταντίνος Πατσάκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Μάιος 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Κωνσταντίνος Πατσάκης Επίκουρος Καθηγητής	Παναγιώτης Κοτζανικολάου Επίκουρος Καθηγητής	Ευθύμιος Αλέπης Επίκουρος Καθηγητής

Περίληψη

Στην παρούσα εργασία παρουσιάζεται η μελέτη και η έρευνα εργαλείων στατικής και δυναμικής ανάλυσης για τον εντοπισμό κακόβουλης JavaScript σε σελίδες του διαδικτύου. Στα πλαίσια αυτής της έρευνας υλοποιήθηκε μια επέκταση για τον φυλλομετρητή Chrome, η οποία ενσωματώνει τις λειτουργίες αυτών των εργαλείων ως μια προσπάθεια εντοπισμού των κρίσιμων επιθέσεων που βασίζονται στην γλώσσα σεναρίων JavaScript. Πιο συγκεκριμένα χρησιμοποιήθηκαν δύο εργαλεία, το JSDetox το οποίο πραγματοποιεί στατική ανάλυση και το jsunpack-n από την άλλη πλευρά που είναι εργαλείο δυναμικής ανάλυσης. Ο συνδυασμός των λειτουργιών και των δύο έχει ως αποτέλεσμα την εκμετάλλευση των πλεονεκτημάτων του καθενός, με απώτερο σκοπό την αποτελεσματικότερη ανίχνευση κακόβουλου κώδικα στις σελίδες που επισκέπτεται ο χρήστης. Στη συγκεκριμένη επέκταση περιλαμβάνεται και μια βασική συνάρτηση, η οποία ελέγχει το εάν ο εξεταζόμενος κώδικας είναι obfuscated ή όχι. Για να γίνει αυτό υπολογίζονται η εντροπία, το n-gram και το μέγεθος λέξης (wordsize), τα οποία μέσω μιας γραμμικής συνάρτησης δίνουν το κατώφλι, με βάση το οποίο γίνεται η ταξινόμηση το αν ο κώδικας είναι obfuscated ή όχι. Όλα τα παραπάνω έχουν υλοποιηθεί με τις βασικές Web τεχνολογίες JavaScript, HTML, AJAX και JQuery.

Στα επόμενα κεφάλαια θα γίνει αναφορά στις τεχνολογίες αυτές, καθώς επίσης και σε βασικά θέματα ασφάλειας της JavaScript, όπως ο τρόπος που οι επιτιθέμενοι την χρησιμοποιούν για να εκμεταλλευτούν το σύστημα του τελικού χρήστη και ο τρόπος που ένας κώδικας μπορεί να μετατραπεί σε obfuscated.

Τέλος, θα γίνει περιγραφή του τρόπου με τον οποίο σχεδιάστηκε η επέκταση, οι τροποποιήσεις που χρειάστηκαν να γίνουν στην εικονική μηχανή που φιλοξενεί τα εργαλεία με τα οποία συνεργάζεται και ο τρόπος λειτουργίας της.

Abstract

The purpose of this study is to investigate the current static and dynamic tools used for the detection of malicious JavaScript residing in different websites. Within the frames of the research an extension for the chrome browser was developed, which embeds the functions of these tools as an effort of detection of critical attacks originating from infected JavaScript code. More specifically two tools were used, JSDetox which conducts static analysis and the jsunpack-n which performs dynamic analysis. Combining the advantages of the two technologies mentioned, results in more efficient detection of malicious code in a website a user visits. In the specific extension a basic function which can detect obfuscated code, is implemented. This is achieved by calculating the entropy, n-gram and wordsize, filtered through a linear function resulting in a threshold for obfuscated code. All the above makes use of basic web technologies like JavaScript, HTML, AJAX and JQuery.

During the research and analysis presented throughout the thesis, the role of these technologies in detection of malicious and obfuscated code will be discussed thoroughly. Moreover, we will examine basic security issues of JavaScript, like the exploitation of these issues to attack the end user and how this code can become obfuscated. Concluding, a detailed analysis will be presented discussing the ways this extension was designed, developed and modified in order to communicate with a virtual machine, hosting the necessary tools to perform the necessary operations.

Πίνακας περιεχομένων

Κεφάλαιο 1ο

1. Εισαγωγή

Οι επιθέσεις που βασίζονται στην γλώσσα σεναρίων JavaScript είναι οι πιο συχνά παρατηρούμενες στο διαδίκτυο. Αυτό συμβαίνει διότι, μέσω της JavaScript οι επιτιθέμενοι μπορούν να διαπεράσουν τα antivirus που βασίζονται στην στατική υπογραφή για την ανίχνευση των κακόβουλων λογισμικών μετατρέποντας τον κώδικα σε μπερδεμένη μορφή (obfuscated), έτσι ώστε να μην ανιχνεύεται εύκολα. Επίσης την JavaScript οι χρήστες την εκτελούν στα συστήματά τους χωρίς δεύτερη σκέψη. Αυτό δίνει το πλεονέκτημα στους επιτιθέμενους και την ευκολία ώστε να ενορχηστρώσουν την επίθεσή τους με μεγαλύτερη ευκολία, αφού την χρησιμοποιούν ως πρώτο βήμα το οποίο τους ανοίγει την «πόρτα» για την περαιτέρω εκμετάλλευση των συστημάτων στόχων. Επίσης τα δυναμικά χαρακτηριστικά της συγκεκριμένης γλώσσας παρ' όλα τα πλεονεκτήματα που προσδίδουν σε μια ιστοσελίδα, γίνονται σημεία εκμετάλλευσης και εμπεριέχουν πολλούς κινδύνους ασφάλειας.

Όλα τα παραπάνω καθιστούν την JavaScript ως την τέλεια υποψήφια γλώσσα μέσω της οποίας οι επιθέσεις έχουν μεγαλύτερο ποσοστό επιτυχίας. Αποτέλεσμα αυτού παρατηρείται μεγάλη ανάγκη στον κλάδο της ασφάλειας η ύπαρξη αποτελεσματικών εργαλείων για την ανίχνευση κακόβουλου κώδικα. Έχουν προταθεί και αναπτυχθεί στην σχετική βιβλιογραφία διάφορες τεχνικές και εργαλεία τα οποία βοηθούν αυτό τον σκοπό. Κάποια από αυτά βασίζονται στην στατική ανάλυση του εξεταζόμενου κώδικα, ενώ άλλα στην δυναμική ανάλυση. Και στις δύο περιπτώσεις υπάρχουν περιορισμοί τα οποία δεν τα καθιστούν πλήρως αποτελεσματικά. Άλλα εργαλεία που έχουν υλοποιηθεί βασίζονται στην μηχανική μάθηση, τα οποία εκπαιδεύονται μέσω ενός συνόλου δεδομένων και κανόνων για τον πιο ακριβή εντοπισμό του κακόβουλου κώδικα μέσω τεχνικών ταξινόμησης.

Το πρόβλημα όμως, αποτελεί το γεγονός ότι δεν υπάρχει κάποιο εργαλείο το οποίο οι χρήστες του διαδικτύου θα μπορούσαν να χρησιμοποιήσουν έτσι ώστε να είναι δυνατόν με αυτοματοποιημένο τρόπο να εντοπίζει σε κάποιο βαθμό την κακόβουλη δραστηριότητα, καθώς όλα τα παραπάνω προϋποθέτουν την γνώση και τον χειρισμό από έμπειρους ερευνητές ασφάλειας. Συνεπώς σε αυτήν την εργασία προτείνεται η υλοποίηση μιας επέκτασης στον φυλλομετρητή Chrome η οποία θα συνδυάζει τις λειτουργίες δύο βασικών εργαλείων στατικής και δυναμικής ανάλυσης (JSDetox, jsunpack-n), η οποία από την μια θα εξαλείφει τα μειονεκτήματα και από την άλλη θα αυτοματοποιεί τις λειτουργίες τους έτσι ώστε οποιοσδήποτε χρήστης θα μπορεί να το εγκαθιστά και να το λειτουργεί χωρίς κάποια επιπρόσθετη δυσκολία.

1.1 Περιγραφή του υπό μελέτη προβλήματος

Οι επιθέσεις που βασίζονται στην JavaScript γίνονται όλο και πιο δύσκολες στην ανίχνευσή τους, καθώς οι επιτιθέμενοι μηχανεύονται συνέχεια καινούριους τρόπους ώστε να μην γίνονται αντιληπτές από τα υπάρχοντα συστήματα antivirus. Αυτό γίνεται όπως έχει προαναφερθεί με τον μετασχηματισμό του κώδικα σε μορφή η οποία δεν μπορεί να διαβαστεί αλλά η εκτέλεσή του έχει ακριβώς το ίδιο αποτέλεσμα. Η τεχνική αυτή ονομάζεται obfuscation. Λόγω των πολλών διαφορετικών τεχνικών και επιπέδων obfuscation η πραγματική φύση και ο σκοπός ενός κακόβουλου κώδικα είναι δύσκολο να καθοριστεί. Μια λύση σε αυτό το πρόβλημα θα ήταν να γίνει η υπόθεση ότι αν υπάρχει κάποιο τέτοιο κομμάτι κώδικα σε κάποια ιστοσελίδα να θεωρηθεί άμεσα κακόβουλος. Όμως αυτή η τεχνική χρησιμοποιείται και από πολλούς προγραμματιστές ώστε να αποτρέψουν την κλοπή του κώδικά τους. Συνεπώς γεννιέται η ανάγκη για την ύπαρξη κάποιου λογισμικού, εργαλείου ή τεχνικής η οποία θα

κάνει αυτόν τον διαχωρισμό και θα μπορεί να ανιχνεύει με επιτυχία την ύπαρξη κακόβουλου κώδικα και να μπορεί να προειδοποιεί και να προστατεύει τους χρήστες όταν επισκέπτονται κάποια ιστοσελίδα.

1.2 Σκοπός και στόχοι της εργασίας

Στόχος της εργασίας είναι η έρευνα σε καινούριες τεχνικές και εργαλεία που αφορούν την ανίχνευση κακόβουλου κώδικα JavaScript καθώς, και η δημιουργία ενός αυτοματοποιημένου εργαλείου ως εναλλακτική πρόταση στις υπάρχουσες που αναφέρονται στην σχετική βιβλιογραφία.

1.3 Παραδοτέα της εργασίας

Η παρούσα εργασία θα παραδοθεί στα εξής επιμέρους τμήματα :

1. Το έντυπο κείμενο της εργασίας που περιλαμβάνει το θεωρητικό τμήμα των τεχνολογιών που χρησιμοποιήθηκαν για την δημιουργία της επέκτασης, αλλά και η περιγραφή βασικών θεμάτων ασφάλειας JavaScript μέσω της οποίας γίνονται οι επιθέσεις στην ανίχνευση των οποίων στοχεύει η επέκταση.
2. Ο κώδικας που αναπτύχθηκε για την δημιουργία της επέκτασης Chrome, σε Back-end και Front-end.

1.4 Δομή της εργασίας

Η εργασία έχει διαιρεθεί σε τρία βασικά μέρη. Στο πρώτο μέρος γίνεται μια σύντομη επισκόπηση των τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση της επέκτασης. Στο επόμενο κεφάλαιο γίνεται αναφορά σε βασικά θέματα ασφάλειας που προκύπτουν μέσω της JavaScript, όπως οι κυριότερες και πιο κοινές επιθέσεις που πραγματοποιούνται, οι τρόποι με τους οποίους γίνεται η εκμετάλλευση (exploitation) του συστήματος του χρήστη και οι τρόποι με τους οποίους οι επιτιθέμενοι μετατρέπουν τον κώδικα σε obfuscated για να καλύψουν τον πραγματικό σκοπό τους.

Στο τελευταίο μέρος γίνεται ανάλυση στον τρόπο με τον οποίο υλοποιήθηκε η επέκταση, μαζί με αναφορά στα εργαλεία ανίχνευσης κακόβουλης JavaScript που χρησιμοποιήθηκαν, η τροποποίηση που χρειάστηκε να γίνει στην εικονική μηχανή, στο δίκτυο αλλά ακόμα και στα ίδια τα εργαλεία. Επίσης αναλύεται η βασική συνάρτηση που κατασκευάστηκε η οποία ελέγχει το εάν ο κώδικας είναι obfuscated ή όχι. Τέλος παρατίθενται οι εικόνες με τα τελικά αποτελέσματα αυτής της υλοποίησης.

Κεφάλαιο 2ο

2. Επισκόπηση τεχνολογιών υλοποίησης

Στο πρώτο κεφάλαιο θα αναλυθούν οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της επέκτασης chrome που πραγματοποιεί την ανάλυση για τον έλεγχο της ύπαρξης ή μη κακόβουλης JavaScript. Οι τεχνολογίες αυτές συνοψίζονται παρακάτω:

1. JavaScript
2. Document Object Model (DOM)
3. AJAX
4. JQuery
5. PHP
6. Bootstrap
7. Chrome Extension

2.1 JavaScript

Η JavaScript γνωστή και ως γλώσσα σεναρίων (scripting language) αποτελεί μια lightweight, δυναμική, διερμηνευμένη γλώσσα προγραμματισμού με βασικό χαρακτηριστικό ότι είναι prototype based, multi-paradigm (όταν για την επίλυση ενός προβλήματος ο προγραμματιστής μπορεί να επιλέξει αντικειμενοστραφές ή διαδικαστικό τρόπο) και χρησιμοποιεί συναρτήσεις πρώτης κλάσης. Η JavaScript εκτελείται στον φυλλομετρητή στην πλευρά του πελάτη κατά κύριο λόγο, αν και η χρήση της στην πλευρά του εξυπηρετητή γίνεται με την χρήση πλατφόρμας ανάπτυξης λογισμικού, όπως το node.js. Βέβαια λειτουργεί και σε περιβάλλοντα εκτός φυλλομετρητών, όπως σε pdf και σε desktop widgets.

Σε συνδυασμό με την γλώσσα επισήμανσης HTML και το CSS αποτελούν τον πυρήνα των σελίδων του ιστού, η οποία διαχειρίζεται στο πως μια σελίδα ανταποκρίνεται στην εμφάνιση κάποιου γεγονότος (event). Επίσης καθιστά δυνατή την ανταλλαγή δεδομένων με τρόπο ασύγχρονο και στην αλλαγή του περιεχομένου ενός εγγράφου με δυναμικό τρόπο που περιλαμβάνει την κατασκευή αντικειμένων κατά τον χρόνο εκτέλεσης και την δυναμική δημιουργία σεναρίου με την χρήση ειδικών συναρτήσεων όπως η eval ( ).

