HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Sigurnosni propusti Web aplikacija

CCERT-PUBDOC-2004-10-93

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.
Rezultat toga rada ovaj je dokument koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, **www.lss.hr**- laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. UVOD 4

2. OPIS KORIŠTENIH ALATA 4

2.1. HACME BANK APLIKACIJA 4
2.2. PAROS PROXY POSLUŽITELJ 5

3. NAJČEŠĆE METODE NAPADA 6

3.1. UMETANJE SQL NAREDBI 6
3.1.1. Prijava za rad bez korisničkog imena i zaporke 7
3.1.2. Korištenje poruka o greškama u izvođenju aplikacije 7
3.1.3. Izvršavanje naredbi na poslužitelju 9
3.2. PRESRETANJE I MODIFIKACIJA HTTP PORUKA 9
3.2.1. Nedovoljna autorizacija korisnika 10
3.2.2. Modifikacija Cookie polja 12
3.2.3. Neispravno korištenje enkripcije 13
3.3. UMETANJE AKTIVNIH SKRIPTI 14
3.4. OSTALE METODE NAPADA 15

4. ZAKLJUČAK 16

5. REFERENCE 16

1. Uvod

Popularnost Web aplikacija u evidentnom je porastu, što se najbolje može vidjeti kroz sve šire područje njihove primjene kao što su npr. Internet bankarstvo, udaljeni pristup podacima, poslovni informacijski sustavi i sl. Kako funkcionalnost Web aplikacija najčešće uključuje i razmjenu povjerljivih podataka preko nesigurnih komunikacijskih kanala, njihova sigurnost postaje sve ozbiljniji sigurnosni problem.

Danas postoji velik broj tehnologija i alata koji programerima olakšavaju razvoj Web aplikacija. No, bez obzira na njihove brojne prednosti i jednostavnost upotrebe, postoje određena pravila koja je potrebno zadovoljiti ukoliko se osim same funkcionalnosti želi osigurati i odgovarajuća razina sigurnosti. Nedovoljna obučenost programera, prekratki rokovi za izradu aplikacija te neispravno korištenje raspoloživih tehnologija samo su neki od razloga koji mogu rezultirati nesigurnim programskim kodom koji neovlašteni korisnici mogu iskoristiti za neautorizirani pristup sustavu. Iskustva pokazuju da je velik broj Web aplikacija prisutnih na Internetu ranjivo na neki od propusta koji će biti kasnije opisani u ovom dokumentu što jasno ukazuje na ozbiljnost problema.

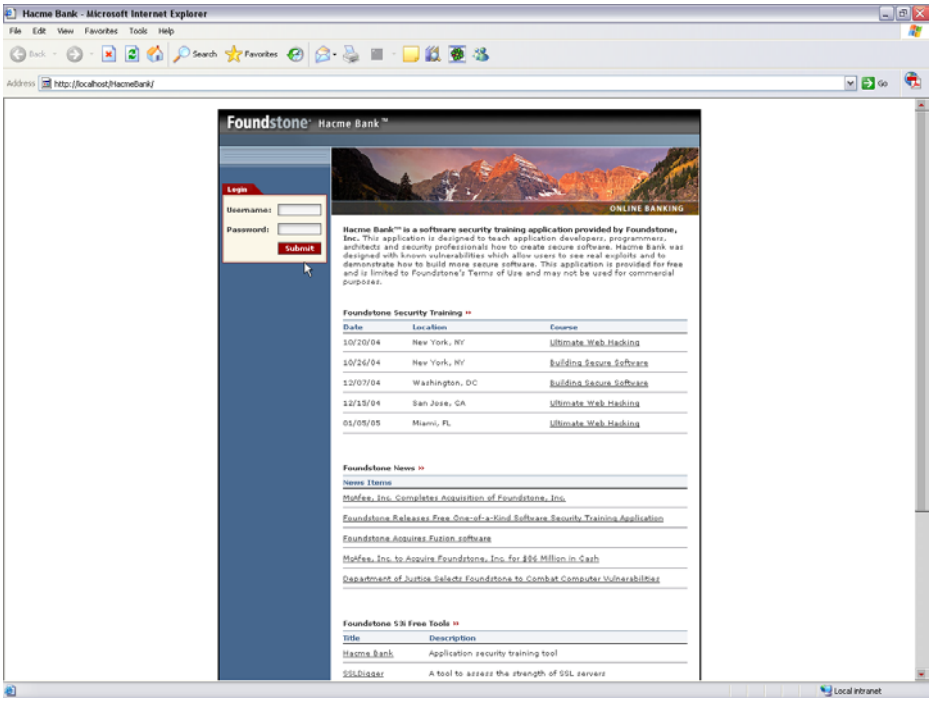
U nastavku dokumenta biti će opisani neki od najčešće korištenih metoda napada na Web aplikacije, načini njihovog provođenja te mogućnosti zaštite. Metode opisane u dokumentu vrijede općenito, ali se primjeri odnose prvenstveno na Web aplikacije bazirane na Microsoft tehnologiji tj. aplikacije koje koriste ASP.NET tehnologiju, rade na IIS Web poslužitelju i koriste SQL Server baze podataka.

Metode opisane u dokumentu ilustrirane su s nekoliko praktičnih primjera najčešćih napada na Web aplikacije. U svrhu demonstracije korišten je Hackme Bank programski paket, besplatna Web aplikacija napisana u ASP.NET tehnologiji, koja je posebno zanimljiva s obzirom da sadrži namjerno ugrađen velik broj najčešće korištenih sigurnosnih propusta, koji se mogu iskoristiti kako za testiranje, tako i za edukaciju programera, sigurnosnih stručnjaka te svih onih koji su zainteresirani za ovo iznimno važno područje računalne sigurnosti.

2. Opis korištenih alata

2.1. Hacme Bank aplikacija

Hacme Bank je primjer Web aplikacije za Internet bankarstvo koja je razvijena s ciljem da se programere i sigurnosne stručnjake educira o tipičnim problemima sigurnosti Web aplikacija. Aplikacija korisniku omogućuje autentikaciju pomoću jedinstvenog korisničkog imena i zaporke, pregled stanja računa, prebacivanje novca između računa te uzimanje kredita kako bi se stimulirala funkcionalnost sličnih aplikacija koje se svakodnevno primjenjuju u praksi.



Slika 1: Izgled početne stranice Hacme Bank aplikacije

U aplikaciju je namjerno ugrađen niz sigurnosnih propusta koji omogućuju provođenje različitih tipova napada te pristup povjerljivim podacima koji se obrađuju u okviru aplikacije. Napadač se korištenjem relativno jednostavnih tehnika, može prijaviti za rad s aplikacijom bez poznavanja korisničkog imena i lozinke, može prebacivati novac s raznih računa u banci, uzimati kredite s nultom stopom kamata, podesiti iznos na svom računu na bilo koju svotu i sl. Svi ovi propusti postavljeni su upravo iz razloga kako bi se Web programere, sigurnosne stručnjake te ostale korisnike što bolje upoznalo sa problemima zaštite Web aplikacija.

