Penetration testing izvještaj – Tim 9

BSEP 2022

# Tim

* Dalibor Malić, SW-50/2018
* Dimitrije Karanfilović, SW-39/2018
* Ivan Mršulja, SW-65/2018

# Korišćeni alati

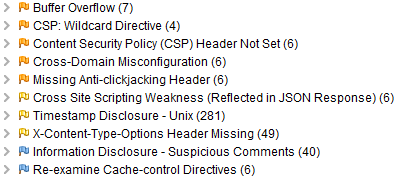
Za realizaciju dodatnog zadatka (penetration testing) odlučili smo se na korišćenje 2 alata:

1. OWASP ZAP
2. Nikto v2.1.6

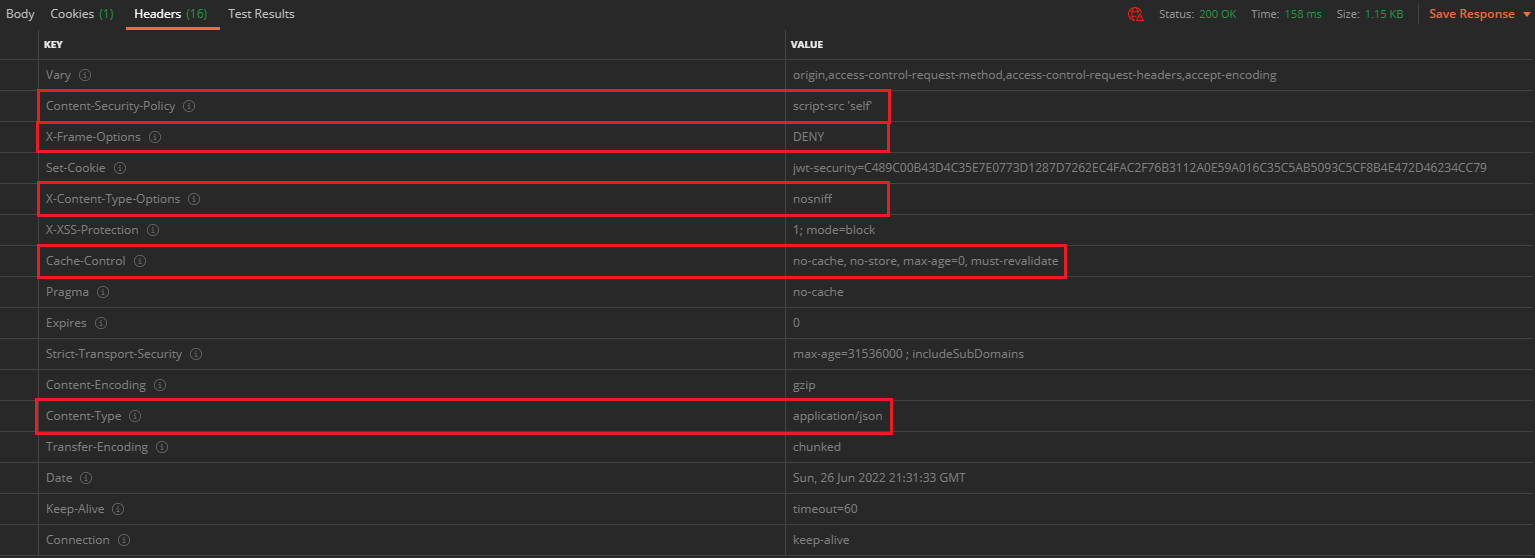
Kao bonus, „konsultovali“ smo i *Github Dependabot*-a kako bi dobili obavještenja o eventualnim ranjivostima u zavisnostima koje naš projekat koristi.