Μερικά παραδείγματα προγραμματισμού JavaScript ακολουθούν παρακάτω:

```
<!DOCTYPE html>

<title>My Example</title>




<time id="date"></time>
```

```
<script>
 /*
 Create a JavaScript Date object for the current date and time,
 then extract the desired parts, then join them again in the desired format.
 */
 var currentDate = new Date(),
    day = currentDate.getDate(),
    month = currentDate.getMonth() + 1,
    year = currentDate.getFullYear(),
    date = day + "/" + month + "/" + year;

 // Output the date to the above HTML element
 document.getElementById("date").innerHTML = date;
</script>
```

Παράδειγμα κώδικα JavaScript για την εμφάνιση ημερομηνίας [18]

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to evaluate/execute JavaScript code/expressions.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>
```

```
    <script>
    function myFunction() {
       var x = 10;
       var y = 20;
       var a = eval("x * y") + "<br>";
       var b = eval("2 + 2") + "<br>";
       var c = eval("x + 17") + "<br>";

       var res = a + b + c;
       document.getElementById("demo").innerHTML = res;
    }
    </script>

    </body>
</html>
```

Παράδειγμα χρήσης της ειδικής συνάρτησης eval ( ) [18]

```
    <!DOCTYPE html>
    <html>
    <body>
    <p>Javascript example with objects</p>

    <script>
    var myObj = new Object(),
       str = 'myString',
       rand = Math.random(),
       obj = new Object();

    myObj.type            = 'Dot syntax';
    myObj['date created']   = 'String with space';
    myObj[str]            = 'String value';
    myObj[rand]           = 'Random Number';
```

```
    myObj[obj]          = 'Object';
    myObj['']           = 'Even an empty string';

    console.log(myObj);
    </script>
    </body>
</html>
```

Παράδειγμα δημιουργίας αντικειμένων [18]

```
    <!DOCTYPE html>
    <html>
    <body>

    <p id="demo"></p>

    <script>
    function Person(first, last, age, eye) {
      this.firstName = first;
      this.lastName = last;
      this.age = age;
      this.eyeColor = eye;
    }

    var myFather = new Person("John", "Doe", 50, "blue");
    var myMother = new Person("Sally", "Rally", 48, "green");

    document.getElementById("demo").innerHTML =
    "My father is " + myFather.age + ". My mother is " + myMother.age;
    </script>

    </body>
</html>
```

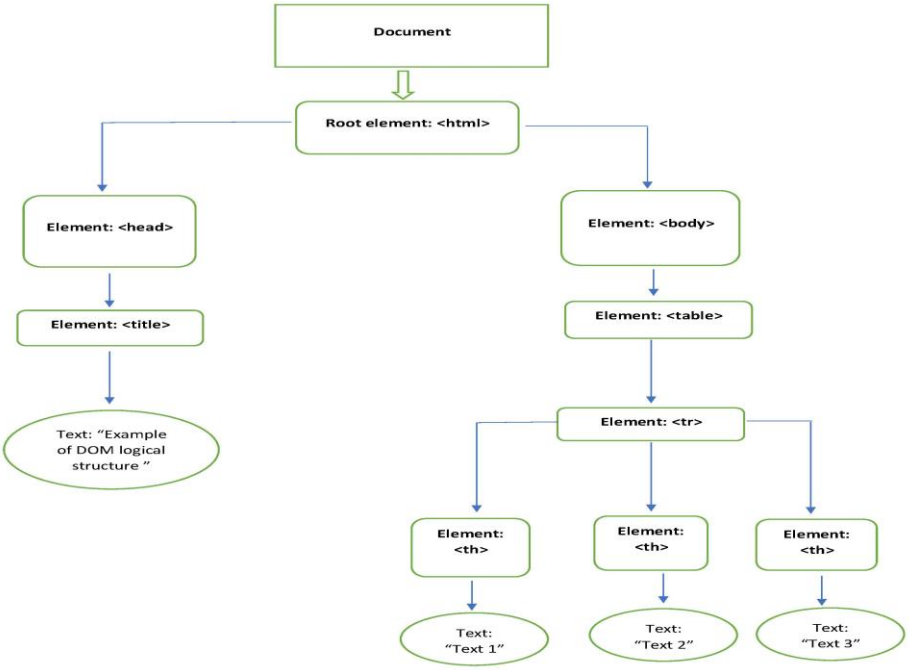
Παράδειγμα prototype based programming [18]

2.2 Document Object Model (Dom)

Το μοντέλο αντικειμένου εγγράφου (DOM) αποτελεί ένα API το οποίο χρησιμοποιείται για να ορίσει την λογική δομή των εγγράφων HTML και XML. Αποτελείται από αντικείμενα, στοιχεία τα οποία διαθέτουν ιδιότητες και μεθόδους, έτσι ώστε να μπορούν να τροποποιηθούν με την βοήθεια της JavaScript. Η λογική δομή του συγκεκριμένου μοντέλου αντιστοιχεί σε ένα δέντρο το οποίο διαθέτει ρίζα, ενδιάμεσους κόμβους και φύλα. Αυτή η δομή βοηθά στην προσπέλαση κάθε στοιχείου ή

περιεχομένου του δέντρου, έτσι ώστε ο προγραμματιστής να μπορεί να το διαγράψει, να προσθέσει ένα καινούριο ή να το τροποποιήσει. Το DOM υποστηρίζεται από κάθε προγραμματιστική γλώσσα και διαφέρει από τον κώδικα σελίδας, καθώς αποτελεί την αντικειμενοστραφή παρουσίαση της σελίδας. Ο περιηγητής όταν διαβάζει μια HTML σελίδα, ουσιαστικά μετατρέπει τον κώδικα σε DOM στοιχεία, όπως φαίνεται στο παρακάτω παράδειγμα:

```
    <html>
    <head>
    <title>Example of DOM logical structure</title>
    </head>
    <body>
    <table>
     <tr>
      <th>Company</th>
      <th>Contact</th>
      <th>Country</th>
     </tr>
     </table>

    </body>
</html>
```



Εικόνα 1. Δεντρική δομή μοντέλου αντικειμένου εγγράφου (DOM)

Οι κύριοι τύποι δεδομένων της διεπαφής DOM συνοψίζονται στον παρακάτω πίνακα:

document	Αναφέρεται στην ρίζα, δηλαδή στο έγγραφο στο οποίο θα γίνει κάποια ενέργεια, όπως προσθήκη , διαγραφή ή τροποποίηση κάποιου στοιχείου.
element	Αναφέρεται σε στοιχείο ή σε κάποιο κόμβο του δέντρου.
nodeList	Αποτελεί έναν πίνακα με τα στοιχεία.
attribute	Οι ιδιότητες των στοιχείων/κόμβων.
namedNodeMap	Αποτελεί ένα είδος πίνακα, αλλά τα αντικείμενα προσπελαύνονται, είτε από το όνομα, είτε από ευρετήριο (index)

Πίνακας 1. NoSQL Databases

Στην συνέχεια δίνονται κάποια παραδείγματα δημιουργίας, εύρεσης, διαγραφής ή τροποποίησης στοιχείων σε ένα έγγραφο σαν αυτά που υλοποιήθηκαν για τις ανάγκες της εργασίας.

Έστω ότι έχουμε ένα κενό div id result, το οποίο ορίζεται ως:

```
<div id="result"> </div>
```

Μέσω του DOM μπορούμε να το εντοπίσουμε και να του προσθέσουμε περιεχόμενο ως εξής:

```
<script>
  var container = document.getElementById("result");
  result.innerHTML = "New Content!";
</script>
```

Στο επόμενο παράδειγμα παρουσιάζεται η προσθήκη ενός νέου στοιχείου στην δεντρική δομή του εγγράφου.

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to make a BUTTON element with text.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var btn = document.createElement("BUTTON");
  var t = document.createTextNode("CLICK ME");
  btn.appendChild(t);
  document.body.appendChild(btn);
}
</script>

</body>
</html>
```

Στο παραπάνω παράδειγμα όταν ο χρήστης πατάει το κουμπί Try it, ενεργοποιείται η συνάρτηση myFunction, η οποία δημιουργεί ένα νέο στοιχείο τύπου button που ονομάζεται CLICK ME και με την βοήθεια των μεθόδων appendChild, προσαρτείται στον γονέα του το οποίο γίνεται αλυσιδωτά προς την ρίζα. Δηλαδή, το κουμπί προσαρτείται στον TextNode και στην συνέχεια στο στοιχείο body.

2.3 AJAX (Asynchronous JavaScript and XML)

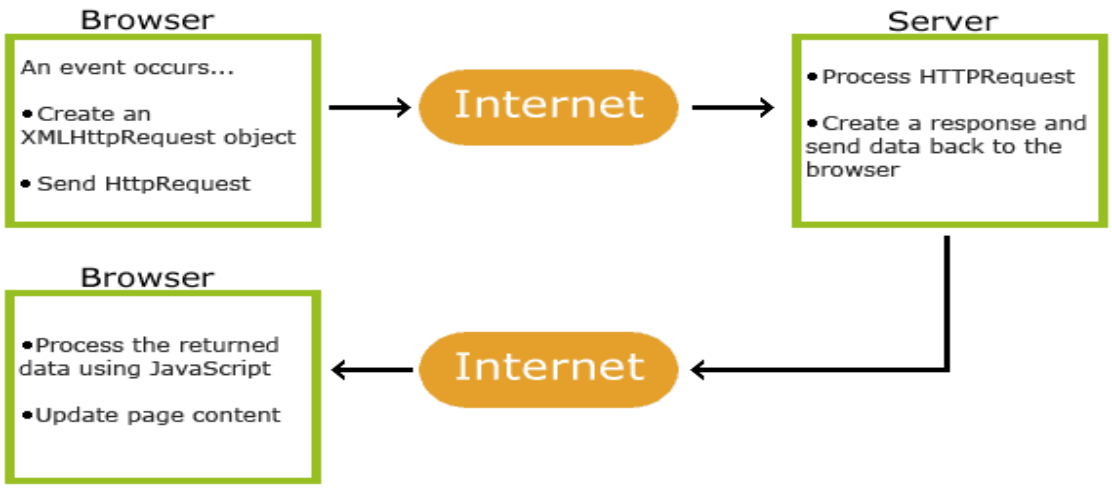
Ένα από τα βασικά σημεία της παρούσας εργασίας αποτελούν οι ασύγχρονες κλήσεις JavaScript στα εργαλεία που είναι εγκατεστημένα στην εικονική μηχανή. Αυτό επιτυγχάνεται με την βοήθεια της τεχνολογίας AJAX. Αποτελεί ένα σύνολο τεχνολογιών που λειτουργούν, όπως και στην JavaScript στην πλευρά του πελάτη ,έτσι ώστε να δημιουργούνται ασύγχρονες web εφαρμογές.

Όταν αναφέρουμε τον όρο ασύγχρονο σημαίνει ότι γίνεται αλληλεπίδραση με την πλευρά του εξυπηρετητή χωρίς να απαιτείται από τον χρήστη να περιμένει την ολοκλήρωσή τους για να μπορεί να επενεργήσει με την σελίδα στην οποία βρίσκεται. Οπότε επιστρέφονται τα όποια αποτελέσματα ή δεδομένα από τον εξυπηρετητή χωρίς να φορτώνεται εκ νέου όλη η σελίδα στην πλευρά του πελάτη. Πιο συγκεκριμένα, τα δεδομένα ζητούνται από τον server και επιστρέφουν αφού έχει φορτωθεί η σελίδα. Μπορεί να γίνει update στο περιεχόμενο της σελίδας, χωρίς να πρέπει να φορτωθεί εκ νέου και να αποσταλούν δεδομένα στον εξυπηρετητή στο παρασκήνιο.

Το σύνολο των τεχνολογιών που απαρτίζουν την τεχνολογία AJAX αποτελείται από:

1. HTML, CSS τα οποία χρησιμοποιούνται για τη σήμανση και τον καθορισμό του τρόπου παρουσίασης των τελικών αποτελεσμάτων στο χρήστη.
2. Η διεπαφή DOM, η οποία με την βοήθεια της JavaScript επεξεργάζεται τα μέρη του εγγράφου τα οποία θα παρουσιάσουν τα αποτελέσματα από την κλήση.
3. Το αντικείμενο XMLHttpRequest που είναι απαραίτητο για την δημιουργία της ίδιας της κλήσης και καθορίζει όλη την επικοινωνία με τον εξυπηρετητή.
4. Τα format JSON και XML τα οποία καθορίζουν τον τρόπο αναπαράστασης των επιστρεφόμενων δεδομένων. Θα πρέπει να σημειωθεί ότι το JSON αποτελεί το πιο διαδεδομένο format, αλλά υπάρχουν και επιλογές, όπως HTML ή απλό κείμενο που μπορούν να χρησιμοποιηθούν.
5. JavaScript που είναι ο συνδετικός κρίκος, ώστε να λειτουργήσουν όλα τα παραπάνω.

Τα βήματα που ακολουθούνται για την πραγματοποίηση μιας κλήσης φαίνονται στην παρακάτω εικόνα:



Εικόνα 2. Διάγραμμα κλήσης AJAX [17]

Ένα παράδειγμα της χρήσης όλων των προαναφερθέντων ακολουθεί παρακάτω [17]:

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

Τα πλεονεκτήματα της τεχνολογίας AJAX παρουσιάζονται στον παρακάτω πίνακα:

Callbacks (συναρτήσεις που καλούνται μέσα σε κάποια άλλη συνάρτηση)	Ελαχιστοποιείται η χρήση του δικτύου και επιτυγχάνονται πιο γρήγορα οι λειτουργίες.
Ασύγχρονες κλήσεις	Ο χρήστης δεν χρειάζεται να περιμένει την απάντηση από τον εξυπηρετητή ώστε να πρέπει να περιμένει για να μπορεί να αλληλοεπιδράσει με την σελίδα.
Αυξημένη ταχύτητα	Με την Ajax επιτυγχάνεται βελτίωση της ταχύτητας, η απόδοση και η ευχρηστία μιας web εφαρμογής.

