ANGULAR JS VE NODE JS GÜVENLİĞİ

Yazar: Kasım FEKE
Mentor: Osman Melih ZEYTUN
Baskı: 2018

İÇİNDEKİLER

Giriş

AngularJS ve Node.js son dönemin popüler javascript kütüphaneleridir. Popüler olması elbette kötü niyetli kişilerin de ilgisini çekmektedir. Bu yazıda AngularJS ile Node.js üzerindeki zafiyetlere ve bu zafiyetlere karşı alınacak önlemlere değinilecektir.

AngularJS ve Node.js'den bahsetmeden önce JavaScript, JavaScript Kütüphaneleri, MVC, açık kaynak kod, element, DOM ve directive kavramlarına değinmemizde yarar vardır.

JavaScript

Java ve Script kelimelerinin birleştirilmesi sonucu bu adı alan JavaScript bir betik dilidir. Java ve script kelimelerinin baş harflerinden hareket ile JS şeklinde kısaltma yapılmaktadır. Script'in Türkçe karşılığı bilişim sözlüklerinde 'Betik' olarak tabir edilmektedir. Bir tiyatro oyununa ait senaryoyu scripte yani betiğe benzetebiliriz, oyun boyunca oyuncuların ne yapacakları ve pozisyonlarını içerir, oyunu sergilemek için oluşturulmuş talimatlar serisi de diyebiliriz.

Java ve JavaScript'in aslında tek ortak yönü isimlerinde java kelimesinin geçmesidir. 1995 yılında Sun Microsystems firması Java adında yeni bir programlama dili piyasaya sürüyor daha sonra Netscape firması bir betik dili geliştiriyor ve bunun daha popüler olması adına ismini JavaScript koyarak piyasaya sürüyor. Java programlama dili iken JavaScript bir betik dilidir ve ikisini karıştırmamamız gerekir.

JavaScript Kütüphaneleri

JavaScript kodlarından derlenip toparlanmışlardır. Direkt olarak projelere dahil edilerek farklı ihtiyaçlara göre kullanılabilirler. jQuery, AngularJS, Node.js gibi JavaScript kütüphaneleri mevcuttur.

MVC

Bir uygulamanın Model, View ve Controller adında üç temel katman içerisinde geliştirilmesi esasına dayanan bir tasarım kalıbıdır diyebiliriz. Tasarım kalıpları, sıkça karşılaşılan ve birbirine benzeyen sorunları çözmek için geliştirilmiş esnek kalıplardır. MVC, hazırlanan uygulamanın iş mantığı ile kullanıcı arayüzünü birbirinden ayrıştırarak karışmasını engellemektedir. Model, view ve controllerdan kısaca bahsedelim.

Model

Uygulamanın veritabanı ile ilişkileri bu katmanda gerçekleşir. Örnek vermek gerekir ise bir satranç uygulamasında taşların hareketinin doğru olup olmadığı veya oyunun bitip bitmediği gibi bilgiler modelda yer alacaktır.

View

Uygulamanın kullanıcılar tarafından görüldüğü kısım yani arayüzdür. View için örnek verecek olursak satranç tahtasının görünümü, piyonların vs. şekilleri burada yer alır.

Controller

Kabaca tabir ile Model ve view arasında getir götür işlemlerini gerçekleştirir. Kullanıcıların view'da yaptığı işlemlerdeki veriyi alarak model'a taşır veya model'dan aldığı veriyi view üzerinden kullanıcıya gösterir. Örneğin kullanıcı "Yeni Oyun Başlat" butonuna bastı, controller model'a giderek böyle bir istek geldiğini söylüyor ve topu artık model'a atıyor.

Açık Kaynak Kod

Kaynak kodları herkese açık olarak sunulan yazılımlardır. Kaynak kodlar üzerinde değiştirmeler yapabilir veya angularJS, node.js gibi açık kaynak kütüphaneler projelere dahil edilebilir.

Element

Daha iyi anlaşılması için Html'den bir örnek ile açıklayalım. Html'de bir içeriği kalın yapmak için etiketleri arasında yazarız, mesela Deneme yazdık bu bir element olmaktadır.

DOM

Web tarayıcıları html dosyalarını yüklerken her elementi bir nesne olarak kabul eder. Bundan hareket ile resim, yazı, form gibi tüm elementlerin bir nesne olduğunu söyleyebiliriz. Betik dilleri vasıtası ile bu nesnelere müdahale edebilmemiz mümkündür.

Directive

Direktifler sayesinde html'e yeni etiketler veya özellikler eklenebilir. AngularJS'in kendi direktifleri haricinde özel direktifler de kullanılabilir. Direktifler ng- ile başlar, detaya AngularJS başlığında değinilecektir.

AngularJS Nedir ve Nasıl Kullanılır?

Google tarafından desteklenen, MVC kullanılarak geliştirilen, tek sayfa uygulamaları geliştirilmesini sağlayan ve istemci taraflı çalışan bir JavaScript kütüphanesidir. İstemci taraflı çalıştığından dolayı sunucuları yormazlar. Html üzerinde kullanımını görelim.

```
<html ng-app>
        <title>AngularJS Örnek</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.5/angular.js"></script>
    <body>
    Ad Soyad: <input type="text" ng-model="adsoyad" />
    <br /><br />
    Merhaba Sn. {{adsoyad}}
   </body>
</html>
```

En başta ng-app direktifini html etiketine ekleyerek Angular'ı html dosyasının tümünde kullanacağımızı belirtiyoruz. Script etiketlerini kullanarak ise AngularJS'in 1.6.5 sürümünü bulunduğu uzak kaynakdan sayfaya dahil ediyoruz, bu işlem https://angularjs.org/ dan Download AngularJS butonu ile indirilerek dahil edilme şeklinde de gerçekleştirilebilir. Ad Soyad: dedikten sonra girilen ad ve soyad verisini almak için input konuluyor, type yani tipi text olarak belirlenmiş tek satırlık bir metin alanı tanımlanıyor ve ng-model direktifi ile inputa girilen verinin değişken şeklinde kullanılması sağlanıyor. Merhaba Sn. kelimesinden sonra da {{adsoyad}} şeklinde yukarıda girilen ad ve soyadı yazdırıyoruz.

Hiçbir veri girilmemiş hâli bu şekilde, şimdi de veri girelim ve bakalım.

Görüldüğü üzere input ile alınan veriyi aşağıda yazdırdı ve AngularJS kullanıldığı için herhangi bir buton aracılığı ile göndermeye gerek kalmadan anlık şekilde işlem gerçekleştirilmiş oldu, istemci tarafında gerçekleştiği için bunun gibi işlemler uygulamaların çalıştığı sunucuları yormayacaktır.

DEMO

AngularJS 1.5.7 ve 1.5.8 sürümlerinde güvenlik amacı ile kullanılan Sandbox bypass edilerek zararlı JavaScript kodları çalıştırılabilmektedir. Sandbox ile sadece toString(), charAt(), trim(), prototype, ve constructor işlevleri ile nesnelerine izin verilmektedir. Sandbox Bypass ile ilgili bir demo yapalım.

```
<!DOCTYPE html>
<html>
<head>
    <title>Sandbox Bypass</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
</head>
<body>

<div ng-app="">
    {{a=toString().constructor.prototype;a.charAt=a.trim;$eval('a,alert("BGA"),a')}}
</div>

</body>
</html>
```

Yukarıda görüldüğü üzere **{{a=toString().constructor.prototype;a.charAt=a.trim;$eval('a,alert("BGA"),a')}} kodu ile BGA** alertini verdirmeye çalışıyoruz.

Sonuç başarılı.

Nasıl Önlem Alınabilir?

1- Direktifler html dosyalarının tamamında değil, sadece Angular kodlarını içeren html elementlerine atanmalıdır ve bu şekilde kapsam sınırlandırılmalıdır.
2- 1.6 veya üstü sürümlerde Sandbox kaldırılmıştır bu sürümler kullanılabilir.

Node.js Sızma Testleri

Node.js 2009 yılında Joyent firmasında çalışan Ryan Dahl tarafından geliştirilen Javascript Çalışma Ortamıdır. Ryan Dahl, Joyent firmasının desteği ve V8 motorunun gücü ile JavaScript'i Back-end kısmına taşımıştır. V8 motoru C/C++ programlama dilleri ile geliştirilmiş JavaScript komutlarını makine diline çevirmek için kullanılan bir ara yazılımdır. Komutların makine koduna çevrilmesi JavaScript komutlarının daha hızlı ve performanslı çalışmasını sağlar.