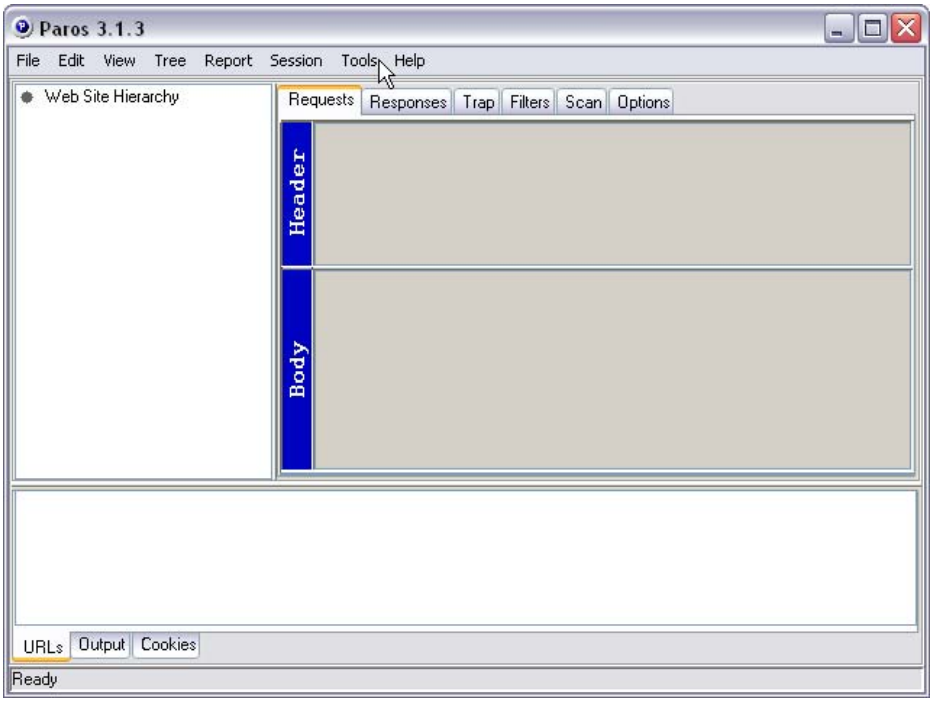
Program koristi Microsoft ASP.NET tehnologiju te radi sa IIS web poslužiteljem i SQL Server bazom podataka.

Aplikacija je besplatna, a besplatni su i svi alati potrebni za njeno pokretanje. Prije instalacije Hacme Bank aplikacije potrebno je instalirati IIS Web poslužitelj koji je dio instalacijskog programa novijih verzija Windows operacijskog sustava. Osim toga potrebno je instalirati .NET okruženje koje je također besplatno i koje je moguće dohvatiti sa Microsoftovih Web stranica. Za rad aplikacije je potreban i SQL poslužitelj za rad s bazama podataka koji je komercijalan proizvod, ali postoji verzija smanjenih mogućnosti pod nazivom MSDE (*Microsoft SQL Server Desktop Engine*) koja je besplatna i dovoljna za rad Hacme Bank aplikacije.

Na slici 1 prikazano je korisničko sučelje Hacme Bank aplikacije. Spomenuta aplikacija ujedno će se koristiti kao testno okruženje prilikom opisivanja različitih tehnika napada na Web aplikacije kako bi se na konkretnim primjerima jednostavnije uočile njihove osnovne karakteristike.

2.2. Paros proxy poslužitelj

Paros proxy poslužitelj je besplatni *open source* programski paket napisan u Javi i za rad tog programa je prethodno potrebno instalirati Java VM okruženje. Paros poslužitelj omogućuje pregled cjelokupne HTTP komunikacije između klijenta i poslužitelja te presretanje i izmjenu podataka u zaglavljima HTTP poruka. Korisničko sučelje programa Paros je prikazano na slici 2.



Slika 2: Izgled korisničkog sučelja Paros proxy poslužitelja

Budući da HTTP protokol bezkonekcijski (eng. *conectionless*) i da ne omogućuje upravljanje i kontrolu sjednica, Web poslužitelj svaki pristigli zahtjev promatra neovisno o prethodnoj komunikaciji te je problem uspostave i raskidanja sjednice potrebno riješiti na aplikacijskoj razini. U tom slučaju je odgovornost klijenta da u svakom HTTP zahtjevu pošalje podatke o prijavljenom korisniku i trenutnom stanju aplikacije kako bi Web poslužitelj na temelju tih podataka mogao pratiti stanje konekcije i odrediti ovlasti pojedinih korisnika.
Samim time podaci koji određuju identitet korisnika i njegova prava pristupa šalju se putem HTTP komunikacijskog kanala i presretanje ili lažiranje takvih poruka predstavlja veliki problem za sigurnost Web aplikacija. Problem je u određeno mjeri moguće ublažiti korištenjem HTTPS protokola, ali i u tom slučaju presretanje i modifikacija korisničkih upita nije onemogućena.
Paros poslužitelj je jednostavan programski alat koji omogućuje provođenje spomenutih napada, budući da omogućuje presretanje i modifikaciju poruka koje se razmjenjuju između klijenta i poslužitelja.

3. Najčešće metode napada

3.1. Umetanje SQL naredbi

Većina dinamičkih Web stranica sastoji se od odgovarajućeg korisničkog sučelja koje korisniku omogućuje interakciju sa samom aplikacijom te pristup podacima koji su pohranjeni u bazi podataka kojoj aplikacija pristupa. Na temelju korisničkog upita, aplikacija generira odgovarajuće SQL upite te klijentu ispisuje rezultat ovisno o izvršenom upitu. Komunikacija između klijenta i poslužitelja najčešće se odvija putem različitih Web formi koje korisniku omogućuju interakciju s aplikacijom.
Ovakav pristup osim svoje jednostavnosti predstavlja i potencijalni sigurnosni propust budući da korisnik preko ulaznih podataka koje unosi u formu posjeduje određenu razinu kontrole nad SQL upitom koji se šalje bazi podataka. Taj postupak zove se umetanje SQL naredbi (*eng. SQL injection*) i to je ujedno najčešći tip napada na Web aplikacije.
U ovom poglavlju ukratko je opisan postupak umetanja SQL naredbi, a putem popratnih primjera ilustrirano je što sve napadač može ostvariti primjenom ovakvih metoda napada. U dokumentu se

podrazumijeva da korisnik posjeduje osnovna znanja vezana uz sintaksu i način korištenja SQL jezika zbog čega navedeni primjeri SQL izraza nisu posebno objašnjavani.

