VISOKA POSLOVNA ŠKOLA
STRUKOVNIH STUDIJA
ČAČAK

MAGISTARSKI RAD

Primena sopstvenog šifarskog algoritma za zaštitu ajax poziva kod web aplikacija

Mentor:_____ Student: _____
Profesor: _____ Br.Indeksa: _____

Mesto, mesec, godina

Primena sopstvenog šifarskog algoritma za zaštitu AJAX poziva kod Web aplikacija

Aleksandar Jevremović, Xxxx Xxxxx, Xxxxx Xxxxx
Fakultet za poslovnu informatiku, Univerzitet Singidunum, Beograd

Sadržaj – *U ovom radu se razmatra kriptografska zaštita podataka kod AJAX poziva u savremenim Web aplikacijama. Predložena je soptvena realizacija algoritma i kontrola svih kriptografskih parametara bez oslanjanja na komercijalno dostupne protokole zaštite koji se zasnivaju na SSL/TLS protokolu transportnog sloja. Analiza je izvršena za različite klijent/server softverske platforme. U rezultatima je diskutovan uticaj kriptografskih funkcija na pad performansi i predložena je primena na bezbednosno osetljive podatke u fazi prijavljivanja na udaljeni sistem (prenos podataka tipa PIN, password, broj kreditne kartice i sl.). Fokus naših ranijih radova [1,2] je bio postavljen na realizaciju sistema u okviru kernela Linux operativnog sistema i C programskog jezika dok se u ovom radu on premešta na HTTP protokol i programske jezike namenjene Web aplikacijama - PHP i JavaScript. Iako se ovakvom izmenom noseće platforme podrazumeva znatan pad performansi, sve veća popularnost AJAX pristupa kao i sve veća procesorska moć računara otvara prostor i potrebu za takvom realizacijom. Rad se ne bavi problemom distribucije ključeva i samog algoritma već isključivo realizacijom i performansama algoritma putem pomenutih tehnologija.*

1. UVOD

Ubrzan razvoj HTTP protokola i pridruženih tehnologija doveo je do stepena interaktivnosti u radu sa hiper-tekstualnim sadržajem koji omogućava razvoj kompletnih korisničkih aplikacija predviđenih za distribuiranje i korišćenje isključivo putem Web-a. Web aplikacije su danas nezaobilazni deo svakog modernog poslovnog sistema koji komunicira sa velikim brojem geografski udaljenih klijenata. Najčešće korisnike ovog tipa softvera predstavljaju banke, vladine institucije, javni servisi i sl. sa jedne strane i građani (čiji stepen tehničkog poznavanja računara i mrežne komunikacije znatno varira) sa druge strane.

Rast realnih vrednosti kojima se može upravljati korišćenjem Web-a putem direktne veze sa stepenom rizika već duže vreme drži otvorenim pitanje bezbednosti i privatnosti podataka korišćenih putem Web aplikacija. U tom cilju su razvijena različita rešenja sa ciljem da ponude što viši stepen bezbednosti uz što manji uticaj na performanse celokupnog sistema. Dodatni prirodni zahtev postavljen pred ova rešenja jeste i što manji potrebni nivo tehničkog znanja za njihovo korišćenje. Najstandardnije (a ujedno i najčešće korišćeno rešenje) se javilo u vidu HTTPS protokola odnosno zaštite HTTP protokola putem SSL/TLS protokola. Jedan od često korišćenih pristupa kod zatvorenih sistema jeste i zaštita na nižim slojevima OSI modela [1,2].