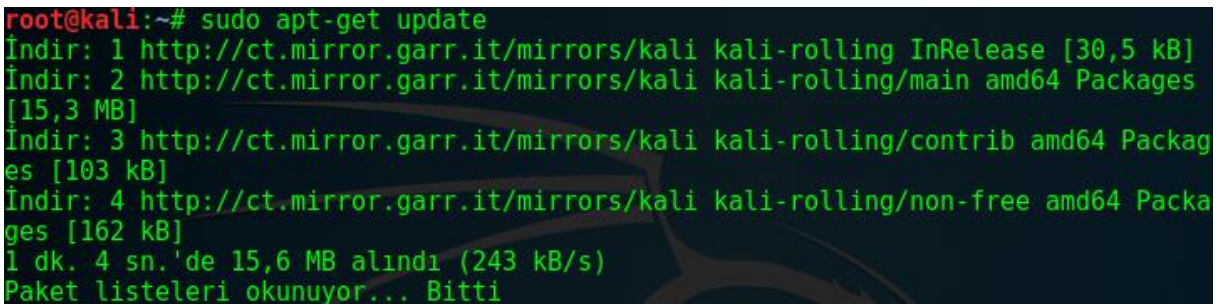
Back-end ise web uygulamasının veri tabanı mimarisinin oluşturulduğu, kullanıcı yetkilendirmelerinin yapıldığı, işlevselliği ile ilgili kodların yazıldığı yerdir kısaca bir web uygulamasının arka planıdır denebilir.

Normalde sunucu taraflı çalışan programlama dillerinde herhangi bir kullanıcı istekte bulunduğunda sunucu sadece o isteğe cevap verir ve diğer istekler kuyruğa alınır. Bir isteğin uzun sürmesi diğer kullanıcıları etkiler ancak Node.js komutları bloklamadan işlediğinden işlemi uzun süren komut sistemi yavaşlatmaz ve Node.js diğer kullanıcılara da cevap verir. Buna güzel bir örnek olarak yemek sipariş sistemi verilebilir.

Klasik sunucu taraflı programlama dillerinde bir yemek siparişi geldiğinde sırada duran diğer müşteriler siparişin hazırlanmasını bekler. Node.js ise herhangi bir yemek siparişi geldiğinde siparişi arka tarafa bildirir ve not alır ve daha sonra sıradaki müşterinin siparişini alır. Verilen yemek siparişlerinin hangisi daha önce hazırlanırsa o yemek siparişine cevap verir. Böylece müşteri daha önce ve hızlı hazırlanacak bir yemek için sırada fazladan beklemez. Bu yapı sayesinde anlık mesajlaşma, oyun sistemleri vb. gerçek zamanlı uygulamalar kolaylıkla ve daha az maliyetle yapılabilir.

Node.js'in Yüklenmesi

Anlatım linux için yapılacaktır, windows kullanıcıları https://nodejs.org/en/download/ üzerinden sisteme uygun dosyayı indirerek kurulumu basit bir şekilde gerçekleştirebilir.



1- Terminali açarak sudo apt-get update komutunu veriyoruz, bu komut ile apt paket yöneticisinin güncellenmesini sağlıyoruz. Sudo bir kullanıcıya root yetkisi verilmesi için kullanılmaktadır.

```
root@kali:~# sudo apt-get install nodejs
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki paketler yükseltilecek:
  nodejs
1 paket yükseltilecek, 0 yeni paket kurulacak, 0 paket kaldırılacak ve 1397 pake
t yükseltilmeyecek.
0 B/3.334 kB arşiv dosyası indirilecek.
Bu işlem tamamlandıktan sonra 545 kB disk alanı boşalacak.
Reading changelogs... Done
(Veritabanı okunuyor ... 305994 dosya veya dizin kurulu durumda.)
Paket açılacak: .../nodejs_4.8.4~dfsg-1_amd64.deb ...
Paket açılıyor: nodejs (4.7.2~dfsg-2+b1) üzerine (4.8.4~dfsg-1) ...
Ayarlanıyor: nodejs (4.8.4~dfsg-1) ...
Tetikleyiciler işleniyor: man-db (2.7.6.1-2) ...
```

2- sudo apt-get install nodejs komutu apt paket yöneticisi ile node.js yüklemesini gerçekleştirir.

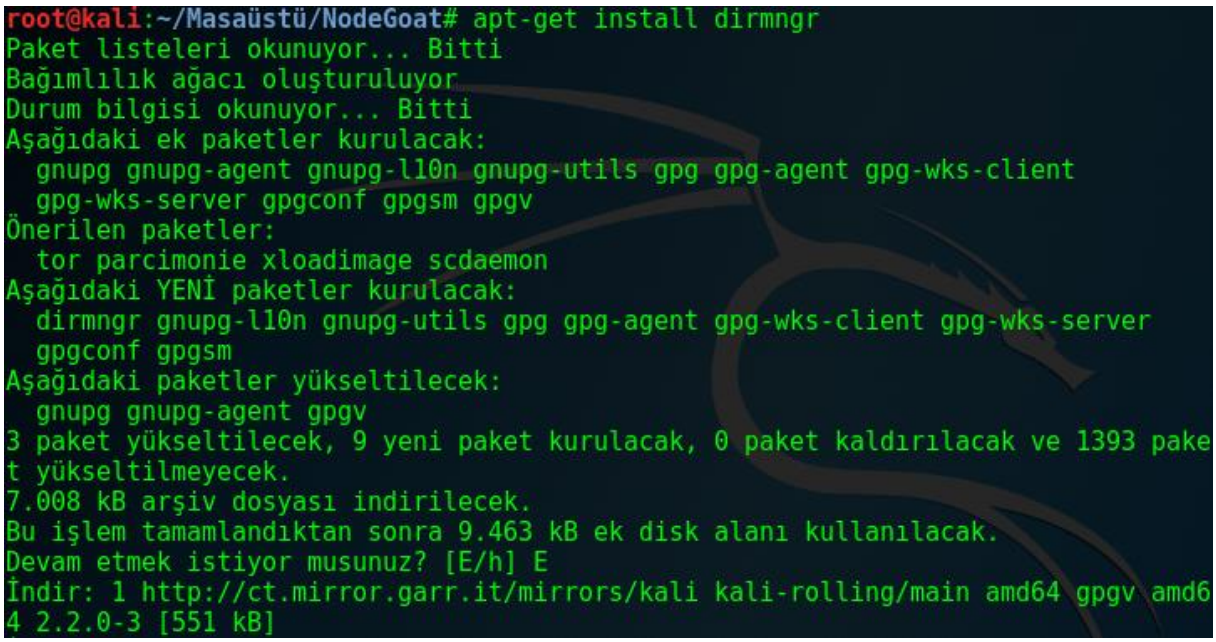
```
root@kali:~# apt install nodejs-legacy
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki YENİ paketler kurulacak:
  nodejs-legacy
0 paket yükseltilecek, 1 yeni paket kurulacak, 0 paket kaldırılacak ve 1397 pake
t yükseltilmeyecek.
292 kB arşiv dosyası indirilecek.
Bu işlem tamamlandıktan sonra 332 kB ek disk alanı kullanılacak.
İndir: 1 http://ct.mirror.garr.it/mirrors/kali kali-rolling/main amd64 nodejs-le
gacy all 4.8.4~dfsg-1 [292 kB]
1 sn.'de 292 kB alındı (207 kB/s)
Daha önce seçili olmayan nodejs-legacy paketi seçiliyor.
(Veritabanı okunuyor ... 305994 dosya veya dizin kurulu durumda.)
Paket açılacak: .../nodejs-legacy_4.8.4~dfsg-1_all.deb ...
Paket açılıyor: nodejs-legacy (4.8.4~dfsg-1) ...
Ayarlanıyor: nodejs-legacy (4.8.4~dfsg-1) ...
Tetikleyiciler işleniyor: man-db (2.7.6.1-2) ...
```

3- Node.js için ufak bir düzeltme yapılması gerekmektedir, bilgi karmaşası olmaması adına detay verilmeyecektir. Kullanılacak komut: sudo apt install nodejs-legacy