3.1.1. Prijava za rad bez korisničkog imena i zaporke

Umetanje SQL izraza napadaču omogućuje prijavu za rad bez posjedovanja legitimnog korisničkog imena i zaporke za rad s aplikacijom. Prijava za rad s aplikacijom najčešće se sastoji od korisničkog imena i zaporke koje korisnik mora upisati.
Podatke o svim legitimnim korisnicima aplikacija pohranjuje u bazi podataka pa za provjeru ispravnosti korisničkog imena i zaporke aplikacija mora na osnovu podataka koje korisnik upiše, generirati odgovarajući SQL upit kojim će u bazi podataka provjeriti postojanje takvog korisnika.
Slijedi primjer takvog SQL upita za provjeru prijavljenog korisnika:

```
SQLQuery = "select username, password from table_users where username = '" +
str_username + "' and password = '" + str_password + "'";
```

Varijabla `str_username` sadrži korisničko ime koje je korisnik unio putem odgovarajućeg sučelja, a varijabla `str_password` sadrži pripadajuću zaporku. Obje varijable se umetnu u konačan tekst SQl upita koji je pohranjen u varijabli `SQLQuery`.
Ukoliko korisnik unese korisničko ime `Admin` i zaporku `1234`, konačan upit će izgledati ovako:

```
select username, password from table_users where username = 'Admin' and
password = '1234'
```

Kada se ovakav upit proslijedi bazi podataka, baza će vratiti zapis iz tablice `table_users` koji odgovara upisanim podacima (ukoliko takav korisnik postoji).
Najjednostavniji način provođenja napada u ovom slučaju je da korisnik unese korisničko ime koje je jednako:

```
'or 1=1 --
```

SQL upit će tada izgledati ovako:

```
select username, password from table_users where username = '' or 1=1 --' and
password = ''
```

Taj SQL upit će vratiti sve postojeće zapise iz tablice `table_users` jer upit traži sve zapise koji odgovaraju logičkom uvjetu `1=1`, što je uvijek istina. Oznaka `--'` u SQL upitu označava početak komentara pa se dio upita koji provjerava zaporku uopće ne izvršava.
Kada se ovaj upit izvrši napadač će se uspješno prijaviti kao prvi korisnik zapisan u bazi podataka.
Ako napadač zna postojeće korisničko ime, ali ne zna zaporku, može upisati korisničko ime na sljedeći način:

```
administrator' --
```

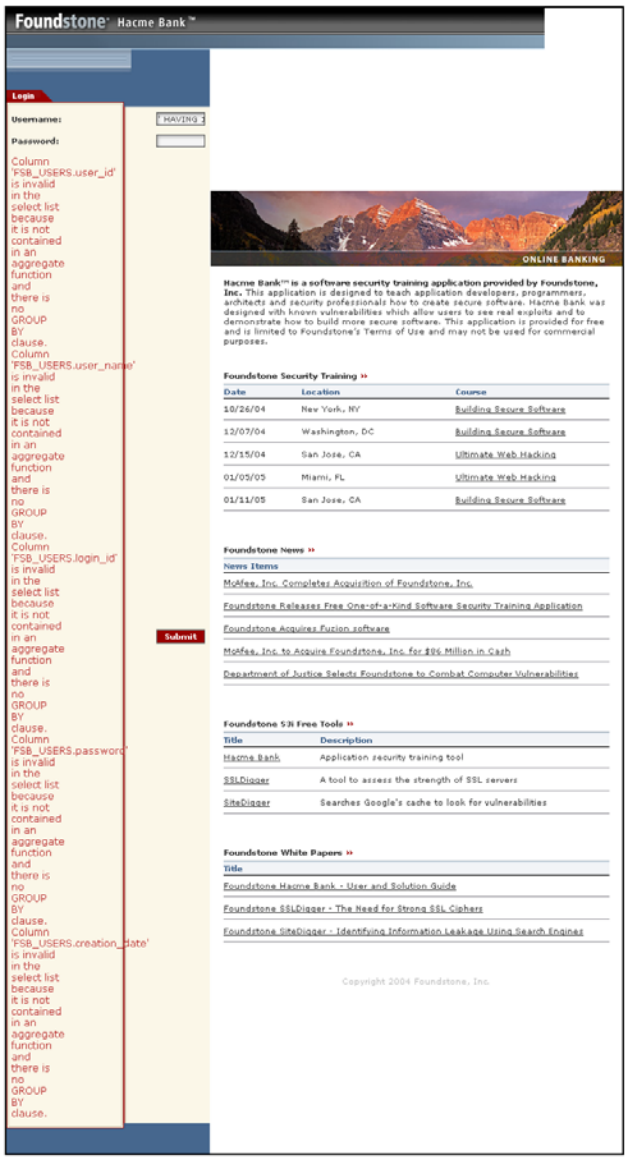
SQL upit će tada izgledati ovako:

```
select username, password from table_users where username = 'administrator' -
-' and password = ''
```

Baza će vratiti zapis gdje je korisničko ime jednako `administrator`, bez obzira na zaporku. Na taj način se napadač može prijaviti kao bilo koji korisnik u bazi.

3.1.2. Korištenje poruka o greškama u izvođenju aplikacije

Prema inicijalnim postavkama poslužitelja, klijentu se prosljeđuju detaljne poruke o svim greškama tijekom izvođenja aplikacije. Generiranje tih poruka iznimno je korisno programerima prilikom razvoja, jer omogućuje točno lociranje grešaka u kodu aplikacije i njihovo uklanjanje.
Međutim napadač može namjerno inicirati grešku u izvođenju aplikacije i iz poruka o grešci saznati detaljnije informacije o inačici Web poslužitelja, strukturi baze podataka i programskom kodu aplikacije. Tako prikupljene informacije moguće je iskoristiti za analizu potencijalnih slabosti unutar sustava te detaljnije planiranje sljedećih faza napada.
Otkrivanje ovakvih informacija moguće je spriječiti podešavanjem Web poslužitelja tako da ne ispisuje poruke o greškama u radu aplikacije. Samim time napadaču se znatno otežava provođenje napada, budući da se ne otkrivaju podaci o strukturi baze i načinu rada same aplikacije.
Poruke o greškama poslužitelja mogu se iskoristiti u aplikaciji Hacme Bank za otkrivanje strukture tablice u bazi podataka koja sadrži podatke o korisničkim imenima i zaporkama. Slijedi primjer,



Slika 3: Poruka o greškama u radu aplikacije

Ukoliko korisnik upiše korisničko ime jednako:

```
' having 1=1 --
```

Generirani SQL upit imat će nedopuštenu sintaksu i aplikacija će korisniku vratiti poruku o grešci (Slika 3).

U detaljnom ispisu pogreške vidljiva su imena svih polja u tablici `FSB_USERS` koja sadrži podatke o korisnicima aplikacije. Polja u toj tablici su: `user_id`, `user_name`, `login_id`, `password` i `creation_date`.

Nakon što su poznata imena polja (stupaca) u tablici, napadač može saznati i tip podataka za svako polje. Tip polja moguće je saznati jednostavno tako da se upiše korisničko ime

```
' union select sum(field_name) from FSB_USERS --
```

, pri čemu je `field_name` ime polja za koje se ispituje tip podataka.