Naša istraživanja u ovoj oblasti su usmerena na rešavanje pitanja bezbednosti kod Web aplikacija u okviru HTTP protokola i bez korišćenja SSL/TLS protokola i na osnovu sopstvenih algoritama za šifrovanje. Poznato je da profesionalni sistemi zaštite zahtevaju sopstveno realizovane šifarske algoritme i potpunu kontrolu kriptografskih ključeva. Ovim istraživanjima su obuhvaćene različite tehnologije vezane za serverski i klijentski deo Web servisa (Java, JavaScript, Adobe Flash, PHP, Ajax i sl.) a za okvire ovog rada je odabran prikaz rešenja realizovanog putem PHP i JavaScript programskih jezika korišćenih putem Ajax poziva. Važna napomena je i da je fokus rada postavljen na analizu ostvarenih performansi rešenja - odnosno na njihovu upotrebljivost u realnim okruženjima - dok je problem upravljanja ključevima ostavljen za neku od narednih faza istraživanja.

2. PREGLED POSTOJEĆIH REŠENJA

Jedan od glavnih uzroka naglog porasta popularnosti Weba (i pridruženih tehnologija) kao platforme za razvoj distribuiranog softvera jeste jednostavan razvoj određenih komponenata rešenja (pre svega korisničkog interfejsa) putem delegiranja određenih funkcionalnosti (funkcije mrežnih protokola, renderovanja prikaza, briga o bezbednosti klijenta i sl.) na HTTP protokol, HTTP servere i HTTP klijente (eng. *Web browser*). U skladu sa tim, jedno od glavnih rešenja za zaštitu saobraćaja kod Web aplikacija jeste i HTTPS, sistem zaštite koji je obezbeđen na nivou pomenutih komponenata Web platforme.

HTTPS (Hypertext Transfer Protocol over Secure Socket Layer) predstavlja kombinaciju HTTP protokola sa SSL (Secure Sockets Layer) ili TLS (Transport Layer Security) sistemima za šifrovanje. Jedna od karakteristika ovog sistema jeste i korišćenje asimetričnih ključeva. Sam HTTPS sistem zaštite se može smatrati dobrim konceptom ali ga sledeće karakteristike čine nekompatibilnim sa rešenjem koje je postavljeno kao cilj ovog rada:

1. Za implementaciju sopstvenog algoritma u okviru HTTPS protokola potrebno je modifikovati standardizovane komponente - HTTP server i HTTP klijent - tj. deo koji je zadužen za rad sa SSL/TLS sistemom šifrovanja.
2. Regulative Ministarstva odbrane u SAD su obavezivale proizvođače Web klijenata (Navigator od proizvođača Netscape i Internet Explorer od proizvođača Microsoft) da njihovi proizvodi koriste 40-bitne ključeve ukoliko se upotrebljavaju van teritorije SAD.
3. SSL/TLS sistem šifrovanja se bazira na korišćenju PKI infrastrukture što zahteva poseban način distribuiranja samih Web klijenata zarad dobijanja pouzdanosti.

HTTPS sistem zaštite pri prenosu podataka nudi određen stepen zaštite autentičnosti, poverljivosti i integriteta podataka. Takođe, performanse ovog sistema su prihvatljive za većinu scenarija u kojima bi on bio korišćen. Međutim, opisani nedostaci čine realizaciju sopstvenog algoritma za

šifrovanje u okviru HTTPS-a previše složenom i zavisnom od komponenata koje nisu pod potpunom kontrolom korisnika.

Sledeće rešenje koje je analizirano jeste aSSL[3]. Ovo rešenje je realizovano putem tehnologija koje su identifikovane kao pogodne za izradu rešenja postavljenog kao cilj ovog rada. aSSL se sastoji od dve komponente:

1. klijentske komponente realizovane putem JavaScript programskog jezika;
2. serverske komponente realizovane putem ASP tehnologije.