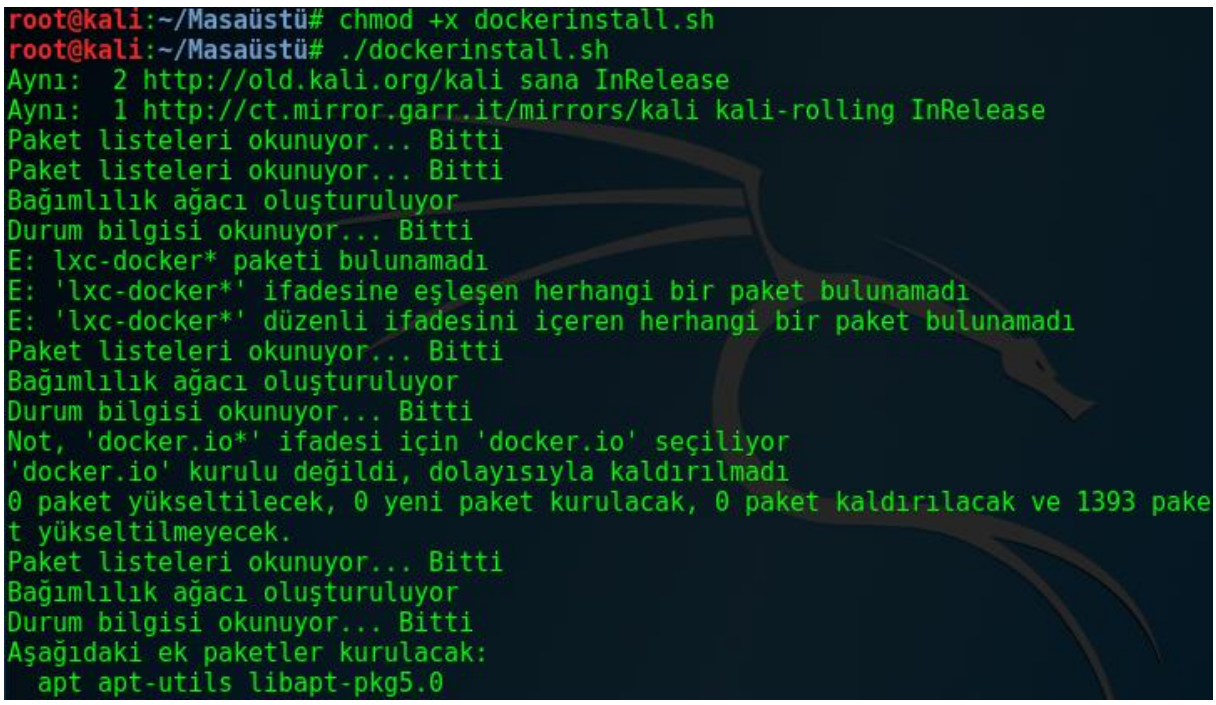
```
root@kali:~# node
>
```

Node komutu verildiğinde böyle bir görüntü ile karşılaşılır ise kurulumun başarılı şekilde gerçekleştiğine işarettir.

Docker Kurulumu



1- NodeGoat'ın çalıştırılması için sanallaştırma platformu olan Docker kurulumunu gerçekleştirirken ilk olarak sudo apt-get install dirmngr komutu kullanılır.



2- https://gist.github.com/apolloclark/f0e3974601346883c731 adresinde bulunan bash kodları kopyalanarak masaüstüne dockerinstall.sh adı ile kayıt edilir daha sonrasında chmod +x dockerinstall.sh ile dosyaya çalıştırılma izini verildikten sonra ./dockerinstall.sh komutu ile çalıştırılır (dockerinstall isimi örnek olarak verilmiştir farklı şekilde de adlandırılarak çalıştırılabilir)

```
root@kali:~/Masaüstü/NodeGoat# apt-get install docker-compose
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki ek paketler kurulacak:
  python-cached-property python-docker python-dockerpty python-docopt
  python-funcsigs python-jsonschema python-mock python-pbr python-texttable
  python-websocket
Önerilen paketler:
  python-funcsigs-doc python-mock-doc
Tavsiye edilen paketler:
  docker.io
Aşağıdaki YENİ paketler kurulacak:
  docker-compose python-cached-property python-docker python-dockerpty
  python-docopt python-funcsigs python-jsonschema python-mock python-pbr
  python-texttable python-websocket
0 paket yükseltilecek, 11 yeni paket kurulacak, 0 paket kaldırılacak ve 1388 pak
et yükseltilmeyecek.
394 kB arşiv dosyası indirilecek.
```

3- sudo apt-get install docker-compose ile de Docker kurulumunu tamamlanır.

NodeGoat Kurulumu

Sızma testleri OWASP'ın hazırlamış olduğu NodeGoat test ortamı üzerinden gerçekleştirilecektir, localhost'a kurulum yapılmasına gerek kalmadan http://nodegoat.herokuapp.com/login adresinden erişilebilir.

Biri Admin olmak üzere üç adet kayıtlı kullanıcı mevcuttur, giriş bilgileri:

admin - Admin_123
user1 - User1_123
user2 - User2_123

Ayrıca sign up sayfasından yeni kullanıcılar eklenebilir. Localhost'da kuruluma geçilecek olursa:

```
root@kali:~# cd Masaüstü
root@kali:~/Masaüstü# git clone https://github.com/OWASP/NodeGoat.git
Cloning into 'NodeGoat'...
remote: Counting objects: 5034, done.
remote: Total 5034 (delta 0), reused 0 (delta 0), pack-reused 5034
Receiving objects: 100% (5034/5034), 7.78 MiB | 985.00 KiB/s, done.
Resolving deltas: 100% (1092/1092), done.
root@kali:~/Masaüstü#
```

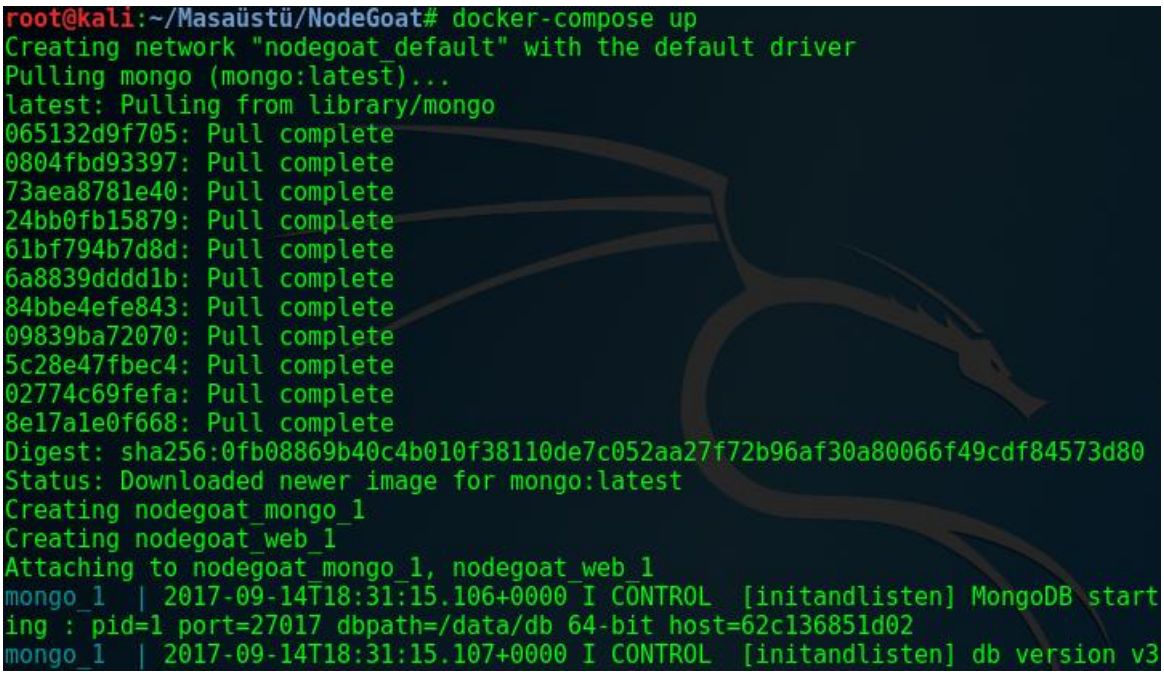
1. cd Masaüstü ile masaüstüne geçiş yaptık ve git clone https://github.com/OWASP/NodeGoat.git komutu ile NodeGoat'ın kaynak dosyalarını çekiyoruz.

