Bezbednost - analiza ranjivosti sistema

## **JWT**

JWT ima nekolicinu mana, koje mogu da dođu do izražaja ukoliko se njegova upotreba ne izvrši na pravi način.

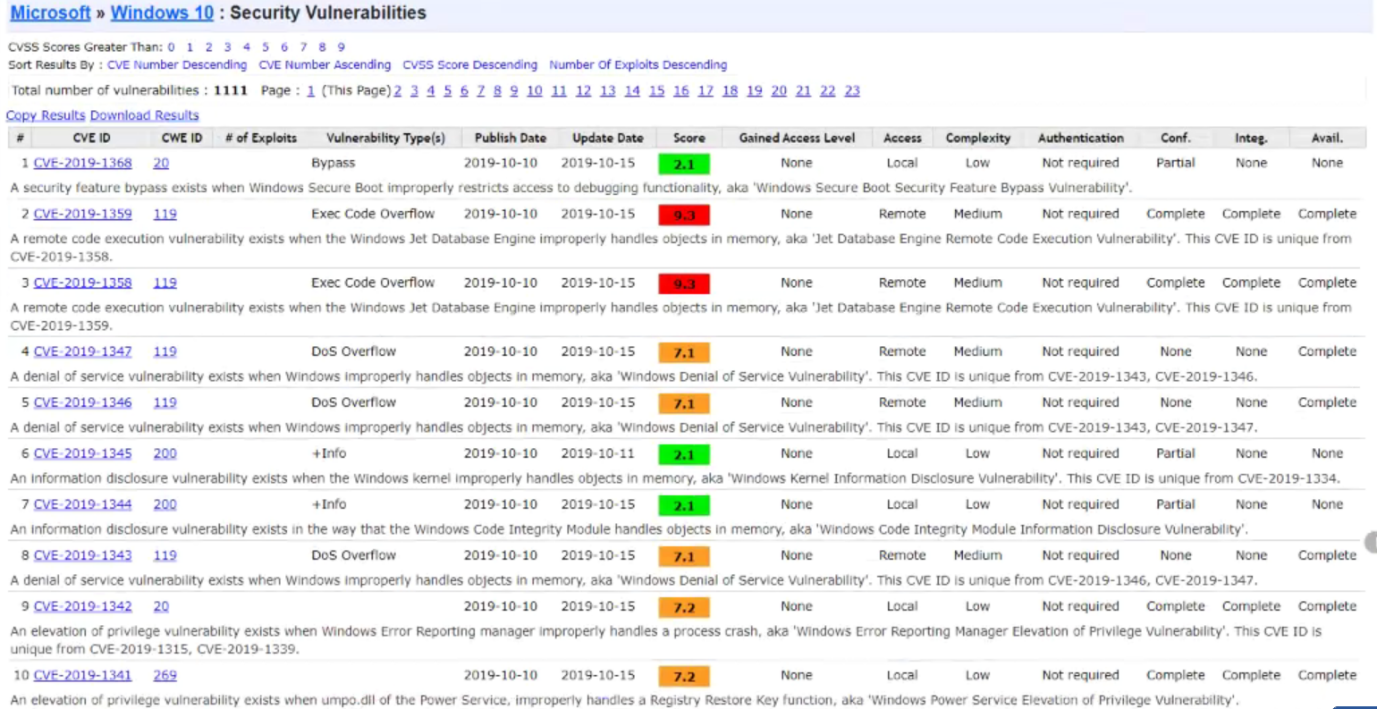
* Algoritam za kriptovanje nije eksplicitno naznačen – svaki token sa praznim poljem za potpis bi se smatrao kao validan
* Ukoliko je odabran RSA algoritam za kriptovanje, napadač može da definiše u svom JWT-u da koristi HMAC algoritam I da iskoristi naš public key za kriptovanje. Pošto mi verifikujemo potpis uz pomoć svog public key-a, taj JWT će biti validan
* Ukoliko se ne validira potpis na backend-u, a JWT se koristi u aplikaciji
* Ukoliko ključ kojim je token potpisan nije dovoljno kompleksan, napadač bi eventualno mogao sa svim informacijama koje poseduje u samom JWT-u da simulira validan potpis
* Ukoliko server na kom su skladišteni podaci o ključevima za potpisivanje tokena sadrži ranjivosti, moguće je da napadač ukrade te informacije sa servera I tim ključem potpiše neki svoj token
* Ukoliko se koristi KeyID I nije sanitizovan pre upotrebe, može da dođe do krađe podataka iz direktorijuma sa servera ili do SQL injekcija
* Ukoliko se koristi JKU, napadač može da umetne url do svog key file-a kojim će da verifikuje token
* Ukoliko se koristi JWK, napadač može da umetne svoj ključ kojim će da verifikuje token
* Ukoliko se odabere localstorage za čuvanje JWT-a, on je podložniji XSS napadima
* Ukoliko se koriste cookies za čuvanje JWT-a, on je podložniji CSRF napadima

