Izveštaj

1. OWASP TOP 10 problemi

10 - Unvalidated redirects and forwards

Aplikacija prihvata input koji može izazvati redirect/forward na URL naveden u inputu.

Ukoliko nema validacije inputa, napadač može da izvrši phishing napad i ukrade korisničke kredencijale ili eksploatiše korisnika na drugi način.

Problem smo rešili tako što smo izabrali da ne koristimo redirect i forward.

9 - Using components with known vulnerabilities

Podrazumeva korišćenje komponenti sa ranjivostima. Svi napadi koje je moguće sprovesti na ranjivoj aplikaciji se mogu sprovesti i na aplikaciji koja ima ranjivu komponentu.

Naš projekat koristi:

Apache Ant 1.8.4 kod kojeg je bio problem DOS napad, ali je on resen.

Apache-Tomee Plus 1.7.4 dozvoljavao je napadaču da izvršava komande putem serijalizovanog Java toka, ali je to rešeno.

Dependency check report pokazao je da od 47 skeniranih zavisnosti naš projekat ima 4 ranjivosti:

1. Httpclient-4.1.1.jar fajl ne provera dobro da li se hostname servera slaže sa imenom domena u X.509 sertifikatu, sto omogućava napadaču da prisluškuje SSl server preko “CN=” stringa. Zamenjen je sa httpclient-4.5.2.jar kod kojeg je ovo uklonjeno.
2. Xalan-java 2.7.0 jar fajl omogućava napadačima da prođu zabrane i pristupe klasama ili eksternim resursima. Zbog toga je zamenjen sa Xalan 2.7.2 jar fajlom kod kojeg su ovi propusti uklonjeni.
3. Jackson-core 2.4.1.jar fajl za koji nismo uspeli da pronadjemo koji ima propust.
4. Jersey-apache-client-1.17.jar za koji takođe nismo uspeli da pronađemo vrstu propusta.

8 - Cross-site request forgery

Napadač formira zahteve koje aplikacija tumači kao validne zahteve od žrtve.

Napad može da rezultuje transferom novca, promenom lozinke, kupovinom proizvoda.

Posledica toga što web čitači automatski uključuju kolačić

Ovo je rešeno uvođenjem generisanog tokena kom napadač ne može da pristupi i zatim slanje tog tokena sa svakim zahtevom. Odradili smo to u paketu rs.townhall.rest.authorization u AuthorizationHandler klasi. Kod nas je implementirano tako da se token generiše svaki put kada se korisnik prijavi.

7 - Missing function level access control

Posledica toga što aplikacija nema kontrolu pristupa podacima od strane različitih vrsta korisnika. Dovodi do neautorizovanog pristupa podacima. Dešava se kada aplikacije ne rade proveru prava pristupa na serveru sa svakim pozivom funkcije. Potrebno je vršiti autentifikaciju i autorizaciju na serverskoj strani sa svakim zahtevom.

Rešeno uvođenjem RBAC-a (paket rs.townhall.rest.autorization) pomoću kojeg se vrši provera na serveru, a vršimo proveru i na frontendu.

6 - Sensitive data exposure

Nedostatak ispravne zaštite osetljivih podataka. Napadač koristi ukradene podatke za krađu identiteta, finansija ili prodaju osetljivih podataka. Potrebno je identifikovati sve osetljive podatke i mesta gde se oni nalaze i koristiti mehanizme za zaštitu podataka.

Sva komunikacija između web čitača i back end aplikacije osigurana je upotrebom HTTPS protokola. U projektu je rešeno tako što je napravljen cilj u Ant-u koji pokretanjem prelazi na HTTPS protokol. On modifikuje konfiguracione fajlove u TomEE-ju neophodne za korišćenje HTTPS protokola u komunikaciji klijenta i servera. Konfiguracioni fajlovi koji se deploy-uju na server pokretanjem ovog cilja se nalaze u folderu conf/secured.

Lozinke se čuvaju putem Hash&Salt mehanizma. Klasa koja radi heširanje nalazi se u paketu rs.townhall.security i zove se Hash i ona se poziva prilikom persist metode unutar klase UsersDaoBean u paketu rs.townhall.dao.

Prilikom slanja poruka između SGNS i IAGNS obezbeđena je poverljivost poruka. Poruke se šifruju simetričnim algoritmom i jednokratnim simetričnim ključem.Jednokratni simetrični ključ se šifruje asimetričnim algoritmom - javnim ključem primaoca poruke. Za formiranje šifrata i podataka o korišćenom jednokratnom simetričnom ključu koristi se XML Encryption standard. Enkripcija poruka je odrađena u metodi archieve klase RegulationsDaoBean u paketu rs.townhall.dao, a dekripcija unutar metode decript klase HistoricalArchieve unutar paketa rs.historicalarchieve.rest. Klase koje sadrže metode koje vrše generisanje jednokratnog ključa, dobijanje privatnog ključa, učitavanje dokumenta, čuvanje dokumenta, enkripciju i dekripciju nalaze se u paketu rs.townhall.security i zovu se EncriptionKEK, DecriptionKEK i LoadDocument. Za generisanje simetričnog kluča koristio se AES sa dužinom ključa 128, a za asimetrično kriptovanje RSA algoritam sa dužinom ključa 1024. Enkripcija se radila tako što smo generisali tajni ključ, kriptovali sadržaj elementa tajnim ključem, njega javnim ključem i kriptovani tajni ključ stavili kao KeyInfo kriptovanog elementa.