2.4 JQuery

Αποτελεί μια lightweight βιβλιοθήκη της JavaScript με πληθώρα εύχρηστων χαρακτηριστικών και συναρτήσεων. Αποτελεί βασικό μέρος πολλών web εφαρμογών, καθώς επιτυγχάνεται η επίλυση βασικών προβλημάτων σε πολύ λιγότερες γραμμές κώδικα από αυτές που θα χρειάζονταν αν ο προγραμματιστής χρησιμοποιούσε κλασική JavaScript. Στην παρούσα εργασία χρησιμοποιήθηκε η συγκεκριμένη βιβλιοθήκη για την δημιουργία AJAX κλήσεων και τροποποίησης των στοιχείων του εγγράφου. Περιλαμβάνει τα ακόλουθα χαρακτηριστικά:

1. HTML/DOM/CSS διαχείριση
2. HTML μεθόδους για γεγονότα
3. Effects and animations
4. AJAX
5. Utilities
6. JSON parsing
7. Επεκτασιμότητα μέσω Plug-ins
8. Υποστήριξη από πολλούς φυλλομετρητές

Τα πλεονεκτήματά της είναι:

1. Διαχωρισμός JavaScript και HTML: λόγω του ότι η σύνταξή της είναι απλή είναι πιο εύκολη η προσθήκη χειριστών γεγονότων (event handlers) στο DOM από την προσθήκη γεγονότων στην HTML για την κλήση κάποιας JavaScript συνάρτησης. Με αυτόν τον τρόπο χωρίζεται η JavaScript από την HTML.
2. Συντομία και καθαρότητα: επιτυγχάνεται με την δημιουργία αλληλένδετων συναρτήσεων και με την χρήση σύντομων ονομάτων συναρτήσεων.
3. Εξάλειψη ασυμβατοτήτων μεταξύ των διάφορων περιηγητών: λειτουργεί σε διαφορετικούς φυλλομετρητές χωρίς προβλήματα σε αντίθεση με την JavaScript, της οποίας η μηχανή μεταγλώττισης διαφέρει από φυλλομετρητή σε φυλλομετρητή με αποτέλεσμα ο κώδικας να έχει πρόβλημα σωστής εκτέλεσης.
4. Επεκτασιμότητα

   Παρακάτω ακολουθεί ένα παράδειγμα κλήσης AJAX μέσω της βιβλιοθήκης JQuery:

```
var jqxhr =
  $.ajax({
    url: "/theServiceToCall.html",
    data: {
      name : "The name",
      desc : "The description"
    }
  })
  .done  (function(data, textStatus, jqXHR)       { alert("Success: " + response) ; })
  .fail  (function(jqXHR, textStatus, errorThrown) { alert("Error")   ; })
  .always(function(jqXHROrData, textStatus, jqXHROrErrorThrown)    { alert("complete"); })
;
```

2.5 PHP (Hypertext Pre-Procesor)

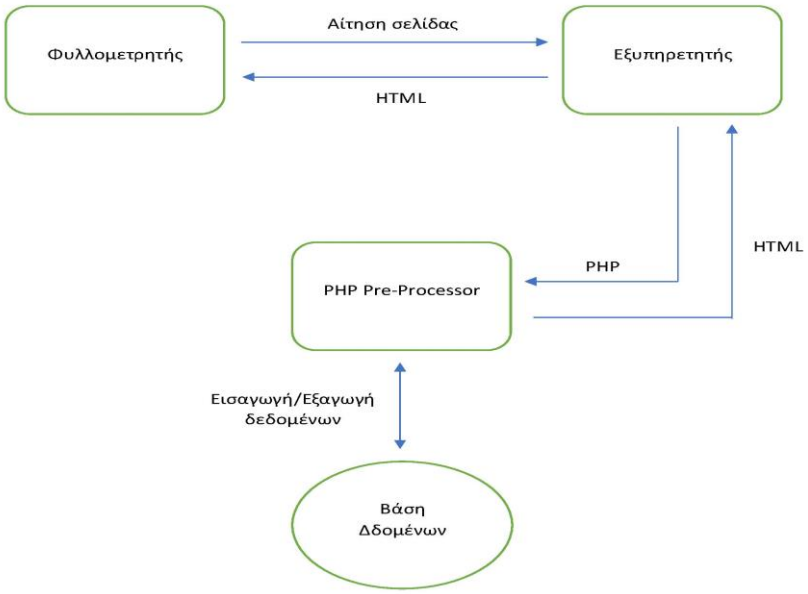
Η PHP αποτελεί και αυτή μια γλώσσα προγραμματισμού σεναρίων, η οποία χρησιμοποιείται για την δημιουργία δυναμικών ιστοσελίδων. Η διαφορά της με την JavaScript είναι ότι εκτελείται στην πλευρά του εξυπηρετητή και όχι του πελάτη. Μπορεί να ενσωματωθεί στον HTML κώδικα και όταν εκτελεστεί να επιστρέψει μαζί με την εκτελέσιμη μορφή της HTML, εικόνες, αρχεία όπως PDF, XHTML, XML ακόμα και απλό κείμενο.

Η συγκεκριμένη γλώσσα διαθέτει αρκετές λειτουργίες χρήσιμες στις web εφαρμογές, όπως η ανταλλαγή δεδομένων με βάσεις δεδομένων στις οποίες μπορεί να προσθέσει, να αφαιρέσει και να τροποποιήσει εγγραφές. Υποστηρίζει πρωτόκολλα επικοινωνίας με υπηρεσίες όπως LDAP, IMAP, SNMP, NNTP, POP3, HTTP και άλλα. Χειρίζεται αρχεία στον εξυπηρετητή, όπου μπορεί να ανοίξει κάποιο νέο, να το διαβάσει, να γράψει σε αυτό, να το διαγράψει και να το κλείσει. Συλλέγει δεδομένα, στέλνει και λαμβάνει cookies και κωδικοποιεί τα δεδομένα.

Συνήθως στις σελίδες η πιο συχνή υλοποίησή της αφορά την δημιουργία φορμών για είσοδο χρήστη στο σύστημα, όπου επικοινωνεί με την βάση δεδομένων. Τα πλεονεκτήματά της είναι ότι

λειτουργεί σε διάφορα λειτουργικά συστήματα, καθώς και ότι είναι συμβατή με αρκετές βάσεις δεδομένων και υποστηρίζεται από αρκετούς εξυπηρετητές.

Παρακάτω ακολουθεί η εικόνα που περιγράφει τον τρόπο λειτουργίας της:



Εικόνα 3. Διάγραμμα λειτουργίας της PHP

Ένα παράδειγμα της PHP για την δημιουργία zip αρχείου ακολουθεί παρακάτω [16]:

```
<?php
```

```
    $zip = new ZipArchive();
    $filename = "./test112.zip";
    if ($zip->open($filename, ZipArchive::CREATE)!==TRUE) {
       exit("cannot open <$filename>\n");
    }

    $zip->addFromString("testfilephp.txt" . time(), "#1 This is a test string added as
testfilephp.txt.\n");
    $zip->addFromString("testfilephp2.txt" . time(), "#2 This is a test string added as
testfilephp2.txt.\n");
    $zip->addFile($thisdir . "/too.php","/testfromfile.php");
    echo "numfiles: " . $zip->numFiles . "\n";
    echo "status:" . $zip->status . "\n";
    $zip->close();
?>
```

2.6 Bootstrap

Είναι ένα web framework ανοιχτού κώδικα για τον σχεδιασμό web εφαρμογών και σελίδων. Περιέχει HTML και CSS templates για βασικά σχεδιαστικά στοιχεία που μπορεί να περιέχει μια σελίδα, όπως κουμπιά, φόρμες, γραμματοσειρές, μπάρες πλοήγησης αλλά και διάφορες επεκτάσεις JavaScript. Σε αντίθεση με άλλα frameworks αυτό είναι μόνο front-end. Για την σχεδίαση μιας σελίδας με bootstrap απαιτούνται τα εξής αρχεία:

1. Bootstrap.css, το CSS framework
2. Bootstrap.js, το JavaScript framework
3. Glyphicons, ένα σύνολο από fonts

Ένα παράδειγμα bootstrap ακολουθεί [19]:

```
    <!DOCTYPE html>
    <html lang="en">
     <head>
       <meta charset="utf-8">
       <meta http-equiv="X-UA-Compatible" content="IE=edge">
       <meta name="viewport" content="width=device-width, initial-scale=1">
       <title>Bootstrap 101 Template</title>
       <link href="css/bootstrap.min.css" rel="stylesheet">
       <!--[if lt IE 9]>
         <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
         <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
       <![endif]-->
     </head>
```

```
    <body>
     <h1>Hello, world!</h1>

     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
     <script src="js/bootstrap.min.js"></script>
    </body>
</html>
```

2.7 Επέκταση Chrome

Στην παρούσα εργασία υλοποιήθηκε μια επέκταση στον φυλλομετρητή chrome. Οι επεκτάσεις αυτές αποτελούν μικρά προγράμματα λογισμικού τα οποία απαρτίζονται από τεχνολογίες ιστού, όπως αυτές που περιεγράφηκαν παραπάνω και μπορούν να τροποποιήσουν και να βελτιώσουν την λειτουργικότητα του συγκεκριμένου φυλλομετρητή. Ουσιαστικά είναι ένα πακέτο τεχνολογιών σε ένα και μόνο αρχείο, το οποίο ο χρήστης εγκαθιστά στον περιηγητή του. Στις επεκτάσεις του chrome υπάρχει το manifest.json, το οποίο αποτελεί ένα αρχείο μεταδεδομένων σε μορφή JSON και περιέχει ιδιότητες όπως το όνομα της επέκτασης, την περιγραφή του, την έκδοσή του, το τι μπορεί να κάνει η επέκταση, τον καθορισμό δικαιωμάτων, την εισαγωγή εξωτερικών βιβλιοθηκών και άλλα. Συνήθως, περιλαμβάνει το browser action στο οποίο δηλώνεται ποια εικόνα θα έχει η επέκταση καθώς και όταν ανοίξει η επέκταση ποια HTML σελίδα θα τρέξει. Ένα παράδειγμα του πως μοιάζει ένα αρχείο manifest ακολουθεί παρακάτω [20]:

```
{
 "manifest_version": 2,

 "name": "Getting started example",
 "description": "This extension shows a Google Image search result for the current page",
 "version": "1.0",

 "browser_action": {
  "default_icon": "icon.png",
  "default_popup": "popup.html"
 },
 "permissions": [
  "activeTab",
  "https://ajax.googleapis.com/"
 ]
}
```

Για την ανάπτυξή του ο προγραμματιστής ακολουθεί τα βασικά βήματα, όπως στην υλοποίηση μιας οποιαδήποτε άλλης web εφαρμογής εκτός από την χρήση βασικών APIS τα οποία βοηθούν στο

πώς ο προγραμματιστής επιθυμεί να ελέγξει την συμπεριφορά του φυλλομετρητή ανάλογα με τις ανάγκες του.

Κεφάλαιο 3ο

3. Τρόποι εκμετάλλευσης μέσω JavaScript και ανάλυση τεχνικών obfuscation

Τα δυναμικά χαρακτηριστικά της JavaScript παρ' όλο που έχουν αρκετά πλεονεκτήματα ως προς την δημιουργία responsive εφαρμογών διαδικτύου αποτελούν σημεία εκμετάλλευσης από κακόβουλους χρήστες. Ο επιτιθέμενος δημιουργεί κακόβουλα προγράμματα JavaScript, μέσω των οποίων στόχος είναι η υποκλοπή ευαίσθητων και ιδιωτικών πληροφοριών του χρήστη με σκοπό την διάρρευσή τους, επιθέσεις εναντίον κοινωνικών ιστοσελίδων όπως το Facebook, την προσθήκη του στόχου χρήστη σε botnet και άλλα. Τα κακόβουλα προγράμματα JavaScript χρησιμοποιούνται ως το πρώτο βήμα για την περαιτέρω εκμετάλλευση του χρήστη. Δηλαδή, μέσω αυτών καθίσταται δυνατή η εγκατάσταση malware στο μηχάνημα στόχο.

Η πιο συνηθισμένη μορφή επίθεσης είναι η drive-by-download επίθεση. Σε αυτό το είδος ο επιτιθέμενος έχει ενσωματώσει κακόβουλα στοιχεία μέσα σε μια ιστοσελίδα η οποία, είτε είναι εξαρχής κακόβουλη και ο χρήστης έχει ανακατευθυνθεί σε αυτή μέσω κάποιας legimitate, είτε αυτά τα στοιχεία έχουν ενσωματωθεί κατευθείαν σε legimitate ιστοσελίδα του διαδικτύου. Εν συνεχεία ακολουθεί εκμετάλλευση (exploitation) των ευπαθειών του φυλλομετρητή ή των plugins του, κατεβαίνει ένα κακόβουλο αρχείο στο μηχάνημα του χρήστη που εκτελείται εν αγνοία του και στο οποίο περιλαμβάνεται κάποιο shellcode σε δεκαεξαδική κωδικοποίηση το οποίο συνήθως ξεκινάει μετά μια heap-spray επίθεση.

Κάποιες από τις τεχνικές μέσω των οποίων γίνεται το drive-by-download είναι οι xss (cross site scripting) επιθέσεις, οι csrf (cross site request forgery) επιθέσεις, μέσω malware διαφημίσεων (malvertisements), μέσω κακόβουλων ανακατευθύνσεων, spam μηνυμάτων, αιτήσεων για την εγκατάσταση πλαστών codecs κ.α.

Τα κακόβουλα στοιχεία που ενσωματώνονται στις ιστοσελίδες ώστε να ξεκινήσει η διαδικασία της επίθεσης είναι συνήθως κάποια iframes.