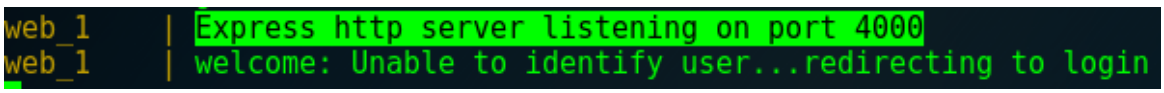
```
root@kali:~/Masaüstü# cd NodeGoat
root@kali:~/Masaüstü/NodeGoat# ls
app         config              Gruntfile.js  Procfile   test
app.json    docker-compose.yml  LICENSE       README.md
artifacts   Dockerfile          package.json  server.js
root@kali:~/Masaüstü/NodeGoat#
```

2. cd NodeGoat komutu ile dosyaların bulunduğu klasöre geçiyoruz ve ls ile baktığımızda NodeGoat dosyalarını görebilmekteyiz.

```
root@kali:~/Masaüstü/NodeGoat# docker-compose build
mongo uses an image, skipping
Building web
Step 1/17 : FROM node:4.4
4.4: Pulling from library/node
357ea8c3d80b: Pull complete
52befadefd24: Pull complete
3c0732d5313c: Pull complete
ceb711c7e301: Pull complete
868b1d0e2aad: Pull complete
3a438db159a5: Pull complete
Digest: sha256:e720e944ce6994a461cd2a9e0ae34c4bc45c0f9ee7b3f48052933182fc5f0bf1
Status: Downloaded newer image for node:4.4
 ---> 93b396996a16
Step 2/17 : ENV user nodegoat_docker
 ---> Running in bbfb2dde2f51
 ---> 8ced485844a4
Removing intermediate container bbfb2dde2f51
Step 3/17 : ENV workdir /usr/src/app/
 ---> Running in 3d9b70367eb2
 ---> 26414114e574
Removing intermediate container 3d9b70367eb2
Step 4/17 : RUN useradd --create-home --system --shell /bin/false $user
```

3. sudo docker-compose build komutu ile Docker üzerinde NodeGoat kurulumunu yapıyoruz.





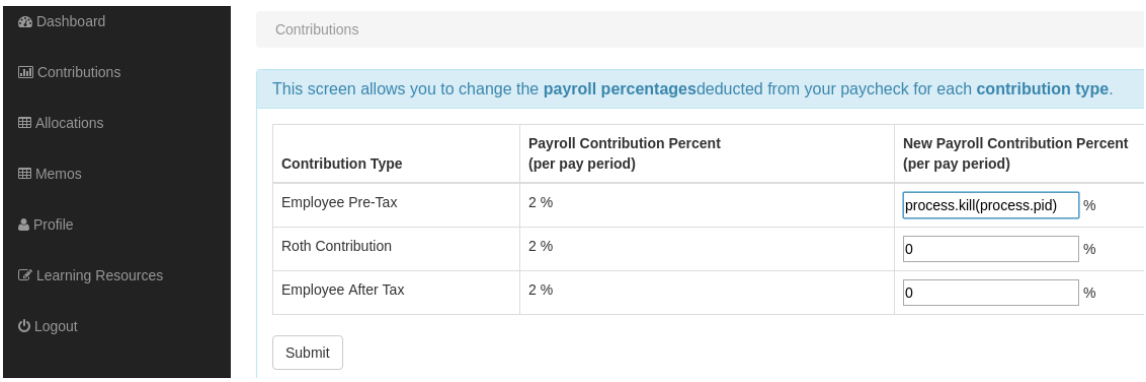
4. Son olarak sudo docker-compose up komutu ile NodeGoat'ı çalıştırıyoruz (4000 portunun dinlenmeye başladığı görülmektedir, localhost:4000 adresinden NodeGoat'a erişilebilir).

Zafiyetler

Server Side JS Injection

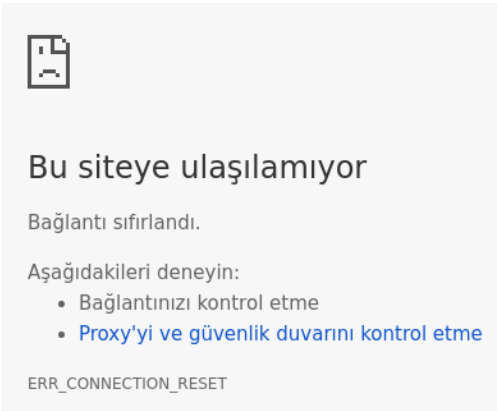
Node.js'in sunucu taraflı çalıştığından bahsetmiştik, server side js injection Türkçe'ye sunucu taraflı javascript enjeksiyonu olarak çevrilebilir. Gerekli önlemler alınmaz ise zararlı kodlar enjekte edilerek Hizmet Aksatma Saldırıları (DOS Attack), Dosyalara Erişme (File System Access) ve domaininiz kullanılarak Oltalama Saldırıları (Phishing) gerçekleştirilebilir bunlar sadece yapılabileceklerden birkaç tanesidir.

DOS Attack



NodeGoat paneline giriş yaptıktan sonra zafiyetin barındığı Contributions sayfasına giriyoruz, ilk amacımız DOS saldırısı gerçekleştirmek olduğu için bu işlemi nasıl gerçekleştirebiliriz diye bir düşünelim.

İlk olarak akla gelen, hedef sunucuyu kendi içinde döngüye sokarak başka işlemlere cevap veremez hale getirmek olabilir. İkinci olarak da mesela çalışan işlemler sonlandırılabilir. Sistemde çalışan, kendi kişiliği olan her işlem bir processdir. Processler numara olarak takip edilir, buna da PID denir.

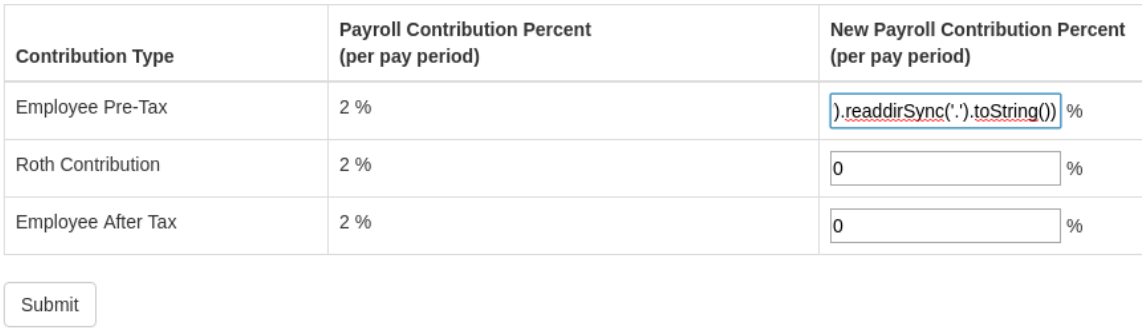


Yukarıdaki ekran görüntüsünde görüldüğü üzere process.kill (process.pid) kodunu sunucuda çalışan işlemleri sonlandırmak için submit diyerek enjekte ediyoruz.

Ve sonuç yukarıdaki gibi. Sunucunun tekrar aktif hale gelmesi için reboot edilmesi yani yeniden başlatılması gerekmektedir.

Döngüye sokarak DOS saldırısı yapılmak istenir ise de while(1) kodu ile bu işlem gerçekleştirilebilir.

File System Access



Contribution Type	Payroll Contribution Percent (per pay period)	New Payroll Contribution Percent (per pay period)
Employee Pre-Tax	2 %	).readdirSync('.').toString()) %
Roth Contribution	2 %	0 %
Employee After Tax	2 %	0 %

Submit

.git,.gitignore,.jshintrc,.nodemonignore,Dockerfile,Gruntfile.js,LICENSE,Procfile,README.md,app,app.json,artifacts,config,docker-compose.yml,node_modules,npm-debug.log,package.json,server.js,test

Sunucudan dosyaların okunması için res.end(require('fs').readdirSync('.').toString()) kodu enjekte edilebilir. res.end ile diğer işlemler durdurularak sunucuda gönderilen kodun işlenmesini sağlıyoruz, require('fs') ile dosyaları istediğimizi belirtiyoruz, readdirSync('.') ile bir üst dizindeki dosyaları çağırıyoruz bu şekilde ana dizine ulaşacağımız için tüm dosya ve klasörleri listeleyeceğiz her üst dizine çıkmak için nokta sayısı arttırılmalıdır ve toString() ile de istediklerimizin metin olarak ekrana bastırılmasını istiyoruz.

```
"use strict";

var express = require("express");
var favicon = require("serve-favicon");
var bodyParser = require("body-parser");
var session = require("express-session");
// var csrf = require('csurf');
var consolidate = require("consolidate"); // Templating library adapter for Express
var swig = require("swig");
// var helmet = require("helmet");
var MongoClient = require("mongodb").MongoClient; // Driver for connecting to MongoDB
var http = require("http");
var marked = require("marked");
//var helmet = require("helmet");
//var nosniff = require('dont-sniff-mimetype');
var app = express(); // Web framework to handle routing requests
var routes = require("./app/routes");
var config = require("./config/config"); // Application config properties
/*
// Fix for A6-Sensitive Data Exposure
// Load keys for establishing secure HTTPS connection
var fs = require("fs");
var https = require("https");
var path = require("path");
var httpsOptions = {
    key: fs.readFileSync(path.resolve(__dirname, "./artifacts/cert/server.key")),
    cert: fs.readFileSync(path.resolve(__dirname, "./artifacts/cert/server.crt"))
};
*/

MongoClient.connect(config.db, function(err, db) {
    if (err) {
        console.log("Error: DB: connect");
        console.log(err);

        process.exit(1);
    }
    console.log("Connected to the database: " + config.db);

    /*
    // Fix for A5 - Security MisConfig
    // TODO: Review the rest of helmet options, like "xssFilter"
    // Remove default x-powered-by response header
    app.disable("x-powered-by");
```