# OWASP ZAP

U skladu sa najboljim praksama za korišćenje ovog alata, za testiranje naše aplikacije korišćena je kombinacija automatskog i ručnog testiranja [3]. Testirane su i „Admin“ i „Moja Kuća“ aplikacije. Za testiranje stranica za čiji pristup nije trebala autentifikacija, korišćen je automatski režim rada sa tradicionalnim i *AJAX spider*-om, za testiranje ostalih stranica korišćen je manuelni režim rada gdje smo se ulogovali ručno a zatim upalili *attack mode* koji je *AJAX spider*-om istestirao sve stanice kojima je imao pristup (ručno smo se prebacivali sa stranice na stranicu i čekali da izvrši testiranje). Alat je pronašao sledeće ranjivosti (prikazane su ranjivosti za „Admin“ i „Moja Kuća“ aplikacije zajedno):

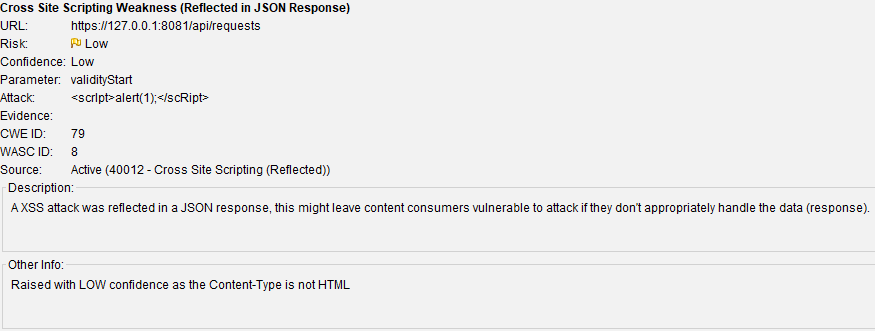


Bitno je naglasiti da je maksimalna *severity* za ranjivost koju je alat pronašao bila MEDIUM. Što se tiče ranjivosti vezanih za nedostajuće *header*-e bilo nam je čudno s obzirom da smo 100% bili sigurni da smo ih dobro konfigurisali u *SpringSecurity*-u. Kratkim testom svih *endpoint*-a iz *postman*-a uvjerili smo se da su svi headeri koji su označeni kao nedostajući zapravo bili tu i da je njihova vrijednost trenutno postavljena u skladu sa najboljim praksama u šta smo se uvjerili jer nam je i sam alat predlagao iste vrijednosti za *header*-e.



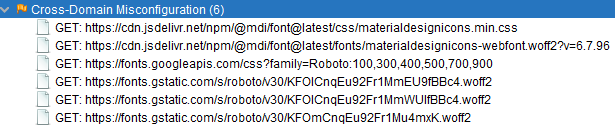
Takođe, iz tabele [1] možemo vidjeti da su upravo ove ranjivosti najviše bile označene kao *false positive* u posljednjih mjesec dana što nas je navelo da ih i mi tretiramo na isti način.

*Buffer overflow i XSS Weakness* su druga priča. Nedostatak validacije za polja gdje se unosio datum dovelo je do toga da ukoliko se za datum unese neki pogrešan string server vrati 500 i *error message* u formatu *stack trace*-a. S obzirom da je server vracao 500 kod unosa ogromnog *string*-a detektovan je *buffer overflow* a pošto je u povratnoj poruci o grešci za 1 primjer pisalo „JSON parse error: Cannot deserialize value of type `java.util.Date` from String \"<scrIpt>alert(1);</scRipt>\"“ Detektovan je *false positive XSS* sa *low confidence score*-om.



Iako ozbiljnost problema nije bila velika, ubacili smo *error handler*-e za slučaj da se desi *JSON Parse Error* tako da je ovaj problem riješen. Pronađene su i *buffer overflow* ranjivosti gdje se u tekstualna polja, prilikom unosa ogromnog stringa, vracao kod 500 jer su sve validacije za dužinu stringa bile na nivou baze podataka. Ubacivanjem *error handler*-a za ovaj tip *runtime* izuzetka ovaj problem je riješen.

*Timestamp Disclosure* ranjivost je bila evaluirana i javljala se jer smo u JS fajlovima imali zakucano da se prilikom dobavljanja svih entiteta (koji podržavaju filtriranje po *timestamp*-u) po *default*-u zada opsjeg od 1970 do 3970 godine ukoliko korisnik nije ništa unio. Analizom koda shvatili smo da nam taj *timestamp disclosure* ne predstavlja problem te smo ostavili to kako jeste. Što se *information disclosure*-a tiče, imali smo komentare i console.log-ove koji su sadržali „pipave“ informacije (kad se šta dobavlja, parametri zahtjeva, uloge koje mogu pristupiti stranicama, obavještenja da se nešto desilo...) a koje smo zaboravili da uklonimo te smo ih uklonili i tako riješili problem. *Cross Domain Misconfiguration* se nije odnosio na našu aplikaciju već na resurse koji su javno dostupni tako da tu ništa nismo dirali.