Ovakvo uneseno korisničko ime generirati će SQL upit sa nedopuštenom sintaksom, ali se greška razlikuje za brojčani i znakovni tip podataka u polju. SQL upit koristi funkciju za računanje sume podataka u jednom stupcu, a ta funkcija je definirana samo za brojčane tipove podataka. Za znakovne

tipove će se ispisati poruka o grešci koja će navesti tip podataka u tom polju i poruku da za taj tip podataka funkcija `sum` nije definirana. Za brojčane tipove podataka nastat će greška jer nije moguće kreirati uniju između zapisa sa različitim brojem polja i poruka o grešci će biti drugačija.
Nakon što su poznati svi navedeni podaci napadač može u bazu podataka unijeti novog korisnika i to tako da upiše sljedeće korisničko ime:

```
'; insert into FSB_USERS values(100, 'Hacker', 'hacker', '1234', getdate()) -
-
```

Prikazana SQL naredba će u bazu dodati korisnika s korisničkim imenom `hacker` i zaporkom `1234`.

3.1.3. Izvršavanje naredbi na poslužitelju

SQL poslužitelj ima ugrađene odgovarajuće dll biblioteke funkcija (*eng. extended stored procedures*) koje programeru omogućuju izvođenje raznih naredbi na operacijskom sustavu poslužitelja. Programer može pokrenuti bilo koju naredbu ljuske operacijskog sustava, čitati i pisati podatke u Windows *registry* datoteku, slanje e-mail pošte i sl. Te naredbe pokreće SQL poslužitelj i programer ih može pokrenuti jednostavnim slanjem poruka poslužitelju za baze podataka. Ovakav sustav je ranjiv na napad umetanjem SQL naredbi jer napadač može umetnuti poruku koja sadrži naredbu za izvršavanje neke od raspoloživih funkcija.
Na primjer za popis datoteka u radnom direktoriju poslužitelja, napadač može upisati sljedeće korisničko ime jednako:

```
'; exec master..xp_cmdshell 'dir' --
```

, pri čemu je `xp_cmdshell` naziv vanjske procedure, a `'dir'` parametar koji se prosljeđuje. Procedura `xp_cmdshell` pokreće bilo koju naredbu iz Windows komandne ljuske pa će npr. korisničko ime:

```
'; exec master..xp_cmdshell 'net1 user' --
```

ispisati sve korisnike prijavljene korisnike na računalu.
Neke od procedura za rad s Windows *registry* podacima su `xp_regread`, `xp_regwrite`, `xp_regdeletevalue` i `xp_regdeletekey` i napadaču omogućuju pregled i izmjenu svih vrijednosti u Windows *registry* datoteci.
Još neke procedure koje napadaču mogu biti korisne prilikom provođenja napada su `xp_servicecontrol`, koja može pokretati i zaustavljati Windows servise, funkcija `xp_loginconfig` vraća podatke o sigurnosnim postavkama poslužitelja, funkcija `xp_terminate_process` može zaustaviti svaki proces na sustavu, funkcija `xp_enumdsn` vraća popis ODBC izvora podataka na računalu, a funkcija `xp_ntsec_enumdomains` vraća popis domena kojima računalo ima pristup. Na temelju iznesenih podataka može se jasno zaključiti da ovakvi sigurnosni propusti mogu imati ozbiljne posljedice s obzirom da se neovlaštenom korisniku omogućuje preuzimanje potpune kontrole nad sustavom.

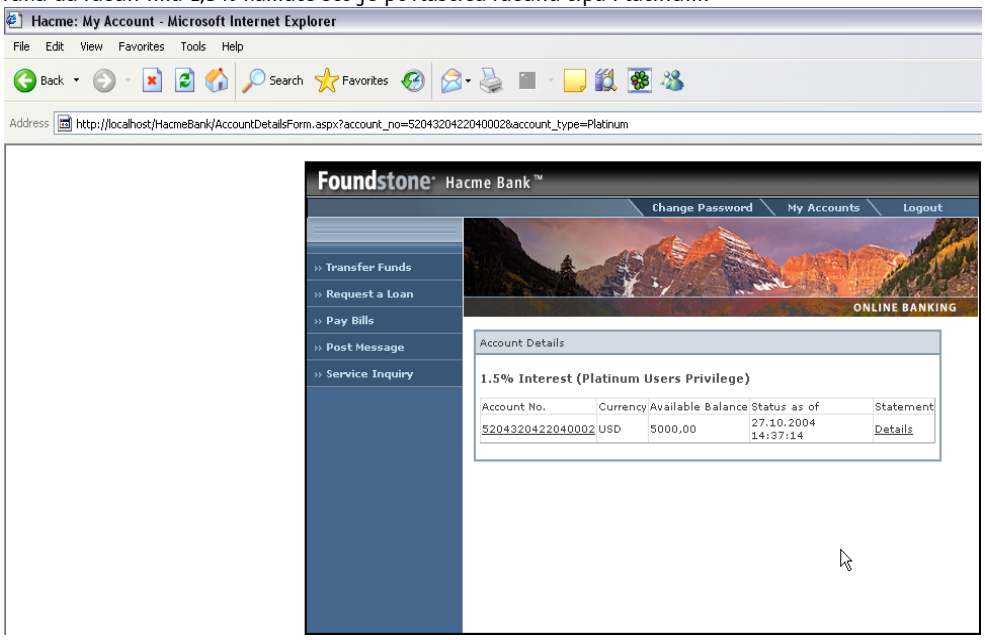
3.2. Presretanje i modifikacija HTTP poruka

Kao što je već ranije spomenuto u poglavlju o Paros poslužitelju, Web klijent mora za cijelo vrijeme korisnikovog rada s aplikacijom, poslužitelju slati podatke o korisničkim pravima za pristupanje povjerljivim podacima. Osim identifikacijskih podataka, klijent mora poslati i podatke koje je korisnik upisao u forme na stranicama aplikacije.
HTTP protokol definira 2 načina na koje Web klijent aplikacija može prosljeđivati poruke poslužitelju. Ta dva načina su GET i POST metode.
GET metoda koristi se u slučajevima kada klijent šalje zahtjev za statičku HTML stranicu. Unutar GET poruke klijent definira ime dokumenta, direktorij u kojem se dokument nalazi na poslužitelju i komunikacijski protokol za slanje dokumenta (HTTP, HTTPS, FTP). Cijeli taj zahtjev je vidljiv u adresnom polju Web klijent aplikacije i zove se URL (engl. *Uniform Resource Locator*) dokumenta.
Kada Web stranica sadrži Web forme ili je dio Web aplikacije, onda klijent mora poslužitelju proslijediti znatno više informacija nego u slučaju statičkih stranica. Te podatke moguće je slati pomoću GET metode tako da se dodaju na kraj URL niza tog dokumenta, ali je ovakav pristup prilično nepraktičan za slanje velike količine podataka pa se u tu svrhu mnogo češće koristi POST metoda.
POST metoda šalje sve podatke koje je korisnik upisao u forme ili podatke o identitetu i pravima korisnika, u tijelu HTTP poruke, a ne u tekstu URL adrese.
Slanje podataka POST metodom omogućuje slanje veće količine podataka, a podaci nisu vidljivi u adresnom polju Web klijenta kao što je to slučaj kod GET metode.