Za razmenu 128-bitnih ključeva aSSL koristi RSA algoritam, a postupak je poznat pod nazivom digitalna envelopa. Tajni simetrični ključ se generiše za svaku sesiju na jednoj strani, a na drugu stranu se prenosi u šifrovanom obliku primenom PKI infrastructure. Nakon ugovaranja ključa šifrovanje podataka se vrši pomoću AES algoritma. Proces rada aSSL rešenja se sastoji od sledećih koraka:

1. Proces se inicijalizuje pozivom od strane klijenta (Web klijenta) ka serveru.
2. Server vraća klijentu javne komponente RSA algoritma.
3. Klijent generiše slučajni 128-bitni ključ, šifruje ga sa javnim ključem servera i takav šifrat vraća serveru.
4. Server prihvata šifrovani 128-bitni sesijski ključ, dešifruje ga preko svog privatnog ključa i vraća klijentu vremenski period u kome će sesijski ključ biti aktivan.
5. Klijent prihvata vreme u kome će važiti ugovoreni sesijski ključ i u tom periodu ga koristi za šifrovanje podataka putem AES algoritma.

S obzirom na to da koncept aSSL-a u velikoj meri odgovara modelu rešenja koje opisuje ovaj rad, to rešenje će u daljem radu biti korišćeno kao referentno rešenje po pitanju razmene ključeva i poređenja performansi. Osnovna razlika između rešenja koje je predloženo u ovom radu i aSSL-a je ta što aSSL koristi javni AES algoritam dok je jedan od osnovnih prostora za rad u našem rešenju primena sopstvenog algoritma.

3. ANALIZA TEHNOLOGIJA

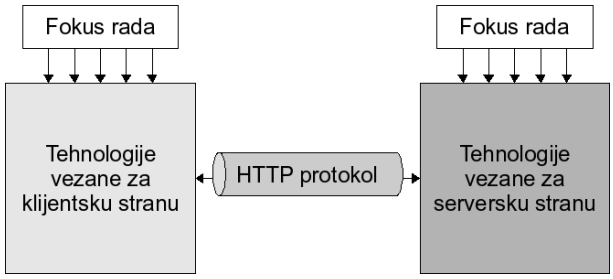
Osnovne komponente Web aplikacija jesu Web klijent, HTTP protokol i Web server. S obzirom na to da predloženo rešenje ne zahteva izmenu HTTP protokola, u ovom delu rada će biti analizirane tehnologije vezane za Web klijente i Web servere, odnosne programski jezici koji omogućavaju modifikaciju (šifrovanje/dešifrovanje) podataka pre slanja, odnosno, nakon prijema.

Tri najpopularnije tehnologija na serverskoj strani jesu jezici za dinamičko kreiranje stranica i Web servisa:

1. PHP (PHP Hypertext Preprocesor)
2. ASP (Active Server Pages)
3. JSP (Java Server Pages)

dok u tri najpopularnije tehnologije za ugrađivanje logike u sadržaj na strani klijenta spadaju:

1. JavaScript
2. Adobe Flash
3. Java apleti



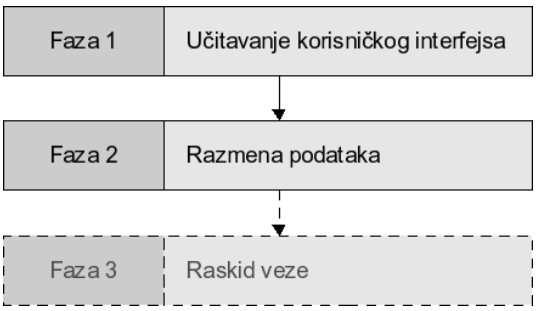
Sl.1. *Komponente Web servisa i fokusi rada*

Gledano na serversku stranu, osnovna prednost PHP i JSP tehnologija jeste platformska nezavisnost tj. mogućnost izvršavanja na različitim operativnim sistemima i pod različitim Web serverima dok je ASP tehnologija usko vezana za operativne sisteme kompanije Microsoft i njihov IIS Web server. U skladu sa tim da suštinske razlike među ovim tehnologijama ne postoje (makar kada su u pitanju aspekti bitni za realizaciju rešenja kojim se bavi ovaj rad) za potrebe ovog rada je odabran PHP programski jezik za realizaciju serverske komponente. Realizacija putem ASP i JSP tehnologija u ovoj fazi istraživanja nije neophodna i kao takva je ostavljena za dalje faze u kojima će biti izvršeno upoređivanje performansi različitih realizacija.