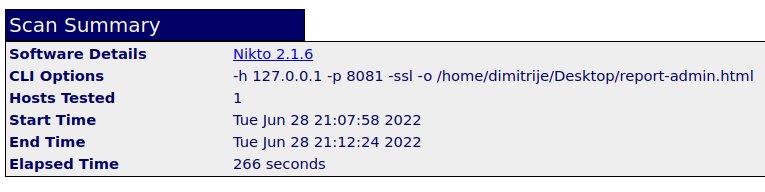


# Nikto

*Nikto* je takođe pokrenut na obje aplikacije. Imajući u vidu da je za obje aplikacije podešen SSL, *nikto* nije dobavio mnogo informacija o samom serveru (u našem slučaju o lokalnoj mašini).

## Admin aplikacija

Opšte informacije o skenu se mogu vidjeti na sledećoj slici

Kao upozorenja, prijavio je sledeće stvari:

* dozvoljenu upotrebu PUT HTTP metode
* dozvoljenu upotrebu DELETE HTTP metode
* nepodudaranje naziva host-a sa imenom sertifikata
* nedostatak X-Content-Type-Options HTTP zaglavlja
* nedostatak Strict-Transport-Security HTTP zaglavlja
* nedostatak X-Frame-Options HTTP zaglavlja

Nepodudaranje naziva hosta i imena sertifikata se dalo I оčekivati, imajući u vidu da je lokalna mašina u pitanju, kao I ručno generisani *self-signed* sertifikat. Što se tiče zaglavlja, ona su već prisutna, što je objašnjeno u poglavlju OWASP ZAP (u *screenshot*-u se nalazi i *Strict-Transport-Security* koje u tom poglavlju nije pomenuto).

Na kraju, i PUT i DELETE su neophodne zbog poštovanja REST principa, tj. posao koji odrađuju funkcije koje gađaju ovim metodama im semantički odgovata.

## Moja kuća

Opšte informacije o skenu se mogu vidjeti na sledećoj slici

# 

Prijavljeni nedostaci su u najvećem broju isti kao kod adminske aplikacije, sa 2 izuzetka:

1. Sadržaj *Content-Encoding* zaglavlja je „deflate”
2. Nepoznato zaglavlje „content-disposition” sa sadržajem inline;filename=f.txt

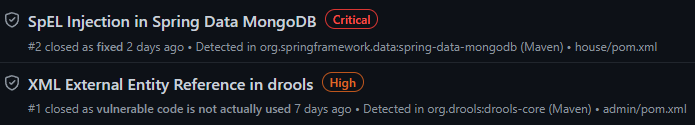
# 

Što se tiče prve stavke, nakon ponavlja zahtjeva zahtjeva iz *Postman-a,* vidi se da je sadržaj datog zaglavlja „gzip“.

Ova druga stavka je nađena na relativnoj putanji /Zo1EyHOR.dll, koja ne znam šta predstavlja.

# Github Dependabot

*Github dependbot* prijavio je 2 ranjivosti:



*SpEL Injection* ranjivost predstavljala je ogroman problem s obzirom da smo vršili filtriranje po tekstualnom sadržaju i da je bilo vrlo lako izvršiti ovaj napad. *Update*-ovanjem zavisnosti „spring-data-mongodb“ sa 3.5.4 na 3.5.5 verziju ovaj problem je otklonjen.

*XML External Entity Reference* nije predstavljao problem s obzirom da se može javiti samo u slučaju da *Drools* bazu znanja punimo sa pravilima koja su zadata u XML formatu [2].

# Reference

[1] - <https://www.zaproxy.org/docs/statistics/highest-false-positives-last-month/>

[2] - <https://stackoverflow.com/questions/34593918/reading-rules-from-xml-in-drools-6-3>

[3] - <https://www.zaproxy.org/getting-started/#exploring-an-application-manually>