Sigurnosni problem kod obje spomenute metode je mogućnost presretanja i modifikacije podataka prije nego što se oni proslijede poslužitelju. Ukoliko se podaci šalju GET metodom moguće ih je izmijeniti u adresnom polju Web klijenta, bez korištenja bilo kakvih dodatnih alata. Za podatke koji se šalju POST metodom potreban je proxy poslužitelj (npr. Paros) koji može presresti i prikazati podatke koji se šalju unutar HTTP zahtjeva.
Modifikacijom podataka o identitetu i pravima pristupa korisnika unutar HTTP zahtjeva, napadač može preuzeti identitet bilo kojeg legitimnog korisnika te ostvariti pristup sustavu sa odgovarajućim ovlastima koje su pridjeljene tom korisniku. Aplikacije se najčešće brane od takvih napada korištenjem enkripcija, ali ukoliko se podaci šalju preko nesigurnog komunikacijskog kanala napad je uvijek moguć.

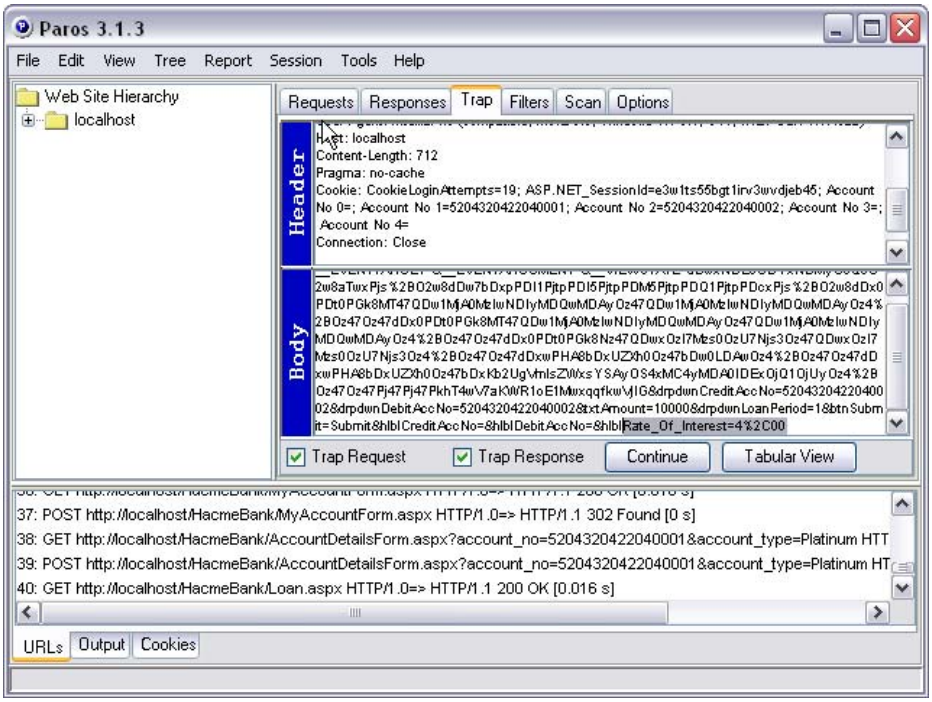
3.2.1. Nedovoljna autorizacija korisnika

U nastavku dokumenta su opisana dva primjera koji koriste nedovoljnu autorizaciju korisnika za napad na aplikaciju Hacme Bank. U prvom primjeru korisnik može povećati ovlasti za svoj tekući račun. Aplikacija ima podršku za dvije vrste tekućih računa: *Silver* i *Platinum*. *Platinum* račun je skuplji, ali ima veću pasivnu kamatu na novac položen na tom računu.
Kada korisnik izabere opciju za prikaz podataka o računu, aplikacija GET metodom šalje podatak o vrsti računa. Cijeli URL niz takvog zahtjeva je prikazan u adresnom polju Internet Explorer Web preglednika prikazanom na slici 4. Na slici je prikazan *Silver* tip računa, ali je u adresnom polju URL izmijenjen tako da umjesto *Silver* piše *Platinum*. Ako se nakon toga ponovno učita ista stranica, aplikacija će prikazati poruku da račun ima 1,5% kamate što je povlastica računa tipa *Platinum*.

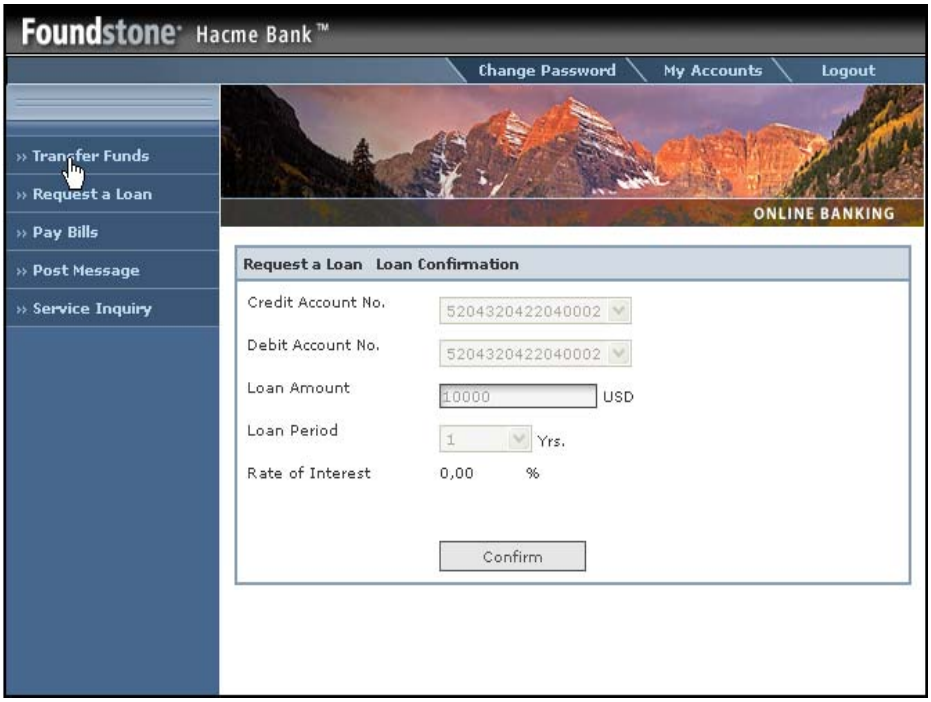


Slika 4: Promjena vrste računa

Drugi primjer prikazuje promjenu kamatne stope kredita modifikacijom HTTP poruke u Paros proxy poslužitelju.
Korisnik može koristiti aplikaciju za uzimanje kredita od banke. U formi za uzimanje kredita korisnik upisuje broj računa, iznos kredita i rok za vraćanje. Aplikacija sve te podatke šalje poslužitelju pomoću POST metode.