Τα iframes στοιχεία χρησιμοποιούνται για την ενσωμάτωση μιας ιστοσελίδας μέσα σε κάποια άλλη. Οπότε οι επιτιθέμενοι δημιουργούν μέσω του DOM, αντικείμενα τύπου iframe, τα οποία προσαρτώνται στην δεντρική δομή του παρόντος εγγράφου με την ιδιότητα hidden, ώστε να μην γίνονται αντιληπτά από τους χρήστες και καλούν κάποια κακόβουλη εξωτερική πηγή μέσω της οποίας θα κατέβει ο μολυσμένος κώδικας. Ένα παράδειγμα ακολουθεί παρακάτω:

```
function MakeFrameEx(){
  element = document.getElementById('yahoo_api');
  if (!element){
    var el = document.createElement('iframe');
    document.body.appendChild(el);
    el.id = 'yahoo_api';
    el.style.width = '1px';
    el.style.height = '1px';
    el.style.display = 'none';
```

```
        el.src = 'hxxp://juyfdjhdjdgh.nl.ai/showthread.php?t=72241732'
       }
      }
      var ua = navigator.userAgent.toLowerCase();
      if (((ua.indexOf("msie") !=- 1 && ua.indexOf("opera") ==- 1 && ua.indexOf("webtv") ==- 1))
       && ua.indexOf("windows") !=- 1){
       var t = setTimeout("MakeFrameEx()", 1000)
}
```

Παράδειγμα κακόβουλου iframe [15]

Όπως φαίνεται από τον παραπάνω κώδικα αρχικά γίνεται ανίχνευση των ιδιοτήτων του φυλλομετρητή και στη συνέχεια καλείται η συνάρτηση MakeFrameEX( ), η οποία δημιουργεί ένα iframe, το οποίο προσαρτάται στον κορμό του εγγράφου με συγκεκριμένες ιδιότητες και αντιστοιχεί σε κακόβουλη τοποθεσία. Άλλα παραδείγματα τέτοιου τύπου παρατίθενται παρακάτω:

```
    if (document.getElementsByTagName('body')[0]){   iframer();
    }
    else {
     document.write("
    &lt;iframe src='hxxp:// orjbhasqs . co . be/forum.php?tp=4a4e85585df3e7c1' width='10' height='10'
    style='visibility:hidden;position:absolute;left:0;top:0;'&gt;&lt;/iframe&gt;");
    }
    function iframer(){
     var f = document.createElement('iframe');
     f.setAttribute('src', 'hxxp://orjbhasqs.co.be/forum.php?tp=4a4e85585df3e7c1');
     f.style.visibility = 'hidden';
     f.style.position = 'absolute';
     f.style.left = '0';
     f.style.top = '0';
     f.setAttribute('width', '10');
     f.setAttribute('height', '10');
     document.getElementsByTagName('body')[0].appendChild(f);
}
```

Παράδειγμα κακόβουλου iframe [15]

Για τον εντοπισμό κακόβουλης JavaScript οι συνήθεις τακτικές που χρησιμοποιούνται αφορούν την χρήση εργαλείων τα οποία κάνουν είτε στατική είτε δυναμική ανάλυση.

Στην δυναμική ανάλυση το εκάστοτε εργαλείο αναλύει την ύποπτη ιστοσελίδα, όπως ακριβώς θα έκανε ένας φυλλομετρητής. Μέσω αυτής της δαδικασίας παρατηρείται τυχόν αλλαγή στο μηχάνημα του χρήστη και αποφασίζει αν υπάρχει κάποια κακόβουλη δραστηριότητα. Το μειονέκτημα της

δυναμικής ανάλυσης είναι ότι όσο χρονικό διάστημα γίνεται η διαδικασία το σύστημα είναι ευπαθές σε επιθέσεις. Κάποια εργαλεία δυναμικής ανάλυσης κακόβουλης JavaScript είναι τα Wepawet (isecLAB), MonkeyWench(G Data Software AG)

Στην αντίθετη πλευρά υπάρχει η στατική ανάλυση, στην οποία ο κώδικας της εξεταζόμενης ιστοσελίδας κατεβαίνει και ο έλεγχος απαιτεί ανθρώπινη παρέμβαση ώστε να ελεγχθεί χειροκίνητα. Σε αυτή την περίπτωση η ανάλυση παίρνει αρκετή ώρα για να πραγματοποιηθεί.

Άλλη τακτική για τον εντοπισμό κακόβουλης JavaScript είναι η χρήση εργαλείων τα οποία βασίζονται σε μηχανική μάθηση όπως το Cujo, Zozzle και IceShield, καθώς και high-interaction και low-interaction honeypots.

3.1 Τεχνικές Obfuscation

Ορίζεται η τεχνική, η οποία μετατρέπει τον κώδικα σε διαφορετική μορφή και είναι σημασιολογικά ισότιμη με το αρχικό πρόγραμμα, αλλά δεν μπορεί να διαβαστεί από το ανθρώπινο μάτι.

Αυτή την τεχνική την χρησιμοποιούν οι επιτιθέμενοι έτσι ώστε να καλύψουν τον κακόβουλο κώδικα για να μην γίνεται ανιχνεύσιμος από antivirus και άλλα εργαλεία ανίχνευσης κακόβουλης JavaScript.

Όμως το ότι ένας κώδικας είναι σε αυτή τη μορφή δεν σημαίνει απαραίτητα ότι εμπεριέχει κάποιο κίνδυνο. Χρησιμοποιείται και από προγραμματιστές, οι οποίοι θέλουν να αποτρέψουν την πνευματική υποκλοπή. Διάφορα εργαλεία τα οποία έχουν αναπτυχθεί για την ανάλυση obfuscated JavaScript κώδικα στην αρχική του μορφή είναι το Caffeine Monkey (Mozilla Spider Monkey), το Wepawet, Jsunpack (iDefense), The Ultimate Deobfuscator (WebSense).

Σε αυτό το μέρος της εργασίας θα αναλύσουμε τις κατηγορίες με τις οποίες μπορεί ένας κώδικας να είναι obfuscated με τα κατάλληλα παραδείγματα.

1) Τεχνική Obfuscation με τυχαιότητα (Randomization Obfuscation) [2], [4]

Σε αυτή την κατηγορία ο αρχικός κώδικας μετατρέπεται σε obfuscated με την αντικατάσταση των ονομάτων μεταβλητών ή συναρτήσεων από τυχαίους χαρακτήρες. Επίσης, προστίθενται σχόλια σε διάσπαρτα σημεία, χαρακτήρες όπως tab, space, linefeed, formfeed, carriage return (whitespace characters). Όλη η παραπάνω διαδικασία δεν αλλάζει τη σημασιολογία του αρχικού κώδικα. Ένα παράδειγμα ακολουθεί παρακάτω:

```
<html>
  <head>
  <script type="text/javascript">
      function sayHello(name, age)
      {
        alert (name + " is " + age + " years old.");

      }
                var string1="name1";
                var string2="25";
                sayHello(string1,string2);
```

```
        </script>
            </head>
            <body>

            </body>
            </html>
```

```
    <html>
      <head>
      <script type="text/javascript">
      function
      sayHello                (_0x98dbx1,_0x98dbx2)
      {
      //_sjh3gh4mbwkjeh
      alert(_0x98dbx1+ " is "+ _0x98dbx2+ " years old.");}var string1="name1";var
string2="25";sayHello(string1,string2)

        </script>
            </head>
            <body>

            </body>
            </html>
```

```
    <html>
      <head>
      <script type="text/javascript">
                    var _$_8759=[" is "," years old.","name1","25"];
    function sayHello(_0x42F5,_0x42A3)
    {
          alert(_0x42F5+ _$_8759[0]+ _0x42A3+ _$_8759[1]);
    }
    var string1=_$_8759[2];//5
    var string2=_$_8759[3];//6
    sayHello(string1,string2)
        </script>
            </head>
            <body>
```

```
        </body>
        </html>
```

2) Τεχνική obfuscation σε δεδομένα (Data Obfuscation) [2], [4]

Με αυτόν τον τρόπο μετατρέπεται μια μεταβλητή ή μια σταθερά σε ένα σύνολο μεταβλητών ή σταθερών, έτσι ώστε με την χρήση κατάλληλων συναρτήσεων να παράγουν κατά τον χρόνο εκτέλεσης το ίδιο αποτέλεσμα. Όταν εφαρμόζεται σε αντικείμενα τύπου string, γίνεται κατακερματισμός του αλφαριθμητικού ή των λέξεων κλειδιών σε μικρότερα κομμάτια (substrings), τα οποία αποθηκεύονται σε μεταβλητές με τυχαία ονόματα και τα οποία εκτελούνται με συναρτήσεις όπως, η eval( ) ή η document.write( ). Μπορεί όμως και δεδομένα όπως αριθμό, να παράγονται μέσω ισοδύναμων εκφράσεων με την χρήση πάλι συναρτήσεων όπως η eval(), ώστε να αυξηθεί η πολυπλοκότητα του κώδικα.

Παραδείγματα που παρουσιάζουν και τις δύο περιπτώσεις φαίνονται παρακάτω:

```
<html>
  <head>
  <script type="text/javascript">
      function sayHello(name, age)
      {
        alert (name + " is " + age + " years old.");

      }
                var string1="name1";
                var string2="25";
                sayHello(string1,string2);

    </script>
      </head>
      <body>

      </body>
      </html>
```

```
<html>
  <head>
<script type="text/javascript">
  eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};if(!''.replace(/^/,String)){while(c--){d
  [c.toString(a)]=k[c]||c.toString(a)}k=[function(e){return d[e]}];e=function()
  {return'\\w+'};
  c=1};while(c--){if(k[c])
  {p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}
  return p}('9 a 8 c 4 5 6.{7(b+" m "+j+" d l.")}0 1="i";0 3="h";
  e(1,3);</f></g><2></2></k>',23,23,'var|string1|body|string2|jQuery|code|here|alert|JavaScript|
  Paste|your|name|or|years|sayHello|script|head|25|name1
  |age|html|old|is'.split('|'),0,{}))    </script>
        </head>
        <body>

        </body>
        </html>
```

3) Τεχνική obfuscation με κωδικοποίηση (Encoding Obfuscation) [2], [4]

Σε αυτήν την περίπτωση ο κώδικας JavaScript μετατρέπεται σε κωδικοποιημένη μορφή χρησιμοποιώντας χαρακτήρες ASCII, Unicode ή δεκαεξαδική κωδικοποίηση. Ο κώδικας, είτε κωδικοποιείται εξαρχής με αυτόν τον τρόπο είτε δημιουργείται ειδική συνάρτηση κωδικοποίησης των χαρακτήρων και στο τελικό αποτέλεσμα υπάρχει και η αντίστοιχη συνάρτηση αποκωδικοποίησης στον obfuscated κώδικα που τον αποκωδικοποιεί κατά τον χρόνο εκτέλεσης. Μια από τις βασικές συναρτήσεις που συναντώνται σε αυτήν την περίπτωση είναι η charCodeAt(), η οποία επιστρέφει την Unicode τιμή ενός χαρακτήρα.

```
<html>
  <head>
  <script type="text/javascript">
      function sayHello(name, age)
      {
        alert (name + " is " + age + " years old.");

      }
              var string1="name1";
              var string2="25";
              sayHello(string1,string2);

   </script>
```

```
        </head>
        <body>

        </body>
        </html>
```

```
    <html>
      <head>
        <script type="text/javascript">
var
_0x879e=["\x20\x69\x73\x20","\x20\x79\x65\x61\x72\x73\x20\x6F\x6C\x64\x2E","\x6E\x61\x6D\x65\x31","\x32\x35"];
function sayHello(_0x30bcx2,_0x30bcx3){alert(_0x30bcx2+ _0x879e[0]+ _0x30bcx3+ _0x879e[1]);}
var string1=_0x879e[2];var string2=_0x879e[3];sayHello(string1,string2)

        </script>
        </head>
        <body>

        </body>
        </html>
```

4) Τεχνική obfuscation χρησιμοποιώντας τις λογικές δομές (Logic Structure Obfuscation) [2], [4]

Όταν εισάγονται λογικές δομές, όπως συνθήκες if/else, for, while,switch/case και διάφορες άλλες μη σχετικές με την λογική ροή του προγράμματος εντολές, οι οποίες δεν αλλάζουν το τελικό αποτέλεσμα του κώδικα.

```
    <html>
      <head>
      <script type="text/javascript">
          function sayHello(name, age)
          {
            alert (name + " is " + age + " years old.");
          }
                    var string1="name1";
                    var string2="25";
                    sayHello(string1,string2);

        </script>
          </head>
```

```
        <body>

        </body>
        </html>
```

```
<html>
  <head>
  <script type="text/javascript">
      function sayHello(name, age)
      {
var i=111;
i=i+1;
if(i>10){
       alert (name + " is " + age + " years old.");
}
i=i+1;
      }
                var string1="name1";
                var string2="25";
                sayHello(string1,string2);

    </script>
      </head>
      <body>

      </body>
      </html>
```

5) Τεχνική obfuscation με χρήση xor λειτουργίας [2], [4]

Με την κατασκευή μιας συνάρτησης xor, γίνεται η xor λειτουργία σε αλφαριθμητικά με ένα προκαθορισμένο κλειδί (κρυπτογράφηση) για την δημιουργία obfuscated κώδικα. Φυσικά, πρέπει να περιλαμβάνεται και η αντίστοιχη συνάρτηση αποκρυπτογράφησης για να εκτελεστεί σωστά ο κώδικας.

Ένα παράδειγμα μιας τέτοιας συνάρτησης κρυπτογράφησης xor φαίνεται παρακάτω:

```
function xorEncode(txt, pass) {

  var ord = []
  var buf = ""

  for (z = 1; z <= 255; z++) {ord[String.fromCharCode(z)] = z}
```

```
        for (j = z = 0; z < txt.length; z++) {
            buf += String.fromCharCode(ord[txt.substr(z, 1)] ^ ord[pass.substr(j, 1)])
            j = (j < pass.length) ? j + 1 : 0
        }

        return buf

}
```

Κεφάλαιο 4ο

4. Δημιουργία επέκτασης Chrome

Σε αυτό το μέρος θα περιγραφεί ο τρόπος που στήθηκε η εικονική μηχανή που περιέχει τα δύο βασικά εργαλεία για την ανίχνευση κακόβουλου κώδικα, καθώς και οι τροποποιήσεις που χρειάστηκαν να γίνουν τόσο σε επίπεδο δικτύου όσο και στα ίδια τα εργαλεία έτσι ώστε να μπορούν να επικοινωνούν με την επέκταση που υλοποιήθηκε.

4.1 Παραμετροποίηση εικονικής μηχανής και εγκατάσταση των εργαλείων JSDetox και jsunpack-n

Το πρώτο βήμα είναι η εγκατάσταση του επιθυμητού λειτουργικού συστήματος που θα φιλοξενήσει τα εργαλεία. Επιλέχθηκε το λειτουργικό σύστημα Linux Ubuntu με την έκδοση 14.04.5 LTS. Τα δύο εργαλεία που θα εγκατασταθούν στην εικονική μηχανή είναι τα JSDetox και jsunpack-n.