Önceki aşamada ana dizindeki dosya ve klasörlerin isimlerine ulaşmıştık bu safhada ise dosyaların içeriğine nasıl ulaşıldığı hakkında fikir sahibi olacağız. res.end(require('fs').readFileSync('dosyaismi')) kodu sayesinde bu işlemi gerçekleştireceğiz. Ana dizinde server.js isimli bir dosya var onu okuyalım, bunun için kodu res.end(require('fs').readFileSync('./server.js')) olarak düzenliyoruz burada okumak istediğimiz dosyanın başına ./ koyarak bir üst dizine çık ve oradan server.js dosyasını bize getir komutunu verdik aslında.

Şimdi de veritabanı ile sunucu arasındaki etkileşimin sağlanması için gerekli olan ve kötü niyetli bir kişi tarafından ele geçirilmesi hâlinde ciddi zararlara sebebiyet verecek config dosyasını okumaya çalışalım, bu sunucuda config dosyası config/env/all.js yolunda barınıyor. Dosya okuma kodumuzu buna göre düzenlersek: res.end(require('fs').readFileSync('./config/env/all.js')) üst dizine çıkarak config klasörünün altındaki env klasörü ve onun da altındaki all.js dosyasını bize getir dedik.

```
// default app configuration
module.exports = {
    port: process.env.PORT || 4000,
    db:  process.env.MONGOLAB_URI || process.env.MONGODB_URI || "mongodb://nodegoat:owasp@ds159217.mlab.com:59217/nodegoat",
    cookieSecret: "session_cookie_secret_key_here",
    cryptoKey: "a_secure_key_for_crypto_here",
    cryptoAlgo: "aes256",
    hostName: "localhost"
};
```

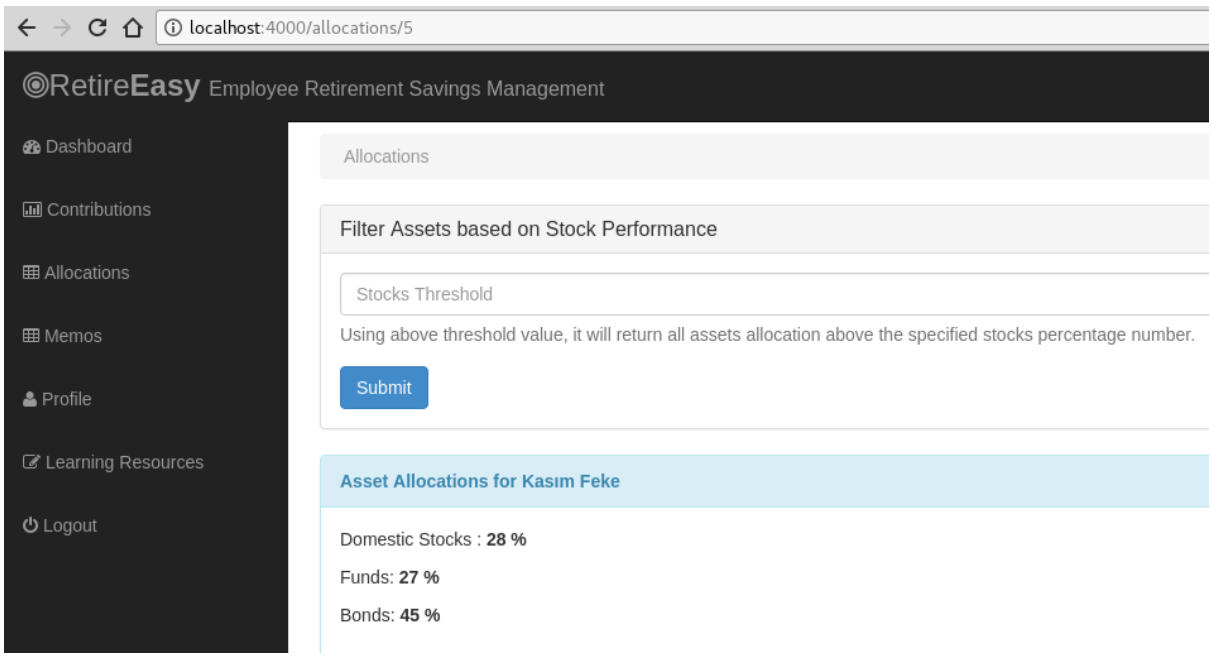
Sonuç başarılı. Bu şekilde dizinler arasında geçişler yapılarak farklı dosyalar da okunabilir. Peki ne şekilde önlem alınabilir ondan bahsedelim.

```
24          var preTax = eval(req.body.preTax);
25          var afterTax = eval(req.body.afterTax);
26          var roth = eval(req.body.roth);
```

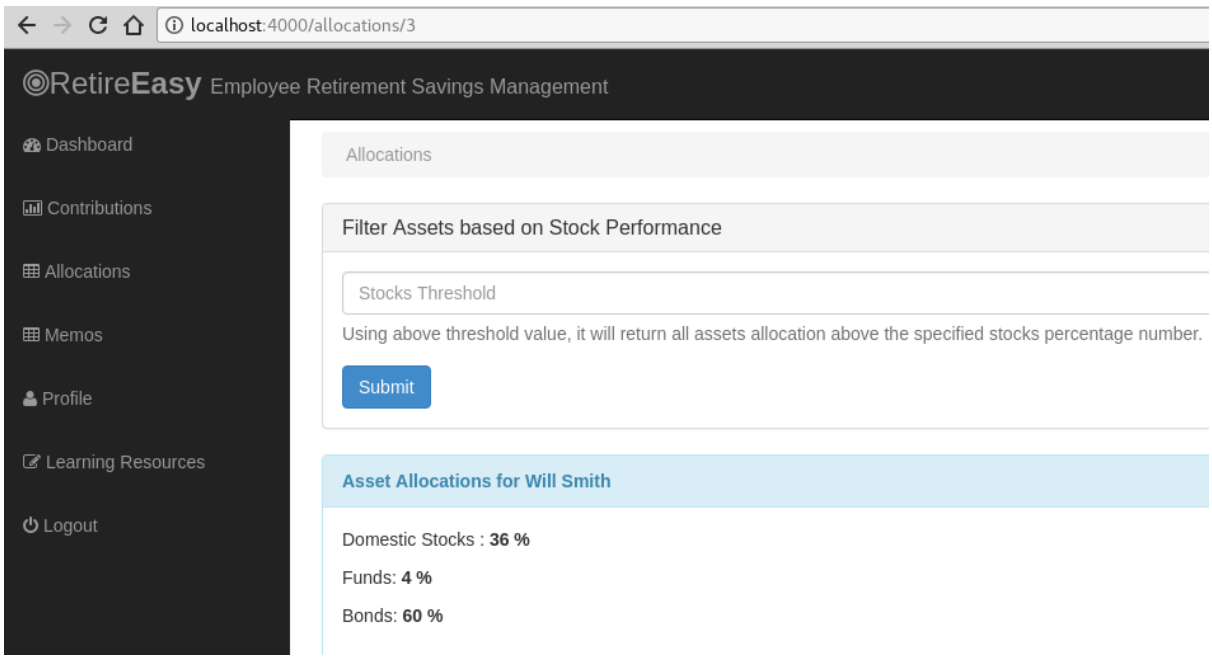
Zafiyetin contributions sayfasında var olduğunu biliyoruz, app/routes/contributions.js bu sayfanın kaynağıdır contributions.js dosyasını incelediğimizde 24, 25 ve 26. satırlardaki kod parçaları dikkatimizi çekiyor. Çünkü burada eval metodu kullanılarak input ile alınan veri direkt olarak JavaScript kodlarına çevriliyor, normalde burada alınan verinin tam sayıya dönüştürülmesi gerekmektedir bundan dolayı eval yerine parseInt metodu kullanılmalıdır. Burada beklenen değer bir sayı (integer) değil de string (metin) olsaydı önlem olarak RegEx (Düzenli ifadeler) veya input validation (girdi kontrolü) kullanılması daha mantıklı olacaktı.