Slika 5: Promjena kamatne stope



Slika 6: Kredit s 0% kamata

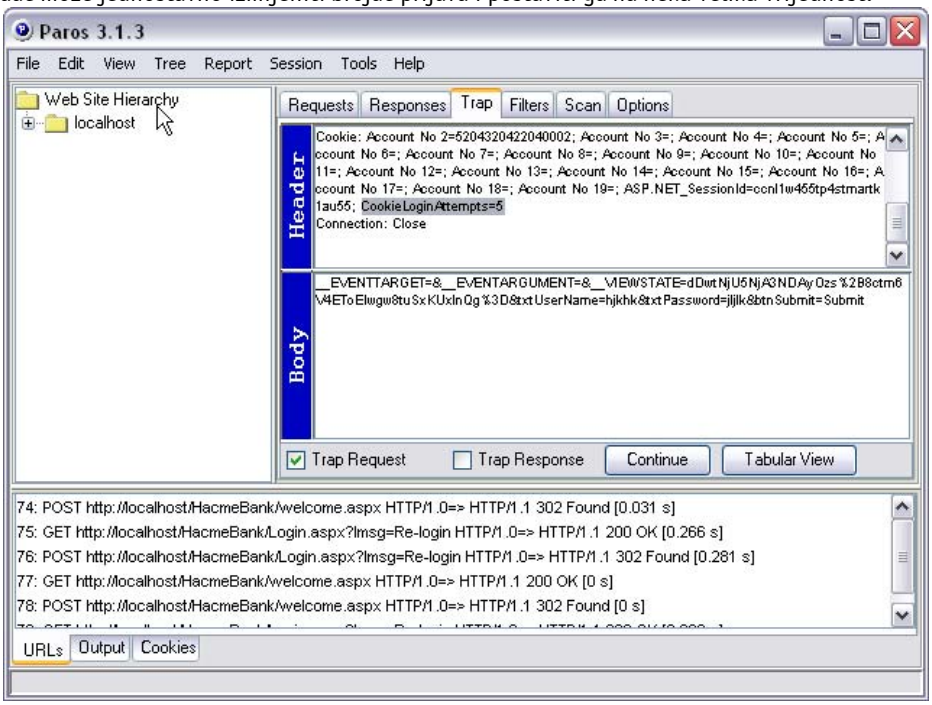
Na slici 5 je prikazan HTTP zahtjev u prozoru Paros poslužitelja. Dio poruke je i niz `Rate_Of_Interest` iza kojeg je upisana kamatna stopa. Napadač može tu brojku promijeniti u 0% i tada poslati zahtjev poslužitelju. Na slici 6 je vidljivo da je aplikacija korisniku odobrila kredit s 0% kamata.

3.2.2. Modifikacija Cookie polja

Cookie je tekstualni niz koji služi za autentikaciju korisnika i održavanje korisničke sjednice. Poslužitelj šalje *cookie* niz klijentu i klijent ga mora vratiti sa svakim novim zahtjevom za vrijeme trajanja korisničke sjednice. Aplikacije koriste *cookie* polja za spremanje različitih podataka koje aplikacija mora pratiti između klijentskih zahtjeva. Napadač može izmijeniti te podatke i tako prevariti poslužitelj da poduzme akciju koja odgovara napadaču.

Najjednostavniji primjer napada modifikacijom *cookie* polja je izmjena brojača neuspješnih prijava. Hacme Bank aplikacija dopušta korisniku da 5 puta upiše korisničko ime i zaporku nakon čega se aplikacija zaključa. Aplikacija ima brojač neuspješnih prijava koji je dio *cookie* tekstualnog niza. Brojač se na početku postavi na 5 i smanji se za jedan prilikom svake neuspjele prijave.

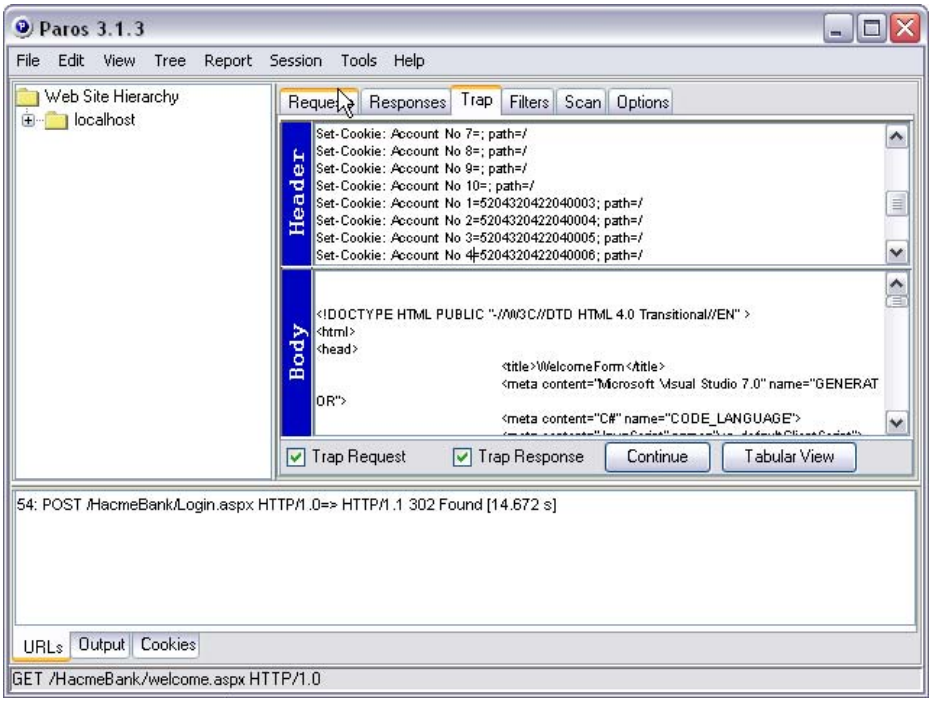
Takav brojač je zaštita od tzv. *brute force* napada u kojima se napadač pokušava prijaviti za rad isprobavanjem velikog broja zaporki korištenjem rječnika (engl. *dictionary attack*). Međutim ovdje napadač može jednostavno izmijeniti brojač prijava i postaviti ga na neku veliku vrijednost.



Slika 7: Promjena cookie polja sa brojačem neuspješnih prijava

Drugi primjer se odnosi na modifikaciju brojeva tekućih računa. Aplikacija nakon uspješne prijave korisnika generira cookie niz sa brojevima svih njegovih računa. Klijent aplikacija cijelo vrijeme vraća taj niz poslužitelju kako bi ovaj znao s kojim računima korisnik smije raditi.