Na klijentskoj strani postoji više tehnologija putem kojih je moguće obaviti šifrovanje zahteva i dešifrovanje odgovora od servera. Na primer, Adobe Flash tehnologija omogućava komunikaciju sa serverskom stranom realizovanom u PHP programskom jeziku putem AMF binarnog formata sa karakteristikama sličnim SOAP protokolu. Korišćenjem Java apleta je takođe moguće ostvariti komunikaciju sa serverskom stranom putem više različitih protokola. Međutim, s obzirom na to je tema ovog rada obezvređivanje AJAX poziva, kao osnovna tehnologija za realizaciju klijentske strane je uzet JavaScript programski jezik. U cilju poređenja performansi ostale klijentske tehnologije će biti obrađene u narednim radovima.

4. MODEL REŠENJA

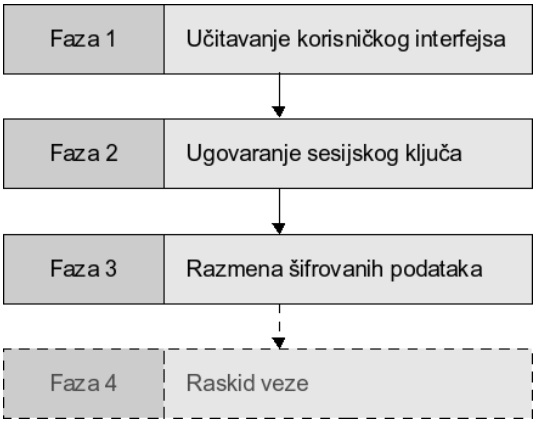
Model rešenja je potrebno izgraditi u skladu sa osnovnim procesom izvršavanja Web aplikacija baziranih na AJAX pozivima. Osnovna razlika između klasičnih Web stranica (statičkih ili dinamičkih HTML dokumenata) i Web aplikacija jeste u tome što se aplikacija najčešće realizuje na taj način da se inicijalno kod klijenta učita samo njen interfejs (faza 1 na slici 2) putem čijih kontrola se vrši slanje narednih zahteva serveru, njihovo prihvatanje i prikaz u korisničkom interfejsu (faza 2 na slici 2). Faza 3 prikazana na slici 2 je u potpunosti opciona u smislu da se već u fazi 2 ostvaruje i raskida veći broj veza te se faza 3 može koristiti za obaveštavanje servera da je potrebno prekinuti sesiju.



Sl.2. *Faze nezaštićene komunikacije putem AJAX-a*

Ovakav pristup, osim što može da ponudi neke dodatne funkcionalnosti, može znatno da smanji količinu prenetog saobraćaja ali može i uticati na povećavanje broja poziva od klijenta ka serveru. Dodatna izmena je i to što se podaci iz faze 2 prenose u različitim formatima - podrazumevani format je XML ali se mogu koristiti i HTML, JSON i sl.