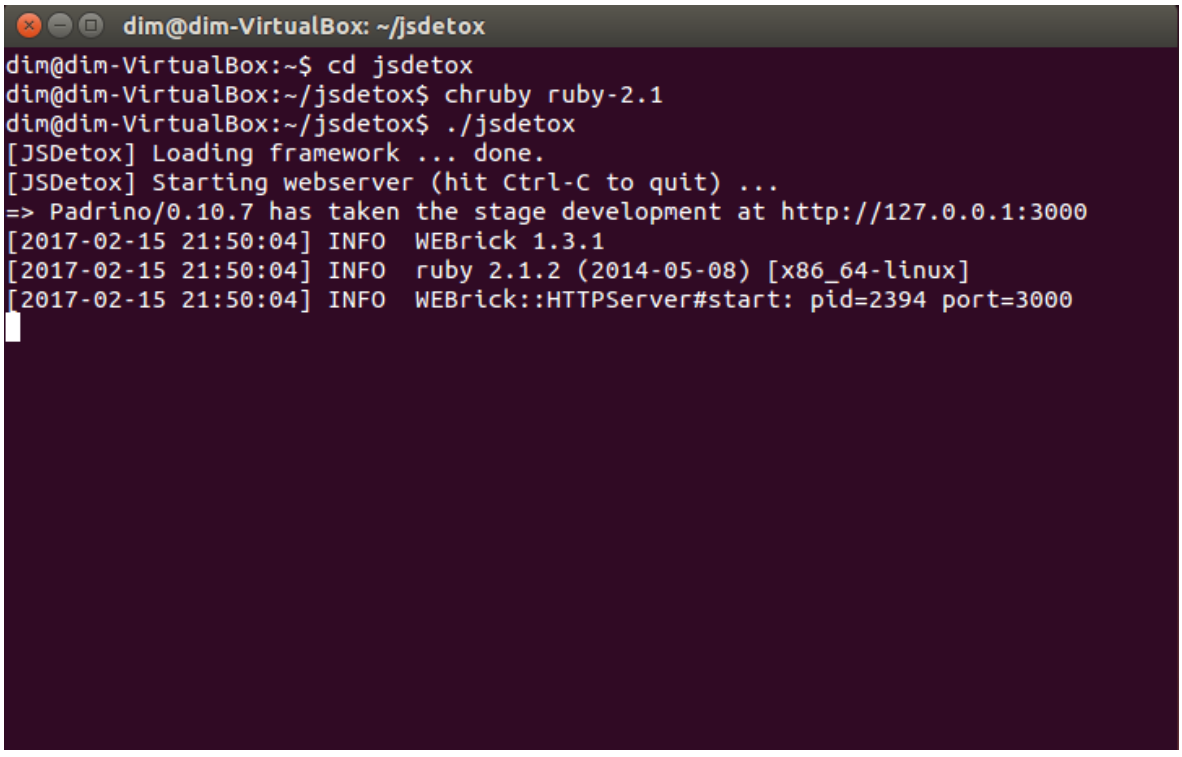
Το JSDetox [12] πραγματοποιεί στατική ανάλυση της εξεταζόμενης JavaScript και διαθέτει τεχνικές de-obfuscating, καθώς και μηχανή εκτέλεσης με χαρακτηριστικά HTML DOM προσομοίωσης.

Έχει υλοποιηθεί σε ruby με το framework Ruby on Rails και λειτουργεί στις περισσότερες Linux διανομές. Η διαδικασία εγκατάστασης του εργαλείου ακολουθεί ως εξής:

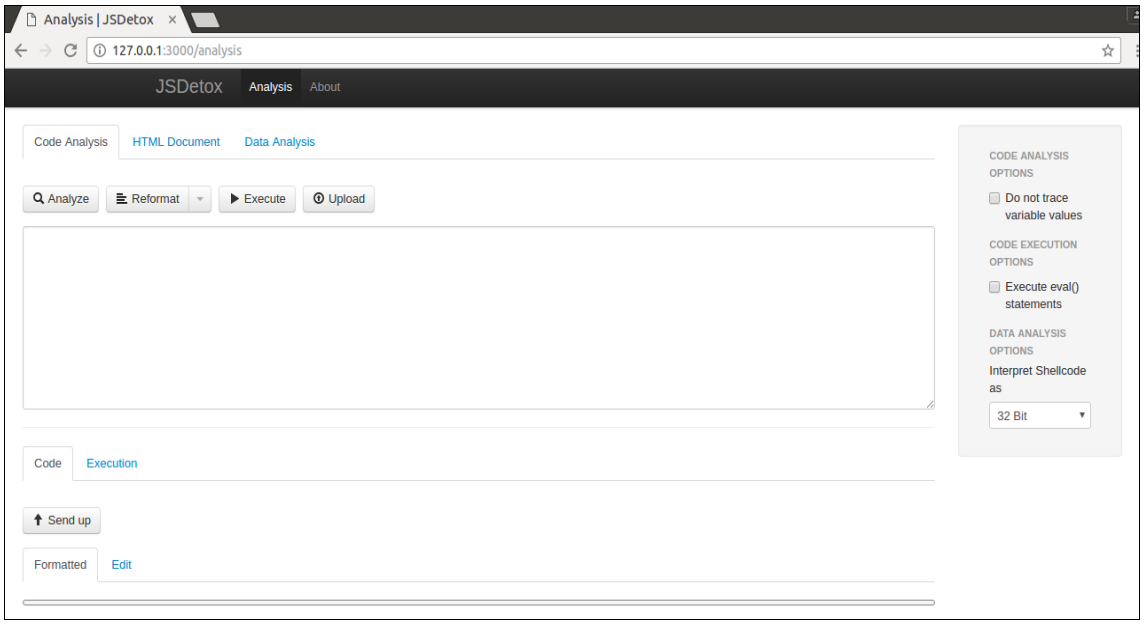
```
# install ruby-install, see https://github.com/postmodern/ruby-install
wget -O ruby-install-0.4.3.tar.gz https://github.com/postmodern/ruby-install/archive/v0.4.3.tar.gz
tar -xzvf ruby-install-0.4.3.tar.gz
cd ruby-install-0.4.3/
sudo make install

# install ruby
ruby-install ruby 2.1

# install chruby, see https://github.com/postmodern/chruby#install
wget -O chruby-0.3.8.tar.gz https://github.com/postmodern/chruby/archive/v0.3.8.tar.gz
tar -xzvf chruby-0.3.8.tar.gz
cd chruby-0.3.8/
sudo make install

# load chruby by default
echo "source /usr/local/share/chruby/chruby.sh" >>~/.bashrc

# load chruby for the current session
source /usr/local/share/chruby/chruby.sh
```

```
# enable ruby 2.1
# needed every time when running JSDetox unless you enable auto switching:
# https://github.com/postmodern/chruby#auto-switching
chruby ruby-2.1

# install jsdetox
gem install bundler
cd #target-dir#
git clone https://github.com/svent/jsdetox.git
cd jsdetox
bundle install
./jsdetox
```

Μετά την επιτυχή εγκατάσταση του JSDetox ξεκινάει ο web server στην ip διεύθυνση http://127.0.0.1:3000, όπως φαίνεται και στις παρακάτω εικόνες:



Εικόνα 4. Εκκίνηση διεργασίας του JSDetox



Εικόνα 5. Περιβάλλον του JSDetox στον browser

Στο συγκεκριμένο εργαλείο έγινε τροποποίηση στον κώδικα των controllers, καθώς το framework βασίζεται σε αρχιτεκτονική MVC, έτσι ώστε να χρησιμοποιηθούν συγκεκριμένες λειτουργίες. Πιο συγκεκριμένα όταν πραγματοποιείται αίτηση AJAX από το extension αρχικά γίνεται εξαγωγή των JavaScript κομματιών κώδικα από τον κώδικα πηγής της εξεταζόμενης σελίδας. Αυτό γίνεται με την χρήση του extract_script_tags του JSDetox. Στις παρακάτω εικόνες παρουσιάζεται η αλλαγή στον κώδικα του backend.rb με path /jsdetox/app/controllers/backend.rb.

```
post :extract_script_tags, :provides => :json do
  begin
    html = params[:htmldoc]
    if html
      doc = Taka::DOM::HTML(html)
      script_tags = doc.getElementsByTagName('script').map(&:innerHTML)
      {
        :status => :ok,
        :tag_count => script_tags.length,
        :tags => script_tags,
      }.to_json
    else
      {
        :status => :ok,
        :tag_count => 0,
      }.to_json
    end
  rescue
    return { :status => :error }.to_json
  end
end

end
```

Εικόνα 6. Αρχείο controller του JSDetox και τροποποίηση στο action method extract_script_tags

Μια άλλη βασική λειτουργία του εργαλείου είναι και η συνάρτηση execute. Η συγκεκριμένη συνάρτηση εκτελεί και αξιολογεί τον κώδικα που είναι obfuscated και περιέχει συναρτήσεις δυναμικής παραγωγής (D-gen, Dynamic generation functions) και συναρτήσεις εκτίμησης κατά τον χρόνο εκτέλεσης (runtime evaluation functions -R-Gen) όπως η eval (). Συνεπώς, με την κλήση της execute πραγματοποιείται de-obfuscation, το αποτέλεσμα της οποίας στην συνέχεια προωθείται στο εργαλείο jsunpack-n.

```
post :execute, :provides => :json do
  if params[:code]
    session[:orig_code] = params[:code]
    htmldoc = session[:htmldoc]
    opts = params[:opts] || {}

    engine = JSDetox::JSEngines::V8Engine::Instance.new
    res = engine.execute(params[:code], opts, htmldoc)
    res.to_json
  end
end
```

Εικόνα 7. Αρχείο controller του JSDetox και τροποποίηση στο action method execute

Μια βασική αλλαγή που έγινε στο εργαλείο, αφορά την εγκατάσταση του rack CORS gem και την τροποποίηση του αρχείου /jsdetox/app/app.rb. Το συγκεκριμένο gem επιτρέπει το Cross -Origin Resource Sharing. Όταν γίνονται κλήσεις AJAX επιτρέπονται μόνο όταν είναι στο ίδιο origin για λόγους ασφάλειας, έτσι ώστε κάποιο κακόβουλο script να μην έχει πρόσβαση σε ευαίσθητες πληροφορίες σε άλλη σελίδα. Δηλαδή τα scripts μιας σελίδας μπορούν να εκτελεστούν και να έχουν πρόσβαση στα δεδομένα μιας άλλης σελίδας μόνο όταν βρίσκονται στο ίδιο origin.

Η διαδικασία εγκατάστασης εφ' όσον έχει εγκατασταθεί το rubygems (package manager για την εγκατάσταση των gems) είναι η εξής:

```
gem install rack-cors
```

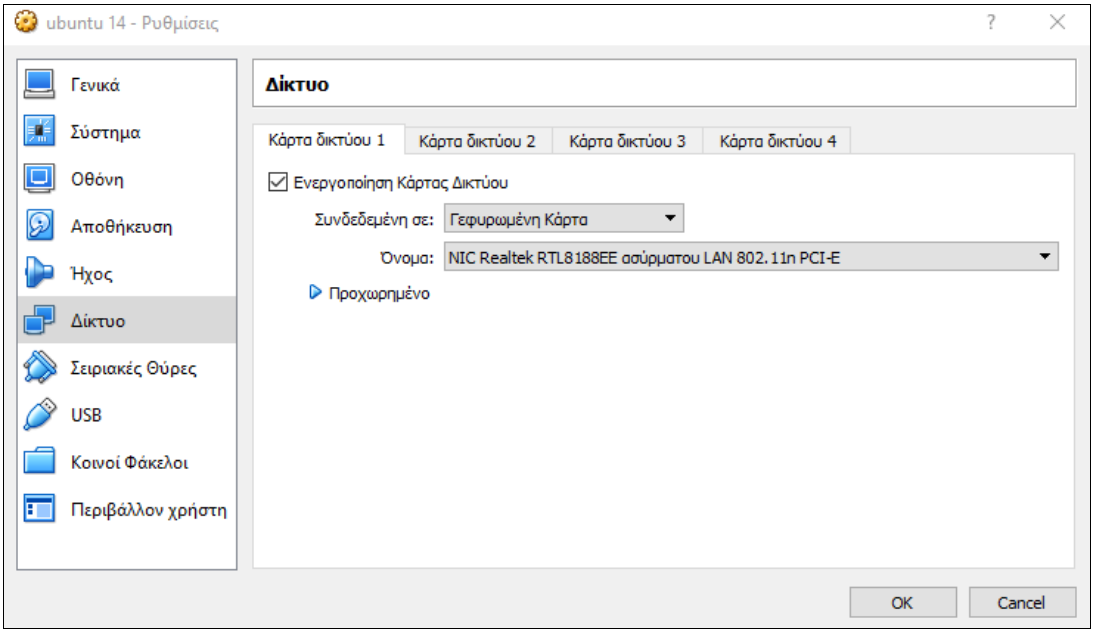
Στην συνέχεια κάνουμε το κατάλληλο configuration στο αρχείο app.rb, όπως φαίνεται και στις παρακάτω εικόνες.

```
class JSDetoxWeb < Padrino::Application
        register SassInitializer
        register Padrino::Rendering
        register Padrino::Mailer
        register Padrino::Helpers
        register Padrino::Cache

        use Rack::Session::Pool

        use Rack::Cors do
  allow do
    # put real origins here
    origins '*'
    # and configure real resources here
    resource '*', :headers => :any, :methods => [:get, :post, :options]
  end
end

        get "/analysis/" do
                redirect url(:analysis, :index)
        end

end
```

Εικόνα8. Διαδικασία ενεργοποίησης Cross-Origin Resource Sharing

Για τις ανάγκες της εργασίας στο Virtual Machine που περιέχει τα εργαλεία που αλληλοεπιδρούν με την επέκταση του Chrome, έπρεπε να ανατεθεί σταθερή ip στην οποία θα γίνονται οι κλήσεις AJAX.

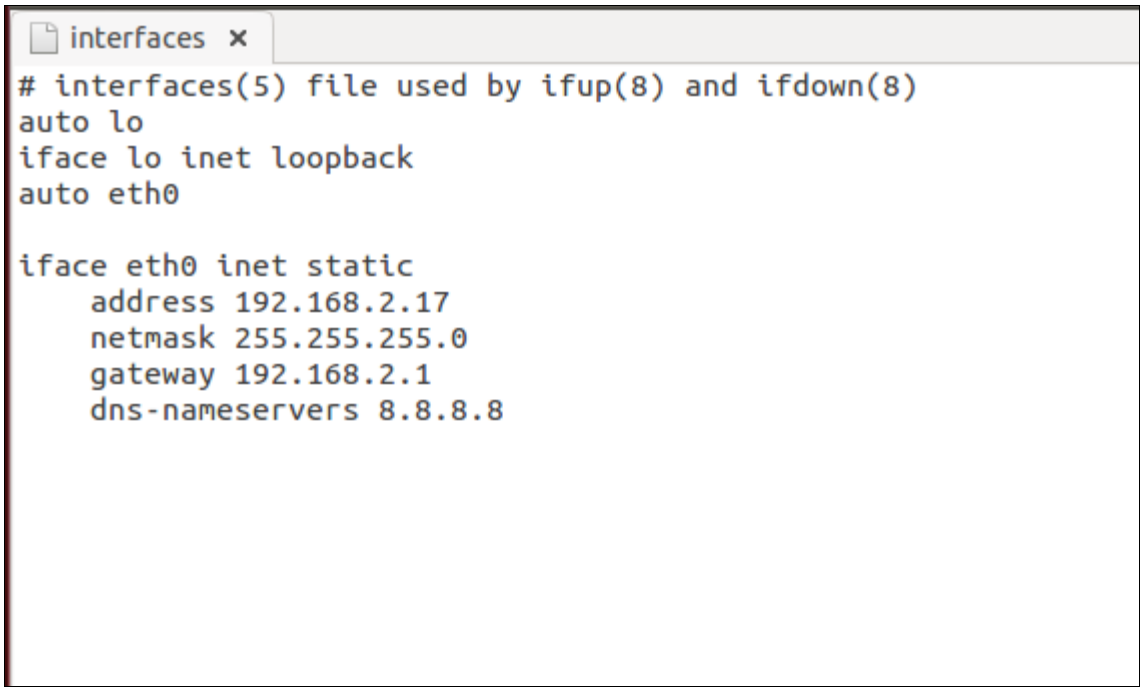
Το πρώτο βήμα είναι η αλλαγή στις ρυθμίσεις της ίδιας της εικονικής μηχανής, ώστε η δικτύωση να είναι σε επιλογή bridged για να πάρει ip από την ίδια την κάρτα του υπολογιστή.