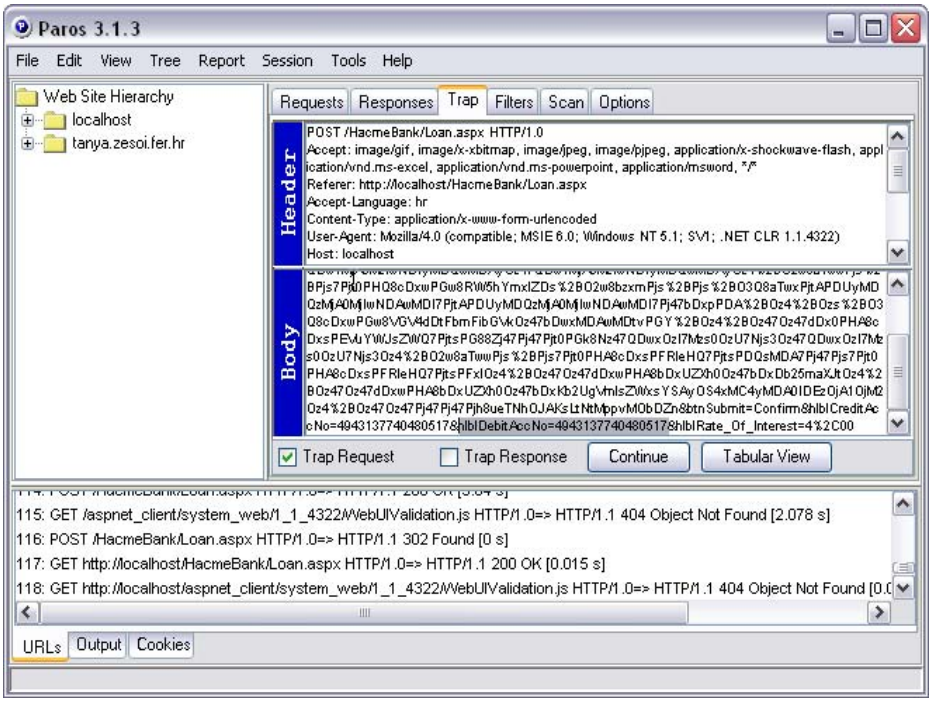
Ako napadač izmijeni brojeve računa kao što je prikazano na slici 8, može koristiti tekući račun koji nije njegov. Jedini uvjet je da napadač u *cookie* polju navede brojeve računa koji stvarno postoje, ali pripadaju nekom drugom. Ako se upiše nepostojeći broj računa aplikacija će prijaviti grešku.



Slika 8: Promjena cookie polja sa brojevima računa

3.2.3. Neispravno korištenje enkripcije

Na sljedećoj slici (Slika 9) prikazana je POST poruka aplikacije Hacme Bank koja se šalje kada korisnik zatraži kredit. Podaci o kreditu sadrže brojeve o dva tekuća računa. Prvi broj označava račun na koji se isplaćuje kredit, a drugi je broj računa sa kojeg se skidaju rate kredita. Oba broja se šifriraju i šalju u tijelu POST poruke koja je prikazana na slici 9.

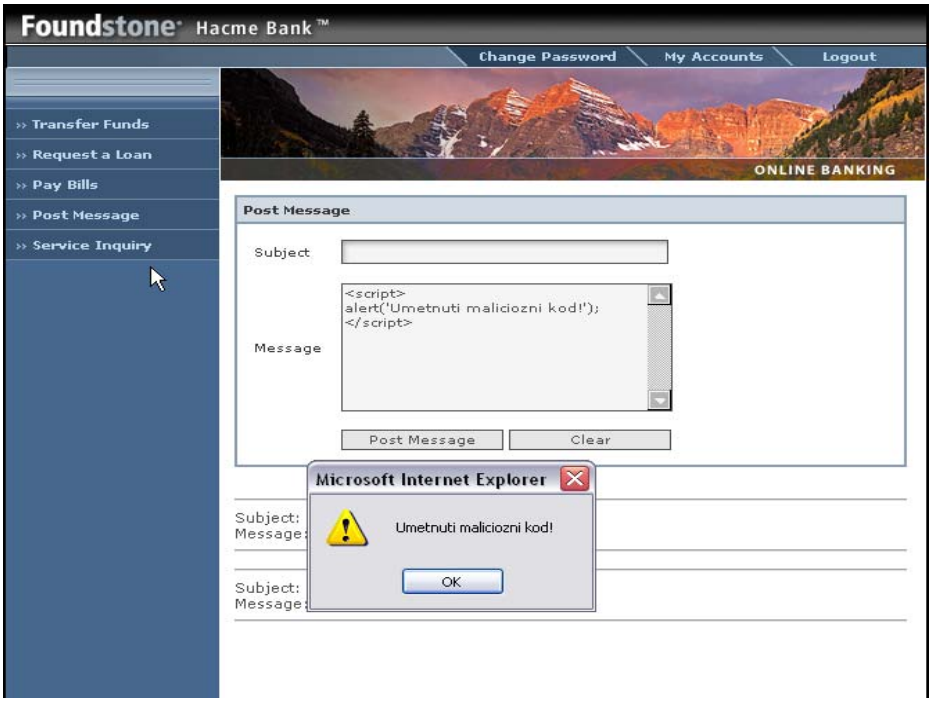


Slika 9: Neispravno korištenje enkripcije

Brojevi računa u poruci su šifrirani pomoću XOR logičke funkcije sa nizom 1111111111111111. To je prilično jednostavna metoda enkripcije i napadači lako mogu dešifrirati prave brojeve računa. Nakon toga napadač može šifrirati neki drugi broj računa i umetnuti ga umjesto svog. Aplikacija će poslani broj računa prihvatiti bez provjere i sa njega skidati rate kredita.
Obrana od napada ove vrste je korištenje snažnijih algoritama za enkripciju podataka i slanje samo najnužnijih podataka putem računalne mreže.

3.3. Umetanje aktivnih skripti

Umetanje aktivnih skripti (eng. *Cross site scripting, XSS*) je postupak u kojem napadač pokušava u stranicu aplikacije umetnuti proizvoljni skriptni kod. Kada klijent aplikacija učitava stranicu s poslužitelja automatski će izvršiti sve skripte na toj stranici. Mogućnost provođenja Cross Site Scripting napada postoji svugdje gdje aplikacija na stranicu ispisuje podatke koje je korisnik prethodno unio putem npr. Web formi. Ukoliko maliciozni korisnik u Web formu unese maliciozni skriptni kod, i ukoliko to aplikacija ne detektira, prilikom ispisa unesenog sadržaja skriptni kod će s izvršiti u okviru Web preglednika korisnika koji pristupa stranici.
Važno je napomenuti da se skripte napisane u skriptnim jezicima kao JavaScript ili VB Script izvršavaju na strani klijenta (za razliku od npr. PHP ili ASP koda koji se izvršava na strani poslužitelja), i takvi programi imaju pristup mnogim resursima klijentskog računala. Hacme Bank aplikacija sadrži stranicu koja korisnicima omogućava umetanje poruka koje će se ispisati na stranicama aplikacije. Budući da aplikacija ne provjerava tekst koji korisnik unosi, napadač može kao poruku umetnuti JavaScript kod kao što je prikazano na Slici 10. taj kod će postati dio HTML koda te stranice i kod ponovnog učitavanja će se izvršiti.
Na slici 10 je prikazan JavaScrip kod koji otvara dijaloški okvir s porukom kao što je vidljivo na slici 10. Međutim napadač može u stranicu umetnuti kod koji je mnogo opasniji od ovdje prikazanog.