Insecure Direct Object References

Bu zafiyet, geliştiricilerin yetki kontrolü yapmaması durumunda ortaya çıkar. Örnek üzerinden uygulamalı şekilde bunun nasıl gerçekleştiğini görelim.

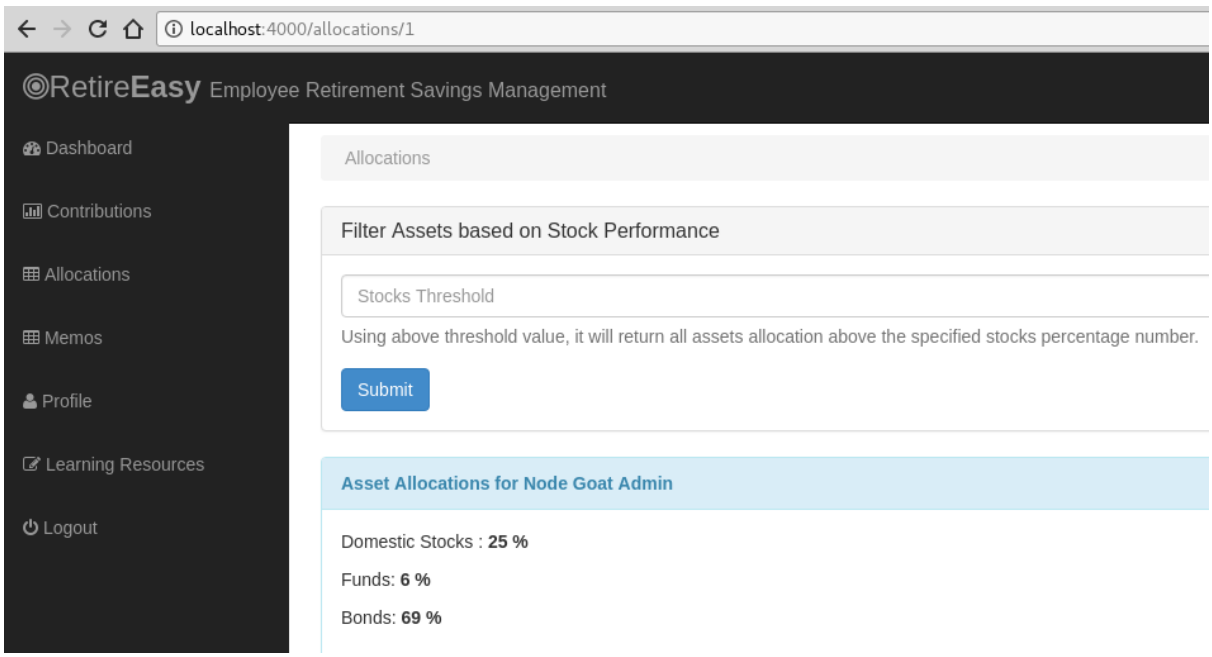


Insecure Direct Object References zafiyetinin bulunduğu Allocations sayfasında sisteme kayıtlı kişilerin varlıkları hakkında bilgiler mevcut, URL incelendiğinde sayfa isminden sonra bir sayı görüyoruz işte bu sayı giriş yapılan kullanıcının veritabanında kayıtlı olduğu ID değeridir bunu sıra numarası olarak da düşünebiliriz örneğin Kasım Feke kullanıcısından sonra bir tane daha kullanıcı kayıt edilse onun ID değeri de 6 olacaktır. Buradan hareket ile daha önce kayıt edilen kullanıcıların ID değerlerinin de 1,2,3,4,5 olduğunu tahmin etmek pek zor olmuyor. Şuna da değinmekte fayda var ID değeri 1 olan kullanıcılar genellikle süper kullanıcı olarak tabir edilen yani yönetici haklarına sahip olanlardır. Şimdi ID değerlerini değiştirerek farklı kullanıcıların verilerini okumaya çalışalım.



ID değeri 3 olan Will Smith isimli kullanıcının verilerine ulaştık, peki adminin verilerine erişebilecek miyiz onu deneyelim.

Başarılı bir şekilde Node Goat Admin'in de verilerine erişmiş olduk.



Nasıl bir önlem alınabileceğine bakalım.

```
1  var AllocationsDAO = require("../data/allocations-dao").AllocationsDAO;
2
3  function AllocationsHandler(db) {
4      "use strict";
5
6      var allocationsDAO = new AllocationsDAO(db);
7
8
9      this.displayAllocations = function(req, res, next) {
10         /*
11         // Fix for A4 Insecure DOR -  take user id from session instead of from URL param
12         var userId = req.session.userId;
13         */
14         var userId = req.params.userId;
15
16         allocationsDAO.getByUserIdAndThreshold(userId, req.query.threshold, function(err, docs) {
17             if (err) return next(err);
18
19             docs.userId = userId; //set for nav menu items
20
21             return res.render("allocations", docs);
22         });
23     };
24 }
25
26 module.exports = AllocationsHandler;
```

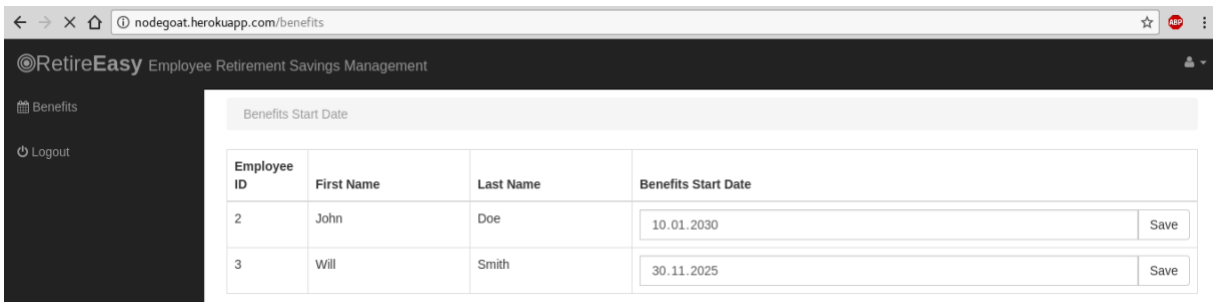
app/routes/allocations.js kaynak kodunun 14. satırında görüldüğü üzere parametreden gelen değer doğrudan userId değerine atanmaktadır. Ancak bu kısım;

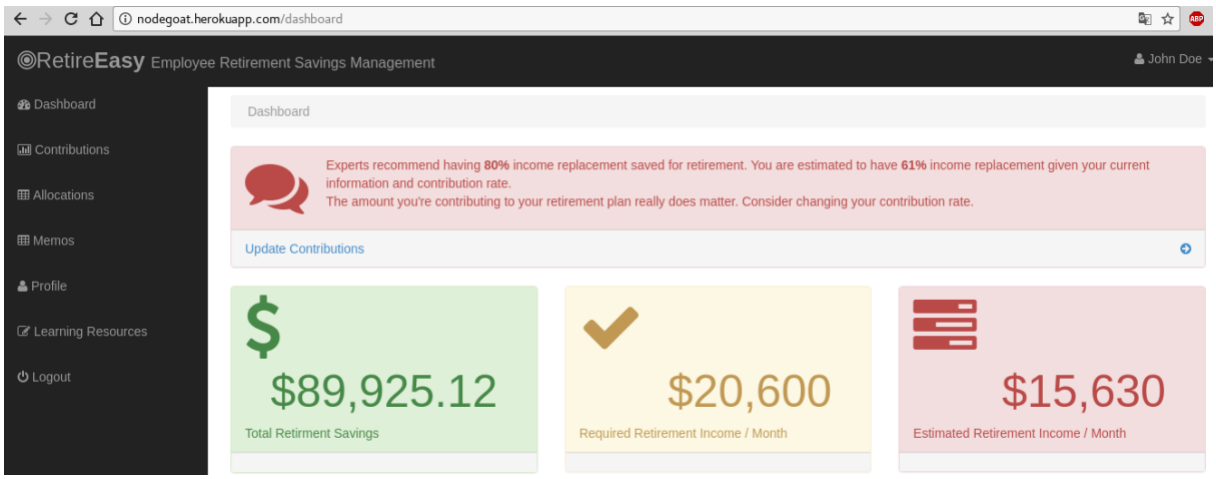
if(req.params.userId==req.session.userId)
userId=req.params.userId
else return 0;

olarak düzenlenir ise session değeri içerisindeki userId ile parametreden gelen değer karşılaştırıldıktan sonra atama işlemi yapılarak zafiyet önlenebilir.