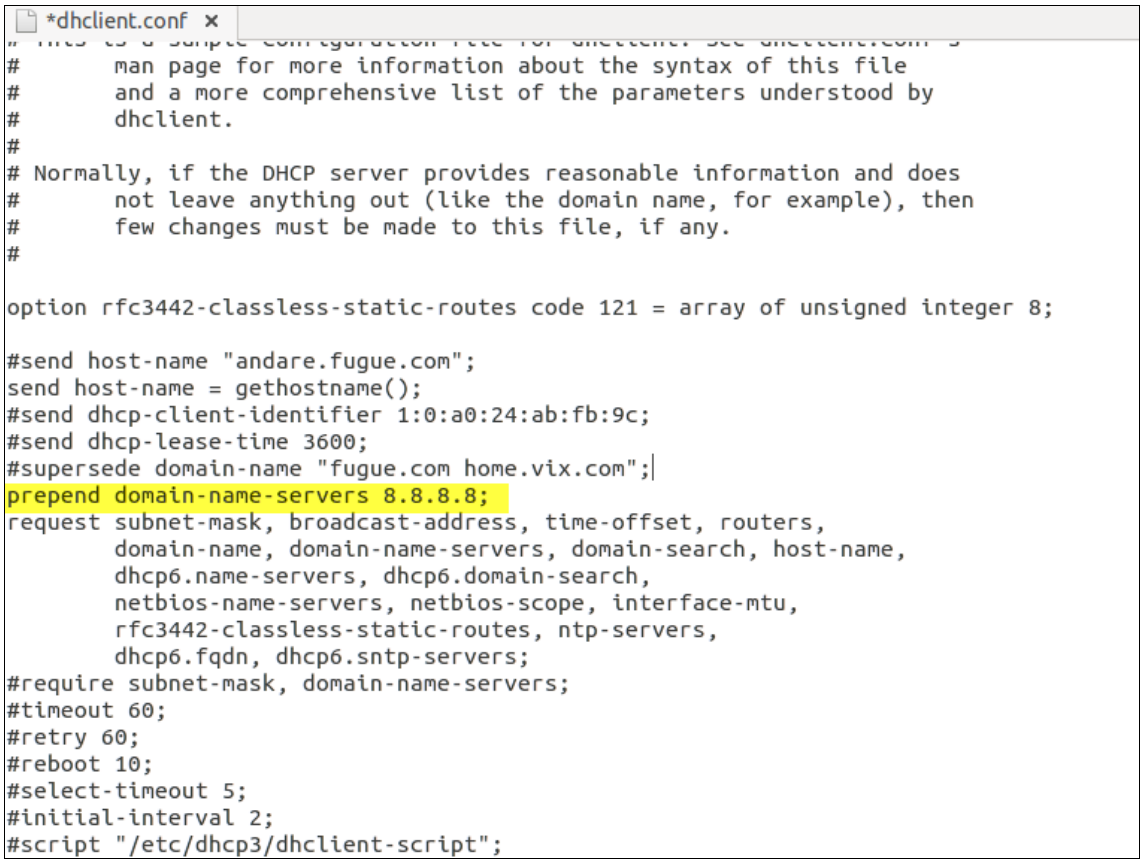
Εικόνα 9. Ρυθμίσεις δικτύου εικονικής μηχανής

Στην συνέχεια το configuration πραγματοποιείται και στο λειτουργικό σε δύο αρχεία στο /etc/network/interfaces και στο etc/dhcp/dhclient.conf.

Στο πρώτο αρχείο ορίζουμε ένα καινούριο interface στο οποίο αναθέτουμε την επιθυμητή διεύθυνση και στο δεύτερο ορίζουμε τους DNS servers στην περίπτωση που θέλουμε η εικονική μηχανή μας να έχει πρόσβαση στο internet. Οι αλλαγές που πραγματοποιούνται φαίνονται στις παρακάτω εικόνες:



Εικόνα 10. Τροποποίηση του αρχείου interfaces για την απόδοση σταθερής ip



Εικόνα 11. Τροποποίηση του αρχείου dhclient.conf για την προσθήκη dns servers

Συνεπώς με την εντολή ifconfig στο τερματικό βλέπουμε ότι δημιουργήθηκε το interface eth0 με ip 192.168.2.17.

Θα πρέπει να σημειωθεί ότι οι τροποποιήσεις θα πρέπει να γίνουν σύμφωνα με το δίκτυο στο οποίο βρίσκεται ο host υπολογιστής.

Μετά την ανάθεση σταθερής δεύθυνσης ip στην εικονική μηχανή το επόμενο βήμα είναι η δημιουργία ενός proxy server ο οποίος θα ανακατευθύνει την κίνηση στην διεύθυνση που ακούει το JSDetox, δηλαδή στην διεύθυνση http://127.0.0.1:3000, έτσι ώστε να γίνονται οι κλήσεις AJAX από το front end. Το πρώτο βήμα για την επίτευξη αυτού του στόχου είναι η εγκατάσταση ενός server στην εικονική μηχανή. Για τις ανάγκες της εργασίας χρησιμοποιήθηκε η σουίτα xampp για Linux.

Για την δημιουργία του proxy server θα πρέπει να τροποποιηθεί το αρχείο /opt/lamp/etc/httpd.conf.

Τα απαραίτητα modules που πρέπει να ενεργοποιηθούν είναι τα εξής:

```
Proxy_module modules/mod_proxy.so
Proxy_balancer_module modules/mod_proxy_balancer.so
Proxy_ftp_module modules/mod_proxy_http.so
Proxy_ajp_module modules/mod_proxy_ajp.so
Proxy_connect_module modules/mod_proxy_connect.so
```

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_express_module modules/mod_proxy_express.so
```

Στο τέλος του αρχείου έχουν προστεθεί οι κανόνες για την ανακατεύθυνση στο JSDetox, όπως φαίνεται στην παρακάτω εικόνα:

```
<IfModule mod_proxy.c>
#
# Reverse Proxy
#
ProxyRequests On
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>
ProxyVia On
ProxyPass /analysis/ http://127.0.0.1:3000/
ProxyPassReverse /analysis/ http://127.0.0.1:3000/
ProxyPreserveHost On
</IfModule>
```

Σύμφωνα με το παραπάνω όταν πληκτρολογείται το 192.168.2.17/analysis/ γίνεται ανακατεύθυνση στην διεύθυνση http://127.0.0.1:3000.

Συνεπώς, η επέκταση είναι έτοιμη για επικοινωνία με το JSDetox. Με την κλήση AJAX στην διεύθυνση 192.168.2.17/analysis/ η επέκταση στέλνει τα εξεταζόμενα κομμάτια στο εργαλείο εξάγει τον κώδικα javascript, και τον φιλτράρει από την συνάρτηση check_obfuscation για να διαπιστωθεί αν είναι obfuscated ή όχι.

Αν η συνάρτηση παράγει τιμή μεγαλύτερη από το κατώφλι τότε ο κώδικας είναι obfuscated και ξαναστέλνεται στο JSDetox, όπου με την χρήση της λειτουργίας execute επιστρέφει de-obfuscated και αποστέλλεται στο επόμενο εργαλείο το οποίο θα αποφανθεί αν υπάρχει κακόβουλη δραστηριότητα ή όχι. Εδώ θα πρέπει να σημειωθεί ότι μετά την πρώτη χρήση της λειτουργίας execute το επιστρεφόμενο κομμάτι ελέγχεται ξανά από την συνάρτηση check_obfuscation, διότι υπάρχει περίπτωση να έχει εφαρμοστεί διπλό ή τριπλό επίπεδο obfuscation.

Συνεχίζοντας στο επόμενο εργαλείο jsunpack-n [13], το οποίο πραγματοποιεί δυναμική ανάλυση προσομοιώνοντας την λειτουργία ενός φυλλομετρητή όπως όταν ένας χρήστης επισκέπτεται κάποια

σελίδα στο διαδίκτυο. Ο σκοπός τους είναι να εντοπίσει exploits τα οποία στοχεύουν σε ευπάθειες των φυλλομετρητών και σε plug-ins όπως το Active x.

Το συγκεκριμένο εργαλείο αφού κατεβάσουμε το αρχείο από το github εγκαθίσταται ως εξής:

```
cd depends
  $ tar xvfz pynids-0.6.1.tar.gz
    $ cd pynids-0.6.1/ directory
    $ python setup.py build
    $ sudo python setup.py install
```

Επόμενο βήμα είναι η εγκατάσταση επιπλέον εργαλείων τα οποία συνεργάζονται με το jsunpack-n.

Το πρώτο από αυτά είναι το SpiderMonkey, το οποίο είναι μια μηχανή JavaScript του φυλλομετρητή Mozilla που έχει αναπτυχθεί σε γλώσσα C/C++. Η εγκατάστασή του γίνεται ως εξής:

```
cd depends/
  $ tar xvfz js-1.8.0-rc1-src.tar.gz
  $ cd js-1.8.0-rc1-src
  $ make BUILD_OPT=1 -f Makefile.ref
  Then, make the 'js' binary available within your path.
$ echo "export $PATH="
```

Συνεχίζοντας στο επόμενο εργαλείο yara το οποίο ταυτοποιεί και κατηγοριοποιεί το κακόβουλο λογισμικό με την χρήση κανόνων.

Χρησιμοποιείται για την ταξινόμηση αρχείων και ενεργών διαδικασιών για τον καθορισμό της οικογένειας malware στην οποία ανήκουν. Η εγκατάστασή του φαίνεται παρακάτω:

```
cd depends
  $ tar xvfz yara-1.6.tar.gz
  $ cd yara-1.6
  $ ./configure
  $ make
$ sudo make install
```

```
sudo echo "/usr/local/lib" >> /etc/ld.so.conf
    $ sudo ldconfig
```

```
cd depends
  $ tar xvfz yara-python-1.6.tar.gz
  $ cd  yara-python-1.6
  $ python setup.py build
```

```
$ sudo python setup.py install
```

Παρακάτω παρατίθεται η εγκατάσταση του beautiful soup, το οποίο αποτελεί μια python βιβλιοθήκη ώστε να εξάγουμε δεδομένα από HTML και XML αρχεία καθώς παρουσιάζεται και η εγκατάσταση του pycrypto.

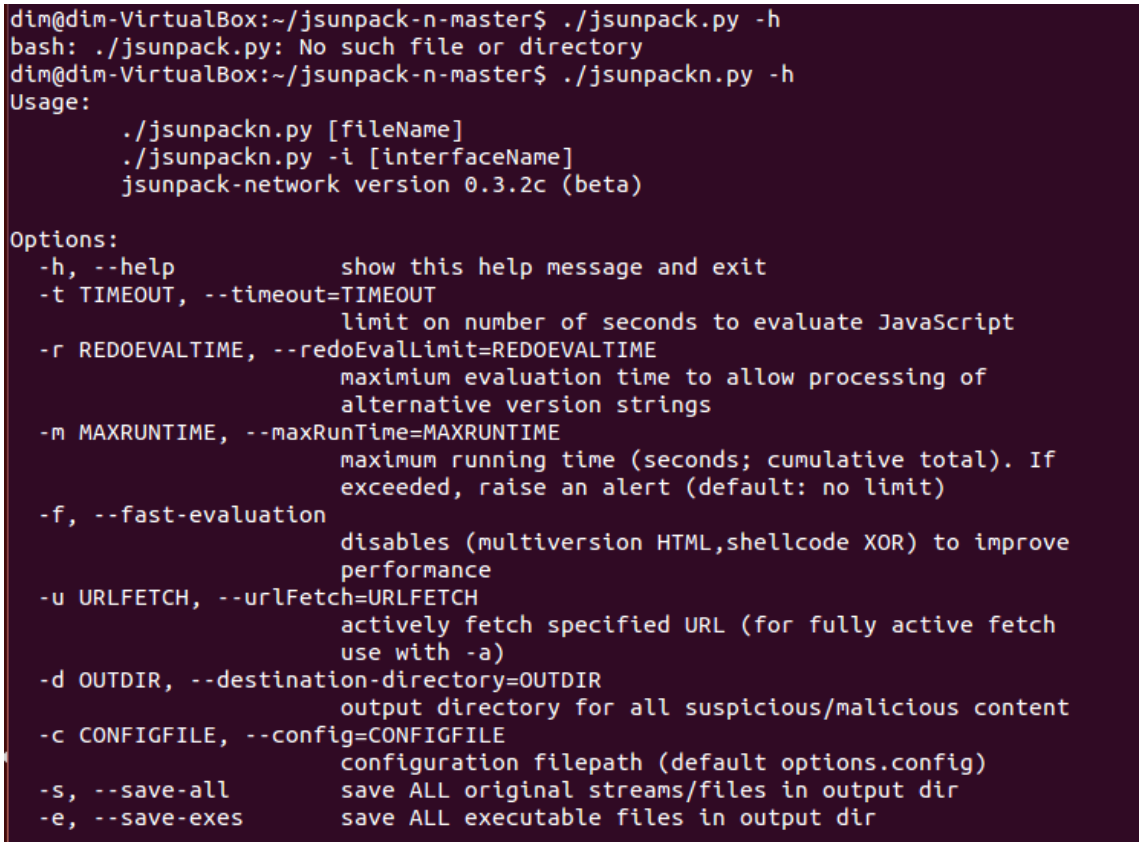
```
cd depends
    $ tar xvfz BeautifulSoup-3.2.0.tar.gz
    $ cd BeautifulSoup-3.2.0/
    $ python setup.py build
$ sudo python setup.py install
```