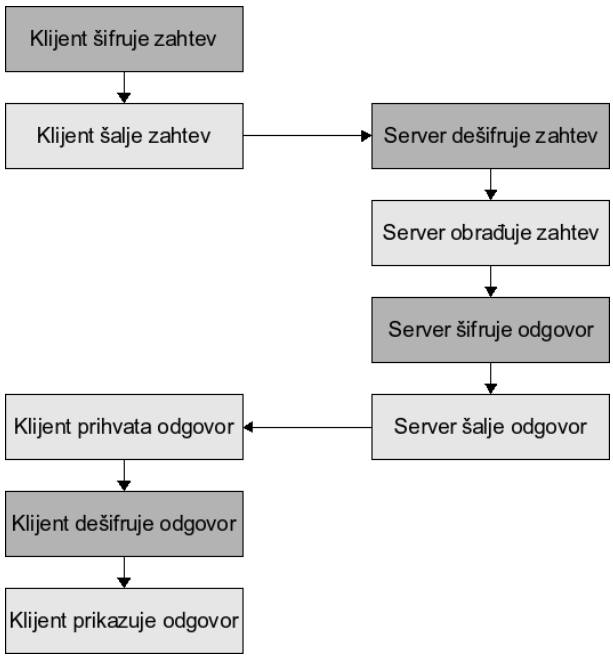
Model predloženog rešenja proširuje osnovni proces dodavanjem faze u kojoj se ugovara sesijski ključ, i to nakon faze učitavanja korisničkog interfejsa a pre faze razmene realnih podataka (slika 3). (Napomena: u ovog fazi razvoja rešenja za svaku od konekcija se generiše zaseban sesijski ključ koji se prenosi kao prvih 16 bajtova šifrata.) Faza 3 na slici 3 (tj. faza 2 na slici 2) je izmenjena na taj način da se svi podaci šifruju pre slanja odnosno dešifruju nakon prijema što podrazumeva ugrađivanje pomenutog sopstvenog algoritma u korisnički interfejs.



Sl.3. *Faze zaštićene komunikacije putem AJAX-a*

Dodatno, za potrebe faze 2 sa slike 3 korisnički interfejs je modifikovan na taj način da sadrži funkcije za ugovaranje sesijskog ključa.

Na slici 4 je prikazana jedna komunikacija iz faze 3 sa slike 3, s tim da su uzete u obzir i klijentska i serverska strana. Koraci označeni tamnijom bojom predstavljaju modifikaciju osnovnog procesa kojom se postiže zaštićen prenos podataka. (Napomena: za potrebe eksperimenta za ovaj rad nije korišćeno šifrovanje zahteva već samo odgovora.)



Sl.4. *Proces zaštićene komunikacije*

5. EKSPERIMENT

Za potrebe ovog rada je realizovano rešenje na osnovu modela opisanog u 4. delu rada. U osnovne nedostatke rešenja i neslaganja sa modelom spadaju:

1. Izvorni kod algoritma se prenosi u izvornom obliku u fazi učitavanja korisničkog interfejsa.
2. Simetrični ključ se prenosi u izvornom obliku u fazi učitavanja korisničkog interfejsa.
3. Za svaki asinhroni poziv se generiše novi sesijski ključ koji se prenosi u vidu prvih 16 bajtova šifrata.

Međutim, 1. i 2. nedostatak izlaze van okvira ovog rada s obzirom da je njegov fokus postavljen na analizu performansi šifrovanja korišćenjem Web tehnologija. Treća stavka se ne može smatrati nedostatkom već drugačijom realizacijom distribucije sesijskog ključa. S obzirom na to da je dodatno opterećenje sistema ovakvom realizacijom minimalno (i u pogledu vremena potrebnog za generisanje slučajnog niza od 16 bajtova, i u pogledu vremena potrebnog za prenos te količine podataka) a u realnoj situaciji može pozitivno uticati na nivo bezbednosti koji rešenje pruža, takva realizacija se može smatrati optimalnom.

Za potrebe eksperimenta je korišćeno okruženje od jednog PC računara sa sledećim hardverskim i softverskim karakteristikama:

- CPU: Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz
- RAM: 2GB
- Slackware Linux 11, kernel 2.6.21
- Apache HTTP server, 1.3.37, PHP 5.2.1
- X11 6.9.0
- Mozilla Firefox 2.0.0.8

S obzirom na to da se procesi na serverskoj i klijentskoj strani Web aplikacija ne odvijaju paralelno, nije bilo potrebe za razdvajanjem softvera na dva zasebna računara.