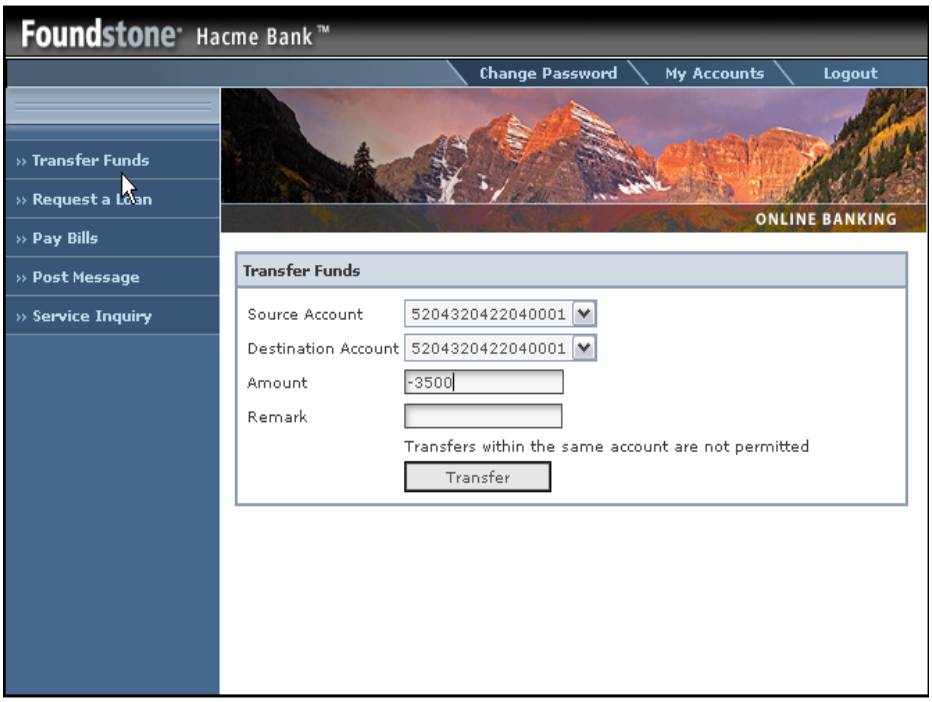


Slika 10: Umetanje aktivne skripte u stranicu

Na umetanje aktivnog koda nisu ranjive samo Web aplikacije. Napadači često šalju e-mail poruke sa kodom koji se izvršava u trenutku kada korisnik otvori takvu poruku. Takve skripte mogu se koristiti za prikupljanje različitih povjerljivih korisničkih podataka.
Od umetanja aktivnog koda aplikacija se može zaštiti tako da parsira upisani tekst i ne dopušta unos teksta koji odgovara kodu nekog skriptnog jezika. Isto tako moderni vatrozidi novije generacije mogu ovisno o sigurnosnim postavkama, preduhitriti izvođenje koda skripte i tako zaštititi računalo od napada.

3.4. Ostale metode napada

Osim opisanih, postoje i mnoge druge metode napada na Web aplikacije, a neprestano se otkrivaju i razvijaju nove metode, tehnike i alati. Većina napada temelji se na propustima u kodu aplikacije koji nastaju zbog nedovoljne provjere ulaznih podataka, površno implementirane enkripcije i sl.
Česti propusti programera uključuju i nedovoljnu provjeru podataka koji se upisuju u varijable u kodu aplikacije. Ako korisnik upiše neispravnu vrijednost u Web formu, ta vrijednost će biti bez provjere upisana u varijablu koja možda ne podržava unesenu vrijednost. Napadač na taj način može upisati tekstualni niz koji je dulji od rezervirane memorije za spremanje tog niza ili numeričku vrijednost krivog formata npr. realnu vrijednost u cjelobrojnu varijablu, negativnu vrijednost u varijablu koja ne podržava predznak ili tekstualni niz u varijablu koja je predviđena za numeričku vrijednost.
Na taj način napadač može uzrokovati grešku u aplikaciji, i iz njenog opisa smisliti novi napad (kao što je to opisano u poglavlju o umetanju SQL izraza). Osim toga napadač umetanjem neispravnih ulaznih podataka može prevariti aplikaciju koja će te podatke obraditi kao da su ispravni.
Jedan primjer takvog napada je upisivanje negativnog broja u formu za prijenos novca između računa u aplikaciji Hacme Bank. Aplikacija korisniku omogućava prijenos željenog iznosa s jednog računa na drugi. Ako se upiše negativni iznos novca, aplikacija će povećati iznos na računu s kojeg se vrši isplata.



Slika 11: Umetanje aktivne skripte u stranicu

Na Slici 11 prikazana je forma za prijenos novca u aplikaciji Hacme Bank.

4. Zaključak

U dokumentu je ukratko opisano nekoliko osnovnih metoda za krađu podataka i preuzimanje kontrole nad izvođenjem Web aplikacija. Na primjeru Hacme Bank aplikacije prikazano je kako se te metode mogu primijeniti u praksi i što sve napadač može postići uspješnim napadom na aplikaciju.
Na temelju provedenog testiranja može se zaključiti da su sigurnosni propusti ugrađeni u Hacme Bank aplikaciju uglavnom prilično jednostavni i "naivni" što je i razumnljivo s obzirom da se radi o testnom sustavu namijenjenom edukaciji i globalnom podizanju svijesti Web programera i sigurnosnih stručnjaka. Većina opisanih problema može se spriječiti uz minimalne promjene koda i konfiguracijskih postavki sustava, no ipak zbog neznanja i neiskustva programera, opisani sigurnosni propusti vrlo se često pojavljuju u praksi i napadači s vrlo malo znanja i iskustva mogu izvršiti uspješne napade na mnoge Web aplikacije.
Web aplikacije pružaju mnoge mogućnosti za udaljeni pristup podacima, pružanje usluga korisnicima, kolaboraciju timova i udaljeni rad, ali osim toga te aplikacije donose mnoge sigurnosne probleme kojima će se u budućnosti morati posvećivati sve veća pažnja.

5. Reference

[1] Advanced SQL injection in SQL server applications,
http://www.nextgenss.com/papers/advanced_sql_injection.pdf
[2] The Cross site scripting FAQ,
http://www.cgisecurity.com/articles/xss-faq.shtml
[3] Hacme Bank,
http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/hacmebank.htm