Dekripcija je podrazumevala dekriptciju tajnog ključa privatnim ključem, a zatim tim tajnim ključem vršila dekripcija podataka.

Za potpisivanje dokumenta koristio se Enveloped stil, a potpis se formirao na osnovu XML Signature standarda. Algoritam koji se koristio je RSA sa dužinom ključa 1024 i SHA-1 heš funkcija. Potpisivanje dokumenata vrši se u persist metodi klase AmendmentsDaoBean i RegulationsDaoBean paketa rs.townhall.dao, pri čemu se vrši provera potpisa. Ako on nije validan, ne postoji mogućnost da se sačuva amandman. Provera vršena tako što se umesto validnog ključa postavljao nevalidni. Skladištenje privatnih i javnih ključeva je odvojeno. Do privatnog kluča ne postoji mogućnost da se dođe ukoliko se poseduje samo sertifikat. Privatni ključ se nalazi na računaru kod korisnika. Jedina mogućnost da se do njega dođe je da se upadne u korisnikov računar.

Vršena je i provera da li se sertifikat nalazi u listi povučenih sertifikata (klase EncryptKEK i SignEnveloped, metode readCertificate). U slučaju da se nalazi, onemogućeno je potpisivanje i enkripcija. Lista povučenih sertifikata se nalazi u paketu data/security.

5 - Security misconfiguration

Potrebno je imati proverenu konfiguraciju koja neće omogućiti napade.

Ceo projekat je rađen i testiran u localhost režimu i nije izvršena nikakva provera. Test klase koje su sadržale neke informacije pomoću kojih je moglo da se dodje do nekih informacija su izbrisane. Svi konfiguracioni fajlovi su smešteni tako da im se ne može direktno pristupiti preko URL, već samo preko aplikacije koja trči na serveru.

4 - Insecure direct object references

Dešava se kada se referenca na neki interni objekat direktno prikazuje. Objekat je slog iz baze, datoteka, direktorijum,… Bez prikladne provere prava pristupa, napadač manipuliše referencom i pristupa podacima za koja nema prava.

Sve reference na neki objekat predstavljaju ključevi koji su slučajno generisani i jedinstveni i samim tim krajnji korisnik nema informaciju šta ti id-jevi predstavljaju. Kod za generisanje id-ja se nalazi u paketu rs.townhall.entity u klasi IDGenerator. Primer ID-ja je 0c1fc0d4-5eff-40ff-abdf-2a5ae98748ee.

3 - Cross-site scripting

Nastaje usled nedostatka validacije korisničkog unosa ili enkodiranja izlaza sa servera. Napadač tera žrtvin browser da izvršava maliciozan JavaScript zaobilazeći same-origin policy. Potrebno validirati svaki ulaz u sistem i vršiti blacklist ili whitelist validaciju.

Za sada nemamo implementirano.

2 - Broken authentication and session management

Dešava se kada aplikacije u bilo kojoj fazi loše rukuju sesijama i kredencijalima. Napadači eksploatišu id sesije koji je vidljiv u tranzitu ili URLu, loše mehanizme upravljanja sesijom (invalidated, timeout, recycle,...), kredencijale koji su vidljivi u tranzitu, propuste mehanizama autentifikacije (policy, storage, security question, cache,...)

U projektu smo implementirali heširanje lozinki, id sesije nije vidljiv. Umesto njega koristimo ID koji je opisan u 4- Insecure direct object references. Prilikom prikaza korisnika, sakrili smo podatke koji bi mogli da se zloupotrebe. Trebalo bi da se odradi i zaštita lozinki u vidu ograničenja minimalnog broja karaktera i onemogućavanja uzastopnog pokusavanja logovanja u određenom vremenskom razmaku.

1 – Injection

Dešava se kada se nevalidiran unos šalje interpreteru kao deo komande ili upita.

Ovo smo pokusali da izbegnemo tako što nismo koristili DOM parser, već XML šemu. Podatke koji nam se pošalju smo validirali putem nje. Takođe, nismo vršili upite za one podatke koji bi mogli da se eksploatišu, već smo im pristupali preko api-ja. U planu je implementacija filtera koji bi radio validaciju svih ulaznih podataka pored šeme.