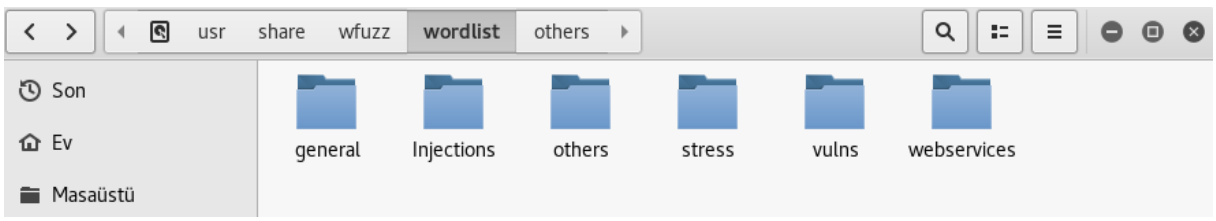
Missing Function Level Access Control

Bu zafiyet yöneticiler dışında erişilmemesi gereken sayfalara sıradan kullanıcıların eksik erişim kontrolü yapıldığı durumlarda ortaya çıkmaktadır, uygulamalı şekilde nasıl gerçekleştiğini görelim.

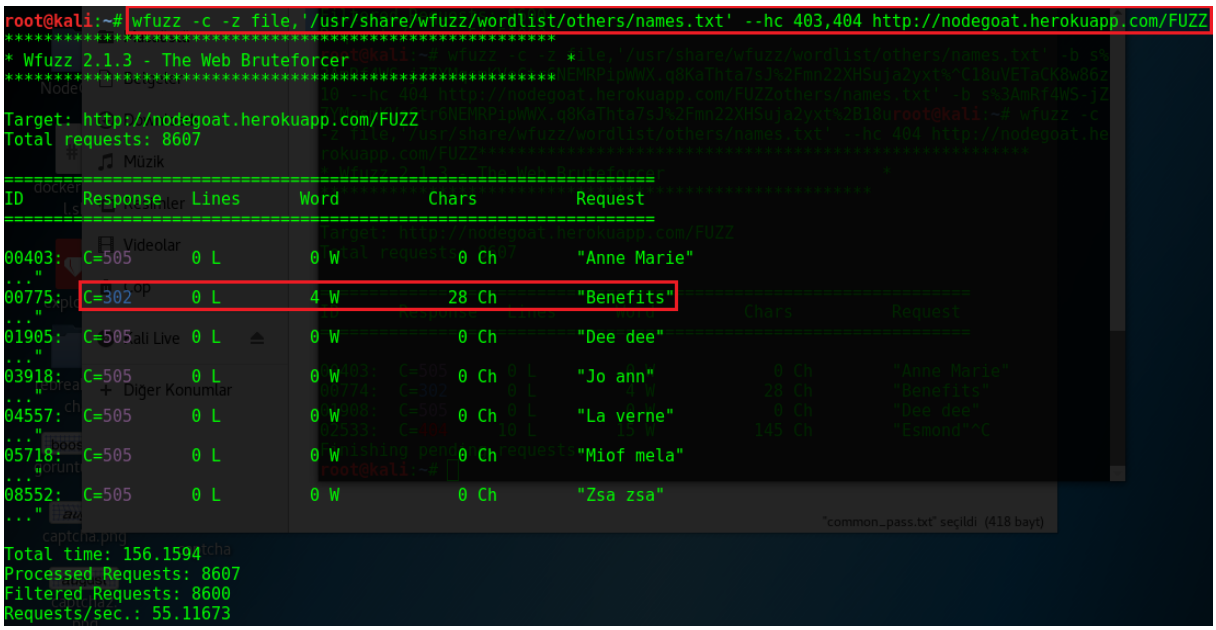




İlk ekran görüntüsü admin hesabından, ikincisi ise user1 kullanıcısının hesabından alınmıştır. Herhangi bir kullanıcı ile giriş yapıldığında Benefits isimli sayfa menüde yer almıyor bu sayfada kullanıcıların emeklilik tasarruf yönetiminden ne zaman faydalanmaya başladıkları bilgisi mevcut artı olarak bu tarihler değiştirilebiliyor, güvenliğin sağlanması için bu sayfaya hiçbir şekilde kullanıcılar tarafından erişilmemesi gerekiyor fakat eksik erişim kontrolü yapıldığında sayfa ismi bilindiği, tahmin edildiği veya wfuzz dirb gibi araçlar kullanılarak atak yapıldığı durumda sayfaya erişim sağlanabiliyor ayrıca yetki kontrolü yapılmayan sayfaların arama motorlarında listelenmesi de söz konusudur. Wfuzz aracı üzerinden deneme yapalım.



Kali Linux'da wfuzz kurulu olarak gelmektedir, farklı bir linux dağıtımı veya windows kullanıcısı iseniz https://github.com/xmendez/wfuzz.git adresinden wfuzz dosyalarını indirdikten sonra pip install wfuzz komutu ile yüklemeyi gerçekleştirebilirsiniz. /usr/share/wfuzz/wordlist dizininde 6 kategori hâlinde deneme yanılma yöntemi için kullanılabilecek kelime listeleri mevcuttur, others klasöründeki names.txt de benefits kelimesi mevcut olduğu için onu tercih ettik.

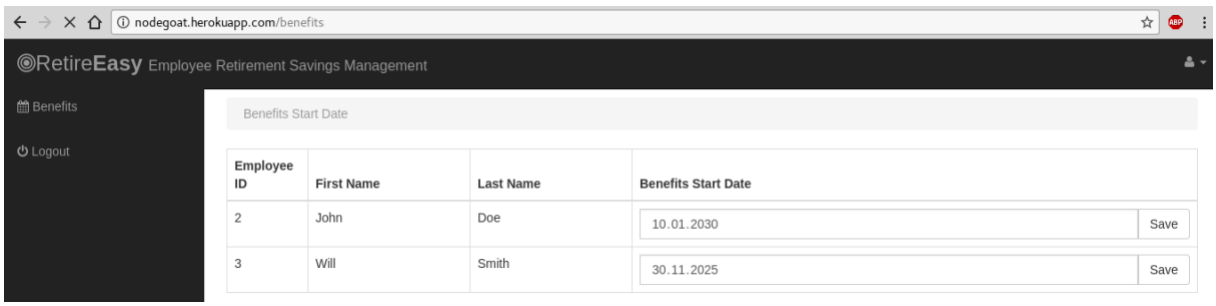


Terminalde atağa başlayabilmek için

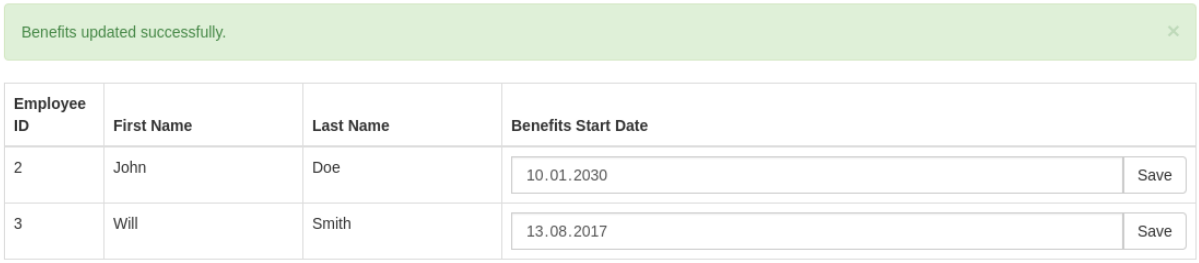
wfuzz -c -z file,'/usr/share/wfuzz/wordlist/others/names.txt' –hc 403,404 http://nodegoat.herokuapp.com/FUZZ

komutunu yazdık, -c parametresi response değerlerini renkli yapar kişisel tercihtir http durum kodlarını belirgin olarak görmek istiyorsanız -c parametresini kullanabilirsiniz, -z parametresinden sonra atakta kullanacağımız wordlistin yolunu belirtiyoruz, --hc parametresinden sonra ise aşağıda hangi durum kodlarına ait çıktıları görmek istemiyorsak onları yazabiliriz burada 403,404 yazarak yasaklanan ve var olmayan sayfaları çıktıda görmek istemediğimizi belirttik 505 de dahil edilebilirdi, en son ise atak yapılacak sitenin adresini ve kelimelerin nerede deneneceğini FUZZ ile belirtiyoruz.