```
cd depends
    $ tar xvfz pycrypto-2.4.1.tar.gz
    $ cd pycrypto-2.4.1
    $ python setup.py build
$ sudo python setup.py install
```

Μετά την επιτυχή εγκατάσταση όλων των παραπάνω μεταβαίνοντας στον φάκελο του jsunpack-n μέσω του command line, μπορούμε να ελέγξουμε κάποιο αρχείο για την ύπαρξη κακόβουλης JavaScript ως εξής:

```
./jsunpackn.py -v test.js
```

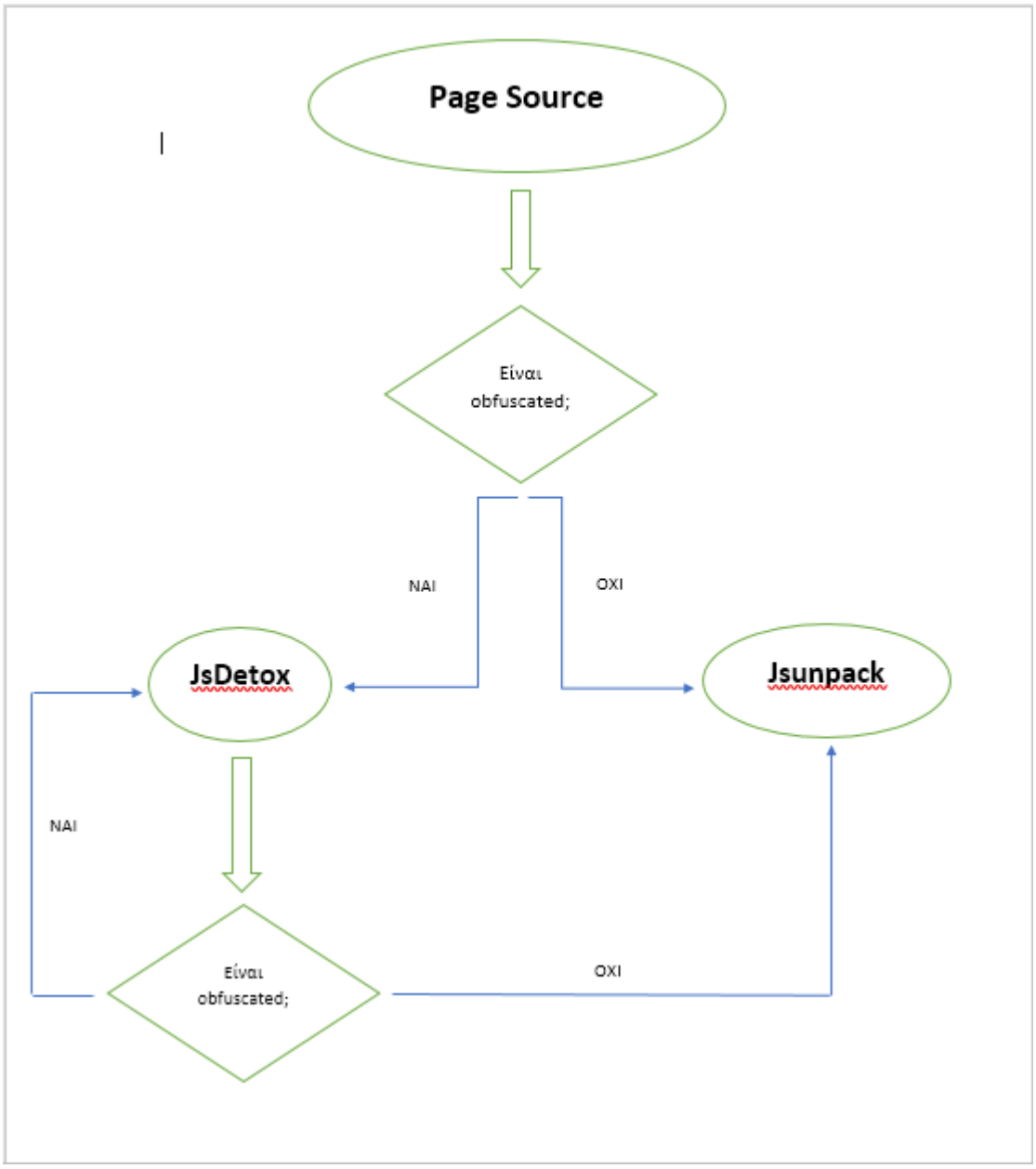
Για περισσότερες πληροφορίες σχετικά με την σύνταξη του jsunpack-n μπορούμε να πληκτρολογήσουμε την εντολή ./jsunpackn.py -h όπως φαίνεται και στην εικόνα παρακάτω:



```
dim@dim-VirtualBox:~/jsunpack-n-master$ ./jsunpack.py -h
bash: ./jsunpack.py: No such file or directory
dim@dim-VirtualBox:~/jsunpack-n-master$ ./jsunpackn.py -h
Usage:
        ./jsunpackn.py [fileName]
        ./jsunpackn.py -i [interfaceName]
        jsunpack-network version 0.3.2c (beta)

Options:
  -h, --help            show this help message and exit
  -t TIMEOUT, --timeout=TIMEOUT
                        limit on number of seconds to evaluate JavaScript
  -r REDOEVALTIME, --redoEvalLimit=REDOEVALTIME
                        maximium evaluation time to allow processing of
                        alternative version strings
  -m MAXRUNTIME, --maxRunTime=MAXRUNTIME
                        maximum running time (seconds; cumulative total). If
                        exceeded, raise an alert (default: no limit)
  -f, --fast-evaluation
                        disables (multiversion HTML,shellcode XOR) to improve
                        performance
  -u URLFETCH, --urlFetch=URLFETCH
                        actively fetch specified URL (for fully active fetch
                        use with -a)
  -d OUTDIR, --destination-directory=OUTDIR
                        output directory for all suspicious/malicious content
  -c CONFIGFILE, --config=CONFIGFILE
                        configuration filepath (default options.config)
  -s, --save-all        save ALL original streams/files in output dir
  -e, --save-exes       save ALL executable files in output dir
```

4.2 Περιγραφή λειτουργίας της επέκτασης Chrome

Αφού εγκαταστάθηκαν τα εργαλεία και έγινε η παραμετροποίηση του δικτύου και της εικονικής μηχανής η επέκταση chrome που αναπτύχθηκε ακολουθεί την διαδικασία όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 12. Διάγραμμα ροής λειτουργίας της επέκτασης

Πιο αναλυτικά ξεκινάει με κλήση native AJAX η οποία επιστρέφει των κώδικα σελίδας που επισκέπτεται ο χρήστης. Το επόμενο βήμα είναι η αποστολή του κώδικα στο πρώτο εργαλείο JSDetox, μέσω του οποίου εξάγονται τα κομμάτια JavaScript.

Για κάθε κομμάτι JavaScript γίνεται έλεγχος από μια ειδική συνάρτηση την check_obfuscation [3], η οποία καθορίζει αν το τμήμα κώδικα είναι obfuscated ή όχι. Πιο συγκεκριμένα υπολογίζει την εντροπία, το μέγεθος λέξης και το n-gram τα οποία εισάγονται σε μια γραμμική συνάρτηση με κατάλληλα βάρη και δίνουν ένα αποτέλεσμα. Αν το αποτέλεσμα είναι μεγαλύτερο από το κατώφλι που έχει οριστεί τότε ο κώδικας είναι obfuscated. Σε αυτή την περίπτωση το εξεταζόμενο τμήμα προωθείται πάλι στο

JSDetox για να μετατραπεί σε de-obfuscated μορφή και να αποσταλεί στο jsunpack-n που θα αποφανθεί αν είναι malicious ή όχι.

Αν μέσω της συνάρτηση ο κώδικας δεν χαρακτηριστεί ως obfuscated τότε αποστέλλεται κατευθείαν στο jsunpack-n. Στο παράρτημα που ακολουθεί στο τέλος της εργασίας παρατίθεται ο κώδικας που υλοποιεί την προαναφερθείσα διαδικασία:

Για την άμεση ανάλυση του κώδικα της σελίδας που επισκέπτεται ο χρήστης από το jsunpack-n η επέκταση πραγματοποιεί μια κλήση AJAX στον server ο οποίος περιέχει το απαιτούμενο php αρχείο μέσω του οποίου γίνεται η εγγραφή σε ένα νέο αρχείο το κομμάτι JavaScript το οποίο έχει γίνει de-obfuscated από το JSDetox και στην συνέχεια το αρχείο αυτό μέσω της εντολής system καλείται εξωτερικά το jsunpack-n, κάνει την ανάλυση του επιλεγμένου αρχείου και επιστρέφει το αποτέλεσμα στην επέκταση. Ο κώδικας του php αρχείου είναι ο ακόλουθος:

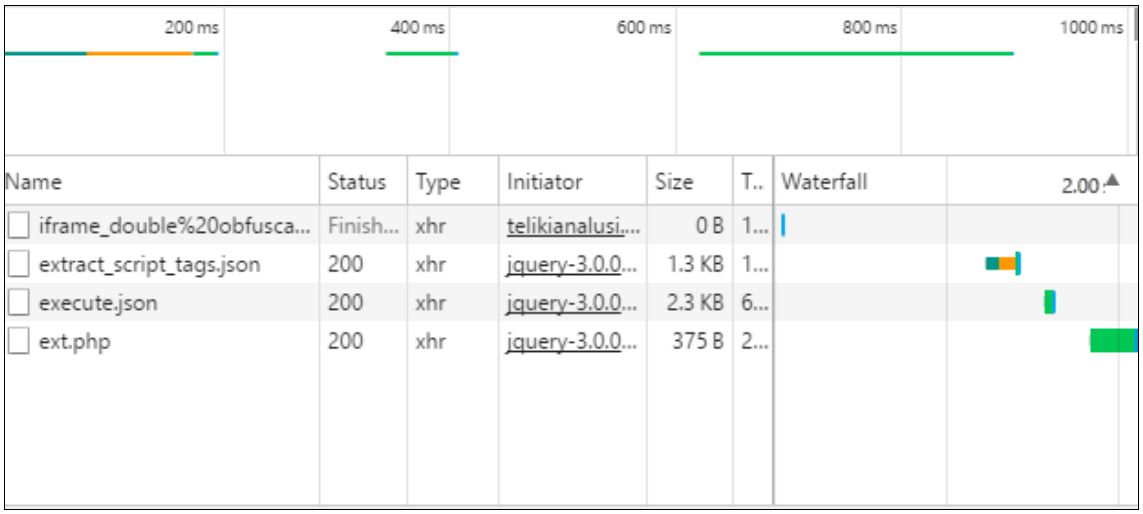
```
<?php


if (isset($_POST['datas'])) {
   $data = $_POST['datas'];
   $myfile = fopen("sample.js", "w") or die("Unable to open file!");

   fwrite($myfile, $data);


   fclose($myfile);




   $output = system('./jsunpackn.py -v /opt/lampp/htdocs/sample.js');
   echo $output;

}


?>
```

Έστω ότι υπάρχει το παρακάτω αρχείο με obfuscated κώδικα:

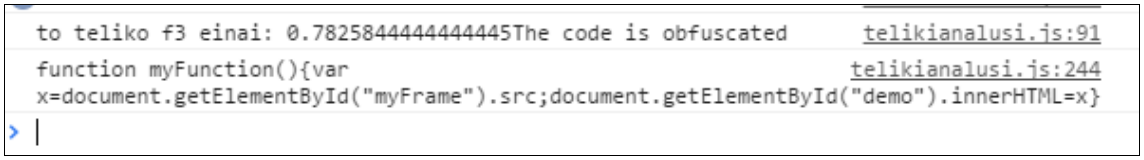
```
<!DOCTYPE html>
<html>
<body>
```

```
    <h3>A demonstration of how to access an IFRAME element</h3>

    <iframe id="myFrame" src="http://www.w3schools.com/"></iframe>

    <p>Click the button to get the URL of the iframe.</p>

    <button onclick="myFunction()">Try it</button>

    <p id="demo"></p>

    <script>
    eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?
    String.fromCharCode(c+29):c.toString(36))};
    if(!''.replace(/^/,String)){while(c--)r[e(c)]=k[c]||e(c);
    k=[function(e){return r[e]}];
    e=function(){return'\\w+'};
    c=1};
    while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);
    return p}('n(d(p,a,c,k,e,r){e=i;j(!\'\'.h(/^/,i)){m(c--)r[c]=k[c]||c;
    k=[d(e){f r[e]}];e=d(){f\'\\\\w+\'};c=1};m(c--)j(k[c])p=p.h(o q(\'\\\\b\'+e(c)+\'\\\\b\',\'g\'),k[c])
    ;f p}(\'3 4(){5 0=1.2("6").7;
    1.2("8").9=0}\',l,l,\'x|s|t|d|u|v|y|z|A|B\'.C(\'|\'),0,{}))',39,39,'||||||||||||||function||return|
    |replace|String|if||10|while|eval|new||RegExp||document|getElementById|myFunction|var|||myFr
ame
    |src|demo|innerHTML|split'.split('|'),0,{}))
    </script>

