Sistem segmentovane zaštite korisničkih podataka u Web aplikacijama

Nenad Ristić
Fakultet za računarstvo i informatiku
Univerzitet Sinergija
Bijeljina, Bosna i Hercegovina
nristic@sinergija.edu.ba

Aleksandar Jevremović, Mladen Veinović
Univerzitet Singidunum
Beograd, Srbija
ajevremovic@singidunum.ac.rs
mveinovic@singidunum.ac.rs

Sadržaj - **U ovom radu analizira se i predlaže novo rešenje problema vezanog za zaštitu korisničkih podataka. Metoda je zasnovana na korišćenju složenih i slučajnih "salt" vrednosti pri individualnom generisanju heš vrednosti za lozinku koju je uneo korisnik kao i upotreba segmentnog hešovanja za šifrovanje lozinke uz standardnu metodu hešovanja SHA512. Lozinka se nakon šifrovanja deli u dva dela i delovi se čuvaju u dve zasebne baze podataka koje se nalaze na različitim platformama.**

Ključne riječi- zaštita podataka, Web aplikacije, šifrovanje, baze podataka

I. UVOD

Jedan od sve značajnijih bezbednosnih problema u savremenoj upotrebi Weba je otkrivanje lozinki korisnika prilikom kompromitovanja servera baze podataka na kome se one nalaze. Skorašnji slučajevi kompromitovanja baza LinkedIn, Twiter ili Sony korporacije [1] pokazuju da čak i veći sistemi imaju ranjivosti koje se mogu iskoristiti za dobijanje pristupa bazi podataka. Čak i kod čuvanja heš vrednosti lozinki savremeni rečnici omogućavaju relativno brzo dolaženje do originala. Primena tzv. "soljenja" pri izračunavanju heš vrednosti takođe nije rešenje, jer se u slučaju Web aplikacije, pri kompromitovanju baze podataka, najčešće kompromituje aplikativni server na kome se nalazi korišćena "salt" vrednost.

II. RANJIVOSTI SUPB

U bazama podataka skladište se brojne informacije iz svih mnogih domena. Različiti programi, različite namene zahtevaju različite informacije, a one se skladište u bazama podataka. Skladištenje i čuvanje podataka je glavna namena SUBP, podataka koji mogu biti izuzetno vredni i tajni zbog toga za tim sistemima raste zanimanje zlonamernih napadača, a samim time i potreba da ih se učini sigurnijim. Osim velike količine informacija koje čuvaju, postoji još nekoliko faktora koji doprinose ranjivosti baza podataka. Uz današnje trendove Interneta, SUBP-ovi koji su tradicionalno bili smešteni u zatvorene mreže i iza zaštitnih barijera, postaju sve otvoreniji prema udaljenim korisnicima, a time i sve podložniji napadima. Takođe veliki faktor predstavlja javna dostupnost programskih paketa poznatih SUBP, je postalo vrlo lako pribaviti programske pakete popularnih SUBP-ova, što zlonamernim korisnicima daje mogućnost istraživanja i pronalaženja sigurnosnih propusta.

Koncept bezbednosti u bazama podataka sličan je bezbednosti u računarskim mrežama. U oba slučaja nastoji se da se korisniku daju samo neophodne privilegije da bi se time smanjila ranjivost sistema, onemogućavanje nepotrebnih funkcionalnosti, obavezno odobravanje izmjena i nadzor pristupa, odvojiti funkcionalne blokove, obavezna upotreba šifrovanja, itd. Jedina stvarna razlika je u tome što kod baza podataka svi ovi mehanizmi djeluju unutar samog SUBP-a.

Ranjivosti baza podataka mogu proizaći iz neispravne konfiguracije SUBP-a, programskih propusta ili sigurnosnih nedostataka unutar aplikacija povezanih s njima. Iako SUBP često ne podržavaju sigurnosne mogućnosti tradicionalno prisutne kod drugih sistema, pravilno postavljanje postojećih mogućnosti može podići nivo sigurnosti podataka te ukloniti veliki broj ranjivosti. SUBP uglavnom nemaju mogućnosti zaštite korisničkih naloga koje su prisutne kod operativnih sistema. Tu se prvenstveno misli na nedostatak kontrole lozinki provjerama u rečniku i na nemogućnost određivanja perioda aktivnosti korisničkog naloga.