U eksperimentu je vršeno slanje zahteva serveru za podacima različitih dužina (1KB, 128KB, 256KB, 512KB i 1024KB). Svako preuzimanje je ponovljeno po 3 puta da bi se za rezultat uzele srednje vrednosi. Kod svakog preuzimanja su merena sledeća vremena:

1. vreme potrebno za šifrovanje podataka na serveru;
2. vreme potrebno za prenos šifrovanih podataka;
3. vreme potrebno za dešifrovanje podataka na klijentu;

Rezultati eksperimenta su predstavljeni u 6. delu rada.

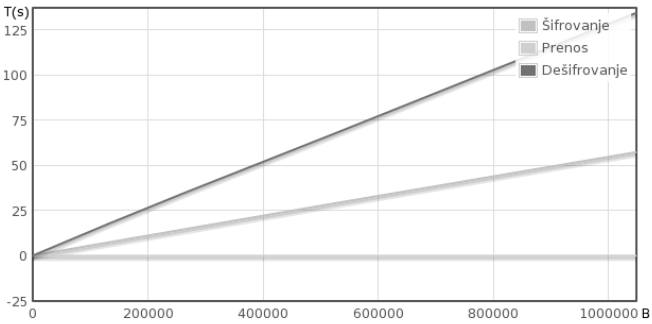
6. REZULTATI

Pre primene sistema za šifrovanje podataka izvršeno je merenje vremena potrebnog za inicijalno punjenje algoritma (dovođenje u radno stanje). Rezultati ovog merenja pokazuju da vreme potrebno za inicijalizaciju algoritma na serverskoj strani iznosi približno 0,142 sekunde dok vreme potrebno za inicijalizaciju algoritma na klijentskoj strani iznosi približno 0,132 skunde.

Rezultati dobijeni na osnovu eksperimenta opisanom u 5. delu rada su predstavljeni u sledećoj tabeli:

Dužina	Šifrovanje	Prenos	Dešifrovanje
1KB	0,201	0,019	0,240
128KB	7,394	0,025	17,627
256KB	14,603	0,028	34,615
512KB	29,016	0,033	67,600
1024KB	57,347	0,050	134,397

Prikazana vremena su izražena u sekundama. Vreme potrebno za prenos podataka nije od većeg značaja u ovom eksperimentu s obzirom na to da je algoritam realizovan tako da se bajtovske dužine originala i šifrata ne razlikuju. Dodatno, eksperiment je izvršen na jednom računaru tako da je vreme prenosa zavisilo samo od brzine internih magistrala računara dok bi se za primenu u realnim uslovima vreme moglo odrediti deljenjem dužine podataka sa brzinom mrežnog linka. Takođe, treba napomenuti i tu da u vremena šifrovanja i dešifrovanja podataka ulazi i ranije opisano vreme za inicijalno punjenje algoritma. Podaci iz tabele su grafički prikazani i na dijagramu (slika 5).



Sl.5. *Odnos bajtovske dužine podataka i vremena potrebnog za šifrovanje, prenos i dešifrovanje*

7. ZAKLJUČAK

U ovom radu je prikazan postupak za realizaciju sopstvenog algoritma za šifrovanje podataka putem tehnologija koje se danas aktivno koriste u izradi Web aplikacija. Dve osnovne tehnologije korišćene za realizaciju jesu PHP i JavaScript programski jezici. Predloženo rešenje je osnova za realizaciju zaštite AJAX poziva primenom sopstvenih kriptografskih mehanizama. Umesto komercijalno dostupnog SSL/TLS protokola, koji predstavlja zatvoreno rešenje, primenjen je sopstveno realizovani algoritam. U eksperimentalnoj analizi praćen je uticaj kriptografskih funkcija na performanse. Evidentan pad performansi se može umanjiti (praktično eliminisati) ako se kriptografske funkcije primene na prenos samo bezbednosno osetljivih podataka koji najčešće čine procentualno zanemarljivi deo ukupnog saobraćaja. Upravo iz tog razloga razmatrani su AJAX pozivi. Predloženo rešenje ima svoju praktičnu primenu i može da garantuje jake kriptografske parametre, umesto slabih i kontrolisanih parametara kod HTTPS poziva.