Dönen response (cevaplar) incelendiğinde benefitsin 302 durum kodu yani yönlendirilmiş sayfa olduğunu görüyoruz.



Dashboard yerine benefits yazdığımızda benefits sayfasına erişmiş olduk. Mesela Will Smith'in tarihini 13.08.2017 olarak değiştirerek save butonu ile kaydedelim ve veritabanına işlenecek mi görelim.



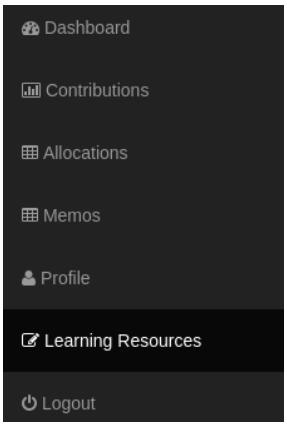
Başarılı bir şekilde tarih 13.08.2017 olarak güncellendi. Bu zafiyete ne gibi önlem alınabilir ondan bahsedelim.

```
52    // Benefits Page
53    app.get("/benefits", isLoggedIn, benefitsHandler.displayBenefits);
54    app.post("/benefits", isLoggedIn, benefitsHandler.updateBenefits);
```

app/routes/index.js 53 ve 54. satırlarında yetki kontrolü yapılmadığı anlaşılıyor isLoggedIn den sonra isAdmin eklenir ise zafiyet önlenmiş olacak ve Benefits sayfasına admin dışında erişim sağlanmayacaktır.

Unvalidated Redirects and Forwards

Bu zafiyet, web uygulamalarında kullanıcıların başka sayfalara veya web sitelerine yönlendirilirken bu yönlendirmelerin kontrol altına alınmaması durumunda ortaya çıkmaktadır. Doğrulanmamış yönlendirmeler ile mevcut uygulama parametreleri değiştirilerek uygulamanın farklı içeriklere yönlendirilmesi sağlanabilmektedir. Zararlı yazılım bulaştırma veya sosyal mühendislik yöntemleri kullanılarak çeşitli saldırılar gerçekleştirilebilir.



Panel menüsünde Learning Resources aracılığı ile farklı bir web sitesine yönlendirme yapılıyor, bağlantı adresi kopyalandığında:

http://localhost:4000/learn?url=https://www.khanacademy.org/economics-finance-domain/core-finance/investment-vehicles-tutorial/ira-401ks/v/traditional-iras

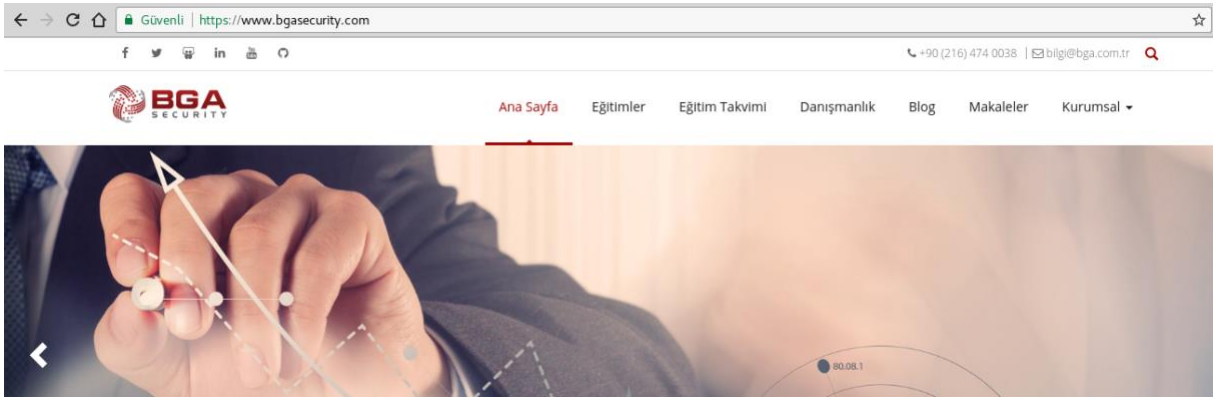
olduğunu görüyoruz.

url parametresinden sonra yönlendirilmek istenen web sitesinin veya yerel kaynakda bulunan sayfanın adresi bulunmaktadır, örnek olarak

URL http://localhost:4000/learn?url=https://www.bgasecurity.com şeklinde düzenlendiğinde şayet https://www.bgasecurity.com adresine yönlendirilir isek Unvalidated Redirects and Forwards zafiyetinin varlığı tespit edilecektir.

Görüldüğü üzere bgasecurity.com'a yönlendirildik.

http://localhost:4000/learn?url=https://www.bgasecurity.com



Önlem için alınması gereken en güzel yöntem bir whitelist oluşturularak bu liste dışındaki adreslere erişim sağlanmaması olacaktır.

```
var whiteList = ['google.com','bgasecurity.com'];
var isAllowed = whiteList.some(piece => piece === req.query.url )
isAllowed ? res.redirect(req.query.url) : res.send('izinsiz url')
```

Örnek olarak app/routes/index.js dosyasına bu kod parçacığı eklenerek google.com ve bgasecurity.com dışındaki adreslere yönlendirme yapılmaması sağlanabilir.

SONUÇ

AngularJS ile Node.js'de ortaya çıkan zafiyetler ve bu zafiyetlere karşı alınabilecek önlemlerden bahsettik. Kötü niyetli kişiler saldırı girişiminde bulunmadan önce zafiyetler tespit edilerek giderilir ise büyük fayda sağlanmış olunacaktır.

BGA Bilgi Güvenliği A.Ş. Hakkında

BGA Bilgi Güvenliği A.Ş. 2008 yılından bu yana siber güvenlik alanında faaliyet göstermektedir. Ülkemizdeki bilgi güvenliği sektörüne profesyonel anlamda destek olmak amacı ile kurulan BGA Bilgi Güvenliği, stratejik siber güvenlik danışmanlığı ve güvenlik eğitimleri konularında kurumlara hizmet vermektedir.

Uluslararası geçerliliğe sahip sertifikalı 50 kişilik teknik ekibi ile, faaliyetlerini Ankara ve İstanbul ve USA'da sürdüren BGA Bilgi Güvenliği'nin ilgi alanlarını "*Sızma Testleri, Güvenlik Denetimi, SOME, SOC Danışmanlığı, Açık Kaynak Siber Güvenlik Çözümleri, Büyük Veri Güvenlik Analizi ve Yeni Nesil Güvenlik Çözümleri*" oluşturmaktadır.

Gerçekleştirdiği başarılı danışmanlık projeleri ve eğitimlerle sektörde saygın bir yer edinen BGA Bilgi Güvenliği, kurulduğu günden bugüne alanında lider finans, enerji, telekom ve kamu kuruluşlarına 1.000'den fazla eğitim ve danışmanlık projeleri gerçekleştirmiştir.

BGA Bilgi Güvenliği, kurulduğu 2008 yılından beri ülkemizde bilgi güvenliği konusundaki bilgi ve paylaşımların artması amacı ile güvenlik e-posta listeleri oluşturulması, seminerler, güvenlik etkinlikleri düzenlenmesi, üniversite öğrencilerine kariyer ve bilgi sağlamak için siber güvenlik kampları düzenlenmesi ve sosyal sorumluluk projeleri gibi birçok konuda gönüllü faaliyetlerde bulunmuştur.

BGA Bilgi Güvenliği AKADEMİSİ Hakkında

BGA Bilgi Güvenliği A.Ş.'nin eğitim ve sosyal sorumluluk markası olarak çalışan Bilgi Güvenliği AKADEMİSİ, siber güvenlik konusunda ticari, gönüllü eğitimlerin düzenlenmesi ve siber güvenlik farkındalığını arttırıcı gönüllü faaliyetleri yürütülmesinden sorumludur. Bilgi Güvenliği AKADEMİSİ markasıyla bugüne kadar "Siber Güvenlik Kampları", "Siber Güvenlik Staj Okulu", "Siber Güvenlik Ar-Ge Destek Bursu", "Ethical Hacking yarışmaları" ve "Siber Güvenlik Kütüphanesi" gibi birçok gönüllü faaliyetin destekleyici olmuştur.