Često se tokom instalacije SUBP-a izvorno postavljeni i opšte poznati korisnički nalozi i lozinke ostaju aktivni bez promjene. Na području upravljanja bazama podataka nema uloge administratora za sigurnost. Zbog toga administratori baza podataka moraju voditi računa o korisničkim nalozima i lozinkama, u isto vrijeme osiguravajući ispravan rad i zadovoljavajuće performanse baze podataka. Takva situacija, uz to što otežava posao administratorima, onemogućava efikasno upravljanje ljudskim resursima. Evidentiranje događaja na SUPB najčešće se formira tako da je visokih performansi a sa što manjom upotrebom prostora diskovima. Zbog ovakvih podešavanja otežava se utvrđivanje odgovornosti kao i potencijalna forenzička analiza baze. Takođe sigurnosni propusti u SUPB predstavljaju veliku ranjivost u SUPB i omogućavaju zlonamernom korisniku izvođenje napada uskraćivanjem usluge (*eng. DoS - Denial of Service*) ili izvršavanje zlonamernog koda.

Činjenica da se baze podataka nalaze iza zaštitne barijere ne čine ih potpuno sigurnim. Napad umetanjem ili ubrizgavanjem (eng. *SQL injection*) predstavlja najčešći napad na bazu podataka. Ovaj napad nije direktan napad na bazu, već predstavlja pokušaj izmjene parametara koji se šalju Web aplikaciji sa namerom da se izmeni sql upit koji će se poslati bazi podataka.

Napad umetanjem najprimenjiviji je u procesu prijavljivanja korisnika na neku Web lokaciju [2]. Korisnik unosi svoje podatke za verifikaciju, korisničko ime i lozinku, pomoću kojih se kreira upit koji proverava tabelu u bazi sa korisničkim podacima. Ako se u tabeli pronađu podaci korisnik postaje autorizovan.

Ako je korisničko ime Nenad a lozinka test123 upit će izgledati

```
SELECT*
FROM Korisnici
WHERE KorisnickoIme=’Nenad’ AND Lozinka=’test123’
```

Napadač može umesto lozinke upisati niz karaktera, niz završiti jednostrukim navodnikom te dodati logički izraz koji je uvek tačan i kao odgovor dobija poverljive tabele iz baze. Upisivanjem umesto lozinke niza karaktera Aa' OR 'A'='A sql upit postaje

```
SELECT*
FROM Korisnici
WHERE KorisnickoIme=’Nenad’ AND Lozinka=’Aa’ OR ’A’=’A’
```

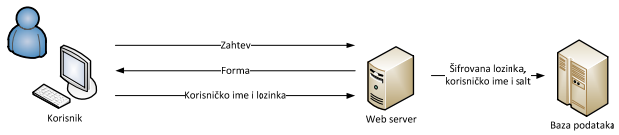
Ovakav upit će vratiti podatke svih korisnika pa čak i administratorskog naloga. Još veću opasnost predstavlja mogućnost izvršavanja više upita pa unošenjem izraza a'; DROP TABLE Korisnici; SELECT * FROM KorisniciInfo WHERE 't' = 't umesto korisničkog imena dobija se upit

```
SELECT * FROM Korisnici WHERE ime = 'a'; DROP TABLE Korisnici; SELECT * FROM KorisniciInfo WHERE 't' = 't';
```

Ovakav upit će obrisati tabelu Korisnici i prikazivanje svih podataka iz tabele KorisniciInfo. Svi SUBP-ovi sadrže ranjivosti i nije moguće odrediti koji je najsigurniji niti koji je najranjiviji među njima. Jedino je sa sigurnošću moguće tvrditi a to je da je najsigurniji onaj sistem koji se najbolje poznaje. Dobro poznavanje arhitekture i funkcionalnosti sistema od strane administratora, omogućuje siguran rad SUPB.

III. MEHANIZMI ZAŠTITE PODATAKA U UPOTREBI

U ovom poglavlju biće analizirani mehanizmi zaštite korisničkih podataka koji se trenutno koriste kroz primer Web aplikacije za registrovanje i logovanje.