    </body>
</html>
```

Όταν το αναλύσουμε με την επέκταση θα παρατηρήσουμε στα dev-tools του chrome τις κλήσεις που γίνονται από και προς τα εργαλεία:



Στην κονσόλα βλέπουμε την τιμή της συνάρτησης check_obfuscation η οποία δείχνει αν ο κώδικας ή όχι είναι obfuscated, καθώς και ο κώδικας πως επιστρέφεται de-obfuscated από το JSDetox για να αποσταλεί στο jsunpack-n:



Τέλος, το αποτέλεσμα από το jsunpack-n επιστρέφει στην επέκταση όπως φαίνεται παρακάτω:

Κεφάλαιο 5ο

5. Συμπεράσματα

Στην παρούσα εργασία παρουσιάστηκε η υλοποίηση ενός αυτοματοποιημένου εργαλείου, το οποίο θα μπορεί να εντοπίζει την κακόβουλη δραστηριότητα της JavaScript στις σελίδες που επισκέπτεται ο χρήστης. Στόχος της εργασίας αποτελεί μια διαφορετική πρόταση εντοπισμού κακόβουλου κώδικα χρησιμοποιώντας τα πλεονεκτήματα από υπάρχοντα εργαλεία που έχουν αναπτυχθεί για τον σκοπό αυτό.

Αρχικά αναλύθηκαν οι τεχνολογίες με τις οποίες υλοποιήθηκε η επέκταση chrome. Στην συνέχεια αναφέρθηκαν οι βασικότερες επιθέσεις που πραγματοποιούνται από τους επιτιθέμενους μέσω της JavaScript και οι τεχνικές obfuscation που εφαρμόζονται για την απόκρυψη της πραγματικής φύσης του κώδικα. Στο τελευταίο κομμάτι γίνεται η παρουσίαση του τρόπου που υλοποιήθηκε η επέκταση. Σε αυτό περιλαμβάνεται, η παραμετροποίηση των open source εργαλείων JSDetox και jsunpack-n, έτσι ώστε να συνεργάζονται με την επέκταση, καθώς και οι βασικές αλλαγές που έπρεπε να εφαρμοστούν στην εικονική μηχανή που φιλοξενεί τα εργαλεία για να γίνει δυνατή η επικοινωνία με το front-end. Επίσης περιγράφεται ο τρόπος με τον οποίο γίνεται η εξέταση του κώδικα πηγής της σελίδας που επισκέπτεται ο χρήστης και την διαδρομή που ακολουθεί μέχρι το σημείο να χαρακτηριστεί αν είναι κακόβουλος ή όχι.

Η επέκταση που υλοποιήθηκε ασχολείται κυρίως, με τον εντοπισμό της πιο κύριας επίθεσης που απαντάται, η οποία είναι η drive-by-download. Η καθολική ανίχνευση των επιθέσεων που γίνονται μέσω της συγκεκριμένης γλώσσας αποτελεί δύσκολο στόχο, καθώς βρίσκεται ακόμα σε αρχικό στάδιο και αποτελεί πολύπλοκο θέμα ο επιτυχής διαχωρισμός ενός κώδικα που είναι obfuscated σε κακόβουλο ή όχι. Στη βιβλιογραφία τα εργαλεία και οι μεθοδολογίες που έχουν προταθεί έχουν καλό ποσοστό επιτυχίας, αλλά η υλοποίησή τους είναι αρκετά δύσκολη, συνήθως όχι αυτοματοποιημένη και ο χειρισμός τους απαιτεί εξειδικευμένη γνώση από έμπειρους ερευνητές. Συνεπώς η επέκταση που υλοποιήθηκε στην συγκεκριμένη εργασία αποτελεί μια εναλλακτική πρόταση που προσπαθεί να υπερκεράσει τα παραπάνω εμπόδια συνδυάζοντας λειτουργίες από υπάρχοντα εργαλεία και μπορεί να χρησιμοποιηθεί από απλούς χρήστες του διαδικτύου.

Θα πρέπει να τονιστεί ότι υπάρχει περιθώριο βελτιστοποίησης της επέκτασης, με την προσθήκη και επιπλέον εργαλείων τα οποία θα πραγματοποιούν πιο ενδελεχή ανάλυση του κώδικα και μέσω του συνδυασμού των λειτουργιών θα γίνεται με μεγαλύτερη επιτυχία ο εντοπισμός κακόβουλου λογισμικού.

Κεφάλαιο 6ο

6. Βιβλιογραφικές Πηγές

1. Marco Cova, Christopher Kruegel, and Giovanni Vigna, Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code, University of California, Santa Barbara

2. Byung-Ik Kim, Chae-Tae Im, Hyun-Chul Jung, Suspicious Malicious Web Site Detection with Strength Analysis of a JavaScript Obfuscation, Korea Internet & Security Agency

3. Corrado Aaron Visaggio, Giuseppe Antonio Pagin and Gerardo Canfora, An Empirical Study of Metric-Based Methods to Detect Obfuscated Code, Department of Engineering, University of Sannio 82100 Benevento, Ital. International Journal of Security and Its Applications, Vol. 7, No. 2, March, 2013

4. Wei Xu, Fangfang Zhang and Sencun Zhu, The Power of Obfuscation Techniques in Malicious Javascript Code: A Measurement Study, Department of Computer Science and Engineering, The Pennsylvania State University, University Park

5. Gerardo Canfora, Francesco Mercaldo, Corrado Aaron Visaggio, Malicious JavaScript Detection by Features Extraction, Department of Engineering, University of Sannio. E-Informatica Software Engineering Journal, Volume 8, Issue 1, 2014, pages 65-78, DOI 10.5277/e-Inf140105

6. Peter Likarish, Eunjin (EJ) Jung, Dept. of Computer Science, the University of Iowa and Insoon Jo, Distributed Computing Systems Lab, School of Computer Science and Engineering, Obfuscated Malicious Javascript Detection using Classification Techniques

7. Wei Xu, Fangfang Zhang, Sencun Zhu, JStill: Mostly Static Detection of Obfuscated Malicious JavaScript Code, Department of Computer Science and Engineering Pennsylvania State University University Park, PA

8. Daiping Liu, Haining Wang, Department of Computer Science, College of William and Mary, Angelos Stavrou, Center for Secure Information Systems, George Mason University, Detecting Malicious Javascript in PDF through Document Instrumentation

9. Igino Corona, Davide Maiorca, Davide Ariu, Giorgio Giancinto, Department of Electrical and Electronic Engineering, University of Cagliari, Lux0R: Detection of Malicious PDF-embedded JavaScript code through Discriminant Analysis of API References

10. Mehran Jodavi, Mahdi Abadi, and Elham Parhizkar Department of Electrical and Computer Engineering Tarbiat Modares University Tehran, Iran

11. Yao Wang*, Wan-dong Cai and Peng-cheng Wei, Department of Computer Science and Technology, Northwestern Polytechnical University, Xi'an, China, A deep learning approach for detecting malicious JavaScript code

12. "JSDetox", http://www.relentless-coding.com/projJSObfusDetector: A Binary PSO-based One-Class Classifier Ensemble to Detect Obfuscated JavaScript Code ects/jsdetox/

13. "jsunpack-n", https://www.aldeid.com/wiki/Jsunpackn

14. "JavaScript Obfuscators", https://javascriptobfuscator.com/Javascript-Obfuscator.aspx, http://www.danstools.com/javascript-obfuscate/index.php

15. "Examples of malicious javascript", https://aw-snap.info/articles/js-examples.php

16. "PHP examples", http://php.net/manual/en/zip.examples.php

17. "AJAX", https://www.w3schools.com/xml/ajax_intro.asp

18. “JavaScript Examples”, https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_prototype1, http://www.quackit.com/html/html_editors/scratchpad/?example=/javascript/examples/javascript_date_current_date_ddmmyyyy, https://www.w3schools.com/jsref/jsref_eval.asp, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects,

19. “Bootstrap Examples”, https://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/

20. “Chrome Extension”, https://developer.chrome.com/extensions/getstarted

Κεφάλαιο 7ο

7. Παράρτημα κώδικα JavaScript που πραγματοποιεί την ανάλυση

```
document.addEventListener('DOMContentLoaded', function() {
  var checkPageButton = document.getElementById('checkPage');
  checkPageButton.addEventListener('click', function() {

    chrome.tabs.getSelected(null, function(tab) {


      var opts = {
        exec_eval: true,
        no_trace_vars: false,
        sc_arch: 32
      }

      //function that checks if page source is obfuscated or not
      function check_obfuscation(check) {
        var metric11 = check.substr(0, check.length - 4);
        var metric1 = metric11.substr(2);
        var charoliko = metric1.length;
        var sizebyteteliko = [];

        //n-gram computation
        var occurence =
metric1.match(/\`|\~|\!|\@|\#|\$|\%|\^|\&|\*|\(|\)|\+|\=|\[|\{|\]|\}|\||\\|\'|\<|\,|\.|\>|\?|\/|\""|\;|\:|\sxu/gi);

        var occurence1 = occurence.length;
        f = occurence1 / charoliko;

        //htmldocsize computation
        var res = metric1.split(";");
        for (var j = 0; j < res.length; j++) {
          var sizebyte = new TextEncoder('utf-8').encode(res[j]).length;
          sizebyteteliko.push(sizebyte);
        }
```

```
var max_htmldocsize = Math.max.apply(null, sizebyteteliko);

//entropy computation
// The string
var str = metric1;
var sizebyteoliko = (new TextEncoder('utf-8').encode(metric1).length);

// A map (in JavaScript, an object) for the character=>count mappings
var counts = {};
var entropytable = [];
// Misc vars
var ch, index, len, count;

// Loop through the string...
for (index = 0, len = str.length; index < len; ++index) {
   // Get this character
   ch = str.charAt(index); // Not all engines support [] on strings

   // Get the count for it, if we have one; we'll get `undefined` if we
   // don't know this character yet
   count = counts[ch];

   // If we have one, store that count plus one; if not, store one
   // We can rely on `count` being falsey if we haven't seen it before,
   // because we never store falsey numbers in the `counts` object.
   counts[ch] = count ? count + 1 : 1;
}

for (ch in counts) {

   var fchar = (counts[ch] / sizebyteoliko).toFixed(2);
   var logarithmos = (Math.log10((counts[ch] / sizebyteoliko))).toFixed(2);

   var entropychunk = -(fchar * logarithmos);
   entropytable.push(entropychunk);

}
```

```
    for (var j = 0; j < entropytable.length; j++) {
      console.log(entropytable[j]);

    }


    //jquery function for finding the sum of an array

    var i = entropytable.length;
    var sum = 0;
    while (--i) sum += entropytable[i];

    //final f3 function for obfuscation check
    f3 = (0.2 * sum) + ((0.4 * f) / 10) + ((0.4 * max_htmldocsize) / 100);

    if (f3 > 0.6) {

      console.log("Final value of the function f3 is: " + f3 + "The code is obfuscated");
      return true;
    } else {
      console.log("The code is not obfuscated");
      return false;
    }




}



xmlhttp = new XMLHttpRequest();
var x = tab.url;


xmlhttp.open("GET", x, true);
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4) {
```

```
source = xmlhttp.responseText;


var htmldoc = source;

alert(htmldoc);
var xhrForm = new XMLHttpRequest();
$('#loadingmessage').show();
$.ajax({
   url: "http://192.168.2.18/analysis/backend/extract_script_tags.json",
   crossDomain: true,
   type: "post",
   data: {
      action: 'extract_script_tags',
      htmldoc: htmldoc
   },
   success: function(response) {
      if (response['status'] == 'ok') {
         if (response['tag_count'] > 0) {

            var newarray = [];
            for (var i = 0; i < response.tags.length; i++) {
               if (response.tags[i] !== null && response.tags[i] !== "") {
                  newarray.push(response.tags[i]);
               }
            }
            response.tags = newarray;




            for (var i = 0; i < response['tags'].length; i++) {


               if (!check_obfuscation(response['tags'][i])) {
```

```
        //send js tags to jsunpack-n for the final analysis

        var data = response['tags'][i];

        $.ajax({
          url: "http://192.168.2.18/ext.php",
          crossDomain: true,
          type: "post",
          data: {
            'datas': data
          },
          success: function(result) {


            result = result.replace("/opt/lampp/htdocs/sample.js", "");
            var div = document.createElement('div');
            document.body.appendChild(div);
            var p = document.createElement('p'),
              txt = document.createTextNode(result);
            p.appendChild(txt);
            document.body.appendChild(p);

          },
          error: function(xhr, status, error) {

            alert("Something went wrong with the network.");


          },
          dataType: 'text'
        });

      } else {


        //sending code to jsdetox
        $.ajax({
          url: "http://192.168.2.18/analysis/backend/execute.json",
          type: "post",
```

```
                                    //async: false,
                                    data: {
                                       action: 'execute',
                                       code: response['tags'][i],
                                       opts: opts,
                                       htmldoc: htmldoc
                                    },
                                    success: function(response1) {

                                       if (response1['log'] != null && (response1['log'][0]['type'] == "error" ||
response1['log'][0]['type'] == "info")) {
                                          alert("Code cannot be executed with jsdetox, so contnue with jsunpack-n");



                                       } else if (response1['log'] != null && !check_obfuscation(response1.log[0].data['raw'])
&& Object.values(response1.log).indexOf('warning') > -1) {
                                          var raw1 = response1.log[0].data['raw'];
                                          alert(response1.log);




                                          var data = raw1;



                                          $.ajax({
                                             url: "http://192.168.2.18/ext.php",
                                             crossDomain: true,
                                             type: "post",
                                             data: {
                                                'datas': data
                                             },
                                             success: function(result) {



                                                result = result.replace("/opt/lampp/htdocs/sample.js", "");
                                                var div = document.createElement('div');
                                                document.body.appendChild(div);
                                                var p = document.createElement('p'),
```

```
                txt = document.createTextNode(result);
            p.appendChild(txt);
            document.body.appendChild(p);

        },
        error: function(xhr, status, errorThrown) {
            alert("Something went wrong with the network");


        },
        dataType: 'text'
    });


} else if (response1['log'] != null && check_obfuscation(response1.log[0].data['raw'])) {

    var data = response1.log[1].data['raw'];
    $.ajax({
        url: "http://192.168.2.18/ext.php",
        crossDomain: true,
        type: "post",
        data: {
            'datas': data
        },
        success: function(result) {


            result = result.replace("/opt/lampp/htdocs/sample.js", "");
            var div = document.createElement('div');
            document.body.appendChild(div);
            var p = document.createElement('p'),
                txt = document.createTextNode(result);
            p.appendChild(txt);
            document.body.appendChild(p);

        },
        error: function(xhr, status, error) {
            alert("Something went wrong with the network");
            // check status && error
```

```
            },
            dataType: 'text'
        });




    } else {
        var data12 = response['tags'][i];
        $.ajax({
            url: "http://192.168.2.18/ext.php",
            crossDomain: true,
            type: "post",
            data: {
                'datas': data12
            },
            success: function(result) {


                result = result.replace("/opt/lampp/htdocs/sample.js", "");
                var div = document.createElement('div');
                document.body.appendChild(div);
                var p = document.createElement('p'),
                    txt = document.createTextNode(result);
                p.appendChild(txt);
                document.body.appendChild(p);

            },
            error: function(xhr, status, error) {
                alert("Something went wrong with the network");
                // check status && error
            },
            dataType: 'text'
        });

    }

},
error: function(xhr, status, error) {
```

```
                        alert("Something went wrong with the network");
                        // check status && error
                    },
                    dataType: 'json'
                });



            }


        }


    } else {
        alert("No js script tags found");
    }
} else {
    alert("Extration of js script tags failed");
}
$('#loadingmessage').hide();
},
error: function(xhr, status, error) {
    alert("Something went wrong with the network");
    // check status && error
},
dataType: 'json'

});



}
}

xmlhttp.send(null)
```

```
            });
        }, false);
}, false);
```