U daljim istraživanjima analiziraće se distribucija ključeva sa ciljem da rešenje bude fleksibilno i realizibilno. Takođe, u daljim istraživanjima se može izvršiti i poređenje performansi različitih realizacija algoritma tj. realizacija putem ostalih tehnologija nevedenih u radu.

LITERATURA

[1] Aleksandar Jevremović, Mladen Veinović, “Zaštita podataka na IP nivou pod Linux OS”, ETRAN, 2006. Beograd

[2] Jevremović A., Veinović M., „IPsec - analiza uticaja algoritma za šifrovanje na saobraćaj u LAN mrežama", 14. telekomunikacioni forum Telfor 2006., Beograd, 2006., CD izdanje

[3] http://assl.sullof.com/assl/

Abstract: This paper is about cryptographic data protection in Web applications based on AJAX http requests. Using of custom algorithm and controlling of cryptographic parameters represent the suggested solution. The dependence on available commercial (SSL/TLS based) transport layer protocols is also avoided. Different client server platforms are considered and analyzed in this paper. The influence of cryptographic functions on the performances are separate discussed in the results. The main motive of this research is using of solution in the secured transfer of confidential data during the sign-in phase (such as PIN, password, credit card number etc.). The previous works, connected with this solution, are focused on integration of security mechanisms in Linux OS kernel. This work is on the upper level, and solution is relocated on the application layer – communications between different Web server and client platforms. Nevertheless the measured performances are worse, the AJAX popularity and the growth of computer’s performances, make possible using of described solution. The key and algorithm distribution are not covered by this paper. The design, implementation and algorithm performances represent the main scope of the research.

SECURING AJAX CALLS BY USING OWN CIPHER

Aleksandar Jevremović, Mladen Veinović, Goran Šimić

BESPLATNI GOTOVI SEMINARSKI, DIPLOMSKI I MATURSKI RAD.

RADOVI IZ SVIH OBLASTI, POWERPOINT PREZENTACIJE I DRUGI EDUKATIVNI MATERIJALI.

WWW.SEMINARSKIRAD.ORG

WWW.MATURSKIRADOVI.NET

WWW.MATURSKI.NET

WWW.SEMINARSKIRAD.INFO

WWW.MATURSKI.ORG

WWW.ESSAYSX.COM

WWW.FACEBOOK.COM/DIPLOMSKIRADOVI

NA NAŠIM SAJTOVIMA MOŽETE PRONAĆI SVE, BILO DA JE TO SEMINARSKI, DIPLOMSKI ILI MATURSKI RAD, POWERPOINT PREZENTACIJA I DRUGI EDUKATIVNI MATERIJAL. ZA RAZLIKU OD OSTALIH MI VAM PRUŽAMO DA POGLEDATE SVAKI RAD, NjEGOV SADRŽAJ I PRVE TRI STRANE TAKO DA MOŽETE TAČNO DA ODABERETE ONO ŠTO VAM U POTPUNOSTI ODGOVARA. U BAZI SE NALAZE GOTOVI SEMINARSKI, DIPLOMSKI I MATURSKI RADOVI KOJE MOŽETE SKINUTI I UZ NJIHOVU POMOĆ NAPRAVITI JEDINSTVEN I UNIKATAN RAD. AKO U BAZI NE NAĐETE RAD KOJI VAM JE POTREBAN, U SVAKOM MOMENTU MOŽETE NARUČITI DA VAM SE IZRADI NOVI, UNIKATAN SEMINARSKI ILI NEKI DRUGI RAD RAD NA LINKU IZRADA RADOVA. PITANjA I ODGOVORE MOŽETE DOBITI NA NAŠEM FORUMU ILI NA MATURSKIRADOVI.NET@GMAIL.COM