Slika 1. Klasični system zaštite korisničkih podataka

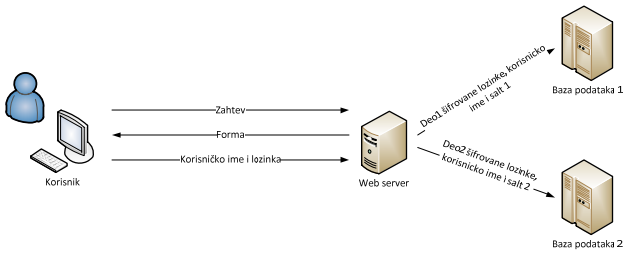
Tokom procesa registrovanja korisnik unosi željeno korisničko ime i lozinku. Upotrebom npr. php programskog koda korisničko ime i lozinka se skladište u bazu podataka. Nakon unosa podataka u Web formu lozinka se šifruje i kreira se upit bazi podataka kojim se podaci smeštaju u bazu. U nastavku je primer dela koda potreban za šifrovanje kao i generisanje upita za bazu.

```
<?php
$lozinkahash = sha1($_POST['lozinka']);
$sql = 'INSERT INTO korisnici (korisnickoime, lozinkahash) VALUES (?,?)';
$result = $db->query($sql, array($_POST['username'], $passwordHash));
?>
```

U ovakvom slučaju lozinka se šifruje SHA1 algoritmom zatim se skladišti u bazu podataka zajedno sa korisničkim imenom. Prilikom sledećeg logovanja korisnika koristi se sličan interfejs kao i kod za proveru korisničkih podataka. Nakon unosa podataka u formu vrši se šifrovanje unesene lozinke zatim provera pristupnih podataka sa podacima koji se nalaze u bazi. Ovakav princip registrovanja korisnika i čuvanja lozinke u šifrovanom obliku pruža određen nivo bezbednosti za podatke koji su u bazi. Osnovna slabost ovakvog sistema je u slučaju dobijanja pristupa bazi podataka napadač može napadom sirovom silom (*eng. Brute Force*) doći do odgovarajuće heš vrednosti koja predstavlja lozinku [3]. Napadač će porediti šifrovane lozinke iz baze sa listom generisanih heš vrednosti dok ne pronađe podudaranje. Ranije su ovakvi napadi zahtevali mnogo vremena pa je nivo zaštite običnim šifrovanjem lozinke sažimanjem bio dovoljan. Razvojem tehnologija došlo je do naprednih napada koji su koristili grafičke čipove za napade sirovom silom [4] kao i mogućnost zakupljivanja resursa samim time ovakav sistem postaje izuzetno nesiguran.

IV. IMPLEMENTACIJA SISTEMA SEGMENTOVANE ZAŠTITE KORISNIČKIH PODATAKA

U ovom poglavlju biće prikazana praktična primena sistema zaštite korisničkih podataka. Sistem je baziran tako da nakon unosa korisničkog imena i lozinke, lozinku šifruje sha512 algoritmom i zatim „umotava“ u dve slučajno generisane „salt“ vrednosti. Postoje dva osnovna pristupa za generisanje slučajnih brojeva korišćenjem računara: pseudo slučajni generatori brojeva (PRNG) i istinito slučajni generatori brojeva (TRNG). Svaki od navedenih pristupa ima svoje prednosti i mane [5]. Salt vrednosti se dodaju prije lozinke i posle. Nakon toga dobijeni blok se šifruje segmentnim hešovanjem. Na kraju se dobijena šifrovana lozinka deli na dva dela. Delovi lozinke se čuvaju u zasebnim bazama podataka. Baze podataka su na različitim platformama što nudi dodatni nivo bezbednosti. Pored dela lozinke čuva se i jedna od salt vrednosti, što znači da kompromitovanjem jedne baze napadač ima samo deo šifrovane lozinke kao i deo salt vrednosti.



Slika 2. Sistem segmentovane zaštite korisničkih podataka

Klasični sistemi čuvaju u jednoj bazi šifrovanu lozinku kao i salt vrednost dok ovaj sistem pored dvostrukog šifrovanja, deli šifrovanu lozinku i skladišti je u dve različite baze podataka. Baze podataka mogu biti na dva različita SUBP kao i različitim platformama operativnih sistema (Unix, Linux, Windows ..). Ovakav princip čuvanja podataka predstavlja veliku prepreku u potencijalnom napadu. Napadač bi morao da pronađe slabosti na različitim platformama da ih iskoristi kao i da nakon toga izvrši sklapanje lozinke i generisanje potpisa za poređenje sa postojećim salt vrednostima što zahteva velike hardverske resurse kao i nerealan vremenski period.