Rešenja za koja smo se odlučili:

* Eksplicitno navodimo algoritam za kriptovanje
* Koristimo HMAC algoritam
* Potpis se validira na backend-u pri svakom zahtevu
* Pri kreiranju ključa navodimo njegovu kompleksnost
* Fizički se nigde ne skladište podaci o ključevima
* Ne koristimo Key ID
* Ne koristimo JKU
* Ne koristimo JWK
* Koristimo localstorage, ali smo ga zaštitili od XSS napada
* Ne koristimo cookies

## **OS**

Operativni sistem na kojem naša aplikacija radi je podložan nekim svojim ranjivostima, koje ne možemo da rešimo. Te probleme moraju da reše tvorci operativnog sistema, a na nama je da u svakom trenutku koristimo onu verziju koja ima najmanje takvih ranjivosti.

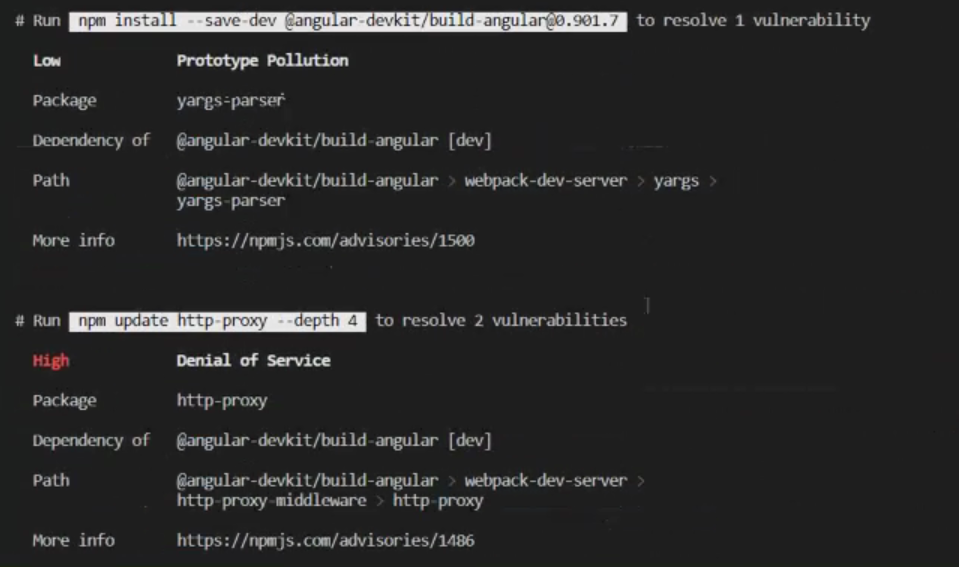


Lista ranjivosti OS Windows 10

Na gorenavedenom sajtu možemo da pratimo sve aktuelne ranjivosti na željenom operativnom sistemu.

## **ANGULAR**

Angular koristi biblioteke koje poseduju svoje ranjivosti I time čine celu klijentsku stranu ranjivom. Te ranjivosti uglavnom proističu od zastarelog instaliranog paketa određene biblioteke I rešava se ažuriranjem iste. Da bi olakšao svojim korisnicima proveru ranjivosti, uveo je komandu “npm audit” koja izlistava sve ranjivosti, njihov stepen rizika i komande kojima bi se one rešile.



Rezultat izvršenja komande “npm audit”

Pomoću komandi koje nam je Angular predložio, rešili smo sve ranjivosti koje je prepoznao.

## **XSRF TOKEN**

Pošto su cookie-i podložni CSRF napadima, naveli smo flag HttpOnly za XSRF token repozitorijum, koji govori da cookie u kom se skladišti XSRF token neće biti dostupan skriptama klijentske strane, tako da ovom cookie-u može pristupiti samo server.

## **H2**

Baza podataka koju koristimo za razvoj ima dve najveće bezbednosne ranjivosti, za koje nemamo informaciju da su ispravljene od 2018. godine.

Prva je ta što je imala problem sa obradom permisija, pa se dešavalo da napadači dođu do osetljivih podataka, iako nisu imali dozvolu za to. Ovo su uspevali pomoću „simboličnog linka“ ka lažnim fajlovima baze podataka.

Druga je ta što komanda „CREATE ALIAS“ dozvoljava izvršavanje proizvoljnog JAVA koda.

Ove ranjivosti, ako i dalje postoje, bismo svakako rešili korišćenjem neke od baza koje su namenjene za široku upotrebu.

## **JPA**

Poslednja poznata ranjivost vezana za JPA je ta da je bio podložan SQL injekcijama kada se koristi @Query anotacija i kada se na taj način izvlače podaci iz baze.

Da bismo zaobišli moguće rizike