Za pokazni model korišten je php programski jezik, MYSQL SUPB na Linux platformi, MSSQL na Windows 2008 platformi. Php je izabran prije sveg jer u trenutku kada korisnik poseti php kreiranu stranicu, Web server automatski obrađuje php kod na osnovu kog određuje šta će prikazati korisniku. Sve ostalo npr. operacije sa fajlovima, promenljive, različite funkcije i operacije, ne prikazuju se korisniku već korisnik dobija generisanu HTML stranicu bez php koda. Modeli platformi kao i SUPB mogu biti zamenjeni bilo kojom drugom platformom i sistemom za upravljanje bazama uz minimalne izmene u programskom kodu i prilagođavanje upita.

Za početak kreirana je forma za registrovanje sa neophodnim poljima, ova forma preuzima podatke i prosleđuje ih skripti koja vrši obradu i kreiranje upita za baze podataka. Prilikom generisanja salt vrednosti koristi se php funkcija mcrypt_create_iv za generisanje inicijalnih vektora dužine 768 bita zatim se vrši pretvaranje u oblik za upotrebu kao salt. Kod kojim dobijamo salt 1 i salt 2 je u nastavku

```
$salt1=bin2hex(mcrypt_create_iv(768, MCRYPT_DEV_URANDOM));
$salt2 = bin2hex(mcrypt_create_iv(768, MCRYPT_DEV_URANDOM));
```

Primer jedne generisane salt vrednosti je :

ced3f282b04a63333e57946a079025be89ee7601f93cd9ea4
891da2f69ca43f58921475db91c580427bfd4b10ee43ccb94268
918af6b65037eea1a0081ff9a4b19ae0fdc0c45c4cb5e74d089a7
c7c73a661f3891c6b1bcc0c5ef2795bb85bd6d0c75dcc00674e3
5f290a52f3dfd890012730e33607cdfd05776b0c3fde814349a0a
1a1dbd5ca72f692cd21415a6b9463a49bfc29efe7872ddfacd20e
5d8b9b0031ec6bc4418f0c4edd8e4e0d2246d7ec016ca98a7082
7516408d1db401b59aaacca874729f764d6cc3bfc9d5c86cc621a
e6895fb4e239d723ea3e086abcf7b0e9fd68772ebfe7bc4e3c881
3f2e40c1391bef21ad1905148dd9e35d2ad8cac3b7fdd140057d
60bd62be57489ae9f8f3a5ee5802267a2eef414426727424a925a
07ee8626efdc823fff03452ece14422ddef7341860af7cd108bec6
9be4c7d9a6d12fee51f1fddeff018a6ecba7ccaa112234bf98ba28
c1aee6e57395c5f4d1028c02c73bb59ab6abd027f83488925984
799ad6b33c5ff025ad5c7c0096c2331ab

Na ovaj način generišu se dve različite slučajne vrednosti koje se koriste kao salt prije završnog šifrovanja lozinke. Ove salt vrednosti će biti različite za svakog novog korisnika. U sledećem koraku vrši se kreiranje bloka za šifrovanje koji sadrži salt1 vrednost zatim šifrovanu lozinku sha512 algoritmom i salt 2 vrednost.

```
$blok = $salt1 . hash ('sha512', $lozinka) . $salt2;
```

Na kraju formiranja šifrovane lozinke vršimo sažimanje/šifrovanje dobijenog bloka upotrebom ssdeep funkcije.

```
$lozinka = ssdeep_fuzzy_hash ($blok);
```

Rezultat šifrovanja je:

96:40cCjDoMAN34W4KijkSy2jaKnZwfWCB7m5nN0Wr:
l7pu4WMkJ0VZlCB7m5N0Wr

Ovako dobijena lozinka se deli na dva dela i vrši se smeštanje u baze podataka. Uz prvi deo lozinke u bazu podataka 1 skladišti se salt 1 kao i korisničko ime. Drugi deo lozinke se uz salt 2 i korisničko ime smešta u bazu podataka 2. Lozinku delimo tako što prvo određujemo dužinu a zatim delimo promenljivu u kojoj je lozinka na dva dela i smeštamo u zasebne promenljive koje će se uskladištiti u baze podataka. U nastavku je primer koda za podelu lozinke na dva dela.

```
$polovina = (int) ( (strlen($sifra1) / 2) );
$deo1 = substr($sifra1, 0, $pola);
$deo2 = substr($sifra1, $pola);
```

Nakon ovog koraka definišu se upiti i podaci se upisuju u baze podataka. Baze podataka pored toga što mogu biti na različitim platformama mogu biti i na različitim lokacijama time se dodatno umanjuje ranjivost sistema.

Kod autentifikacije korisnika koji je registrovan na osnovu korisničkog imena iz baza podataka uzimaju se salt1 i salt2 vrednosti, lozinka koju je korisnik uneo prolazi proces šifrovanja kao i prvobitna lozinka a zatim se vrši poređenje delova u bazama. Nakon uspešne provere segmenata lozinke korisnik je završio proces logovanja i uspešno je autentifikovan.

V. ZAKLJUČAK

Trenutno jedan od značajnijih bezbednosnih problema u savremenoj upotrebi Weba je otkrivanje lozinki korisnika prilikom kompromitovanja servera baze podataka na kome se one nalaze. Trenutni mehanizmi zaštite uglavnom ne pružaju dovoljan nivo bezbednosti.

U ovom radu predloženo je i analizirano novo rešenje problema zaštite korisničkih podataka zasnovano na korišćenju složenih i slučajnih "salt" vrednosti pri individualnom generisanju heš vrednosti za lozinku koju je uneo korisnik kao i upotreba segmentnog hešovanja za šifrovanje lozinke uz standardnu metodu hešovanja SHA512. Lozinka se nakon

šifrovanja deli u dva dela i delovi se čuvaju u dve zasebne baze podataka koje se nalaze na različitim platformama. Smeštanje delova lozinke na različite baze podataka koje su na različitim platformama obezbeđuje podatke u slučaju kompromitovanja jedne platforme napadač dolazi u posed samo dela šifrovane lozinke. Da bi se izvršilo kompromitovanje obe baze neophodno je izvršiti napad na različite platforme (Linux, Windows, Unix …) kao i na različite sisteme baza podataka.

Metoda koja se koristi u ovom radu je nova i kombinuje više standardnih mehanizama kao i novih metoda koji zaštitu korisničkih podataka podižu na viši nivo i čine bazu otpornijom na napade. Savršeno bezbedan sistem ne postoji, ali postavljanjem dovoljno prepreka i otežavanjem procesa napadaču možemo nivo bezbednosti korisničkih podataka postaviti na visok nivo uz minimum ulaganja i promena u postojećim sistemima.

LITERATURA

[1] http://www.troyhunt.com/2011/06/brief-sony-password-analysis.html

[2] Anley C. „Advanced SQL Injection In SQL Server Applications“, An NGSSoftware Insight Security Research (NISR) Publication, 2002

[3] Bertino, E. Sandhu, R. „Database security - concepts, approaches, and challenges“ Dependable and Secure Computing, IEEE Transactions, vol.2, no.1, pp.2-19, Jan.-March 2005

[4] Zonenberg A. „Distributed Hash Cracker: A Cross-Platform GPU-Accelerated Password Recovery System“, Rensselaer Polytechnic Institute, 2009

[5] Adamović S. Milenković M. Šarac M. Radovanović D. “Generatori slučajnih sekvenci i njihov uticaj na sigurnost”, INFOTEH-JAHORINA Vol. 9, Ref. E-VI-1, p. 820-822, March 2010.

ABSTRACT

In this paper we analyze and propos a new solution to the problem of the protection user data. The method is based on the use of complex and random "salt" values of the individual generating the hash value of the password entered by the user as well as the use of segmented hashing to encrypt passwords with SHA512 hash method. Password after encryption divides into two parts and the parts are stored in two separate databases that are on different platforms.

SYSTEM OF SEGMENT PROTECTION OF USER DATA IN THE WEB APPLICATIONS

Nenad Ristić, Aleksandar Jevremović, Mladen Veinović