UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
NOVI SAD

Grupa 22
Maja Ožegović E3 111/2014
Danijel Merli PR 63/2016
Nikolina Lazović PR 140/2017
Jelena Stanojoski PR157/2017

Zadatak 12

Sigurnost i bezbednost u elektroenergetskim sistemima
- Primenjeno softversko inženjerstvo -

Novi Sad, 07.01.2021.

SADRŽAJ:

1. OPIS REŠAVANOG PROBLEMA

Cilj projekta jeste implementacija servisa koji ima ulogu *Domen Kontroler* (DC) komponente koja predstavlja treću stranu od poverenja prilikom uspostavljanja bezbedne komunikacije između klijenta i servisa. Domen kontroler se sastoji od dve komponente: *Authentication Service* (AS) i *Ticket Granting Service* (TGS).

Projekat se može podeliti na tri dela.

Prvi deo je proces autentifikacije klijenta i servisa. Proces autentifikacije se obavlja *challenge-response* protokolom. DC dobija zahteve za autentifikaciju od klijenta i servisa. DC prosleđuje zahtev *Authentication service-u* koji, ukoliko klijent tj. servis postoji u njegovoj bazi, vraća klijentu tj. servisu *challenge(16-bitni broj).* Klijent tj. servis će vratiti DC-u *response,* koji se dobija tako što se spoji *hash* vrednost korisničke šifre i dobijeni *challenge* i nad tim nizom bajtova se odradi *hash* funkcija. Nakon što primi *response,* DC prosleđuje tu vrednost AS-u, zajedno sa proslatim *challenge-om* i korisničkim imenom. AS će na osnovu korisničkog imena tj. na osnovu *hash* vrednosti korisničke šifre, odraditi isti proces – spojiti *hash* vrednost šifre i *challenge* i tada *hash* funkcijom izračunati vrednost koju će porediti sa odgovorom koji je klijent tj. servis poslao. Ukoliko se izračunata vrednost i dobijeni odgovor poklapaju, klijent tj. servis se smatra autentifikovanim.

U drugom delu zadatka, u slučaju uspešne autentifikacije, DC prosleđuje zahteve TGS komponenti. Ukoliko je servis uspešno autentifikovan, TGS će obaviti validaciju servisa tako što će proveriti da li postoji servis u DNS tabeli. TGS takođe vodi evidenciju o startovanim servisima. Nakon uspešne autentifikacije klijenta, zahtev se takođe prosleđuje TGS komponenti. U ovom slučaju TGS proverava da li postoji zahtevani servis u domenu i da li je taj servis startovan. Ukoliko servis postoji i startovan je, TGS generiše ključ sesije. Kako bi se obezbedila poverljivost prilikom razmene ključa sesije, ključ sesije se enkriptuje 3DES algoritmom i koristi se *hash* vrednost korisničke šifre kao deljeni tajni ključ. DC prvo prosleđuje servisu enkriptovani ključ sesije zajedno sa korisničkim imenom klijenta za kojeg važi, a nakon uspešnog slanja servisu tj. potvrde da je servis i dalje aktivan, prosleđuje ključ sesije i adresu servisa klijentu.

U trećem delu zadatka, klijent i servis dekriptuju primljeni ključ sesije 3DES algoritmom. Ovaj ključ sesije koriste kao deljeni tajni ključ kojim enkriptuju tj. dekriptuju poruke u toku njihove komunikacije.

Servis je implementiran tako da čuva ključ sesije za više klijenata. Pri pozivu metoda servisa, servis proverava koji klijent je pozvao metodu tj. da li ima ključ sesije za tog klijenta. U koliko ima ključ sesije za tog klijenta, servis smatra da je klijent autentifikovan jer je dobio ključ sesije i korisničko ime od DC servisa i nastavlja komunikaciju sa klijentom.

DC loguje u specifičnom *Windows Event log-u* bezbednosne događaje koji se odnose na uspostavljanje komunikacije:

Uspešna ili neuspešna autentifikacija klijenta tj. servisa na AS komponenti

Uspešna ili neuspešna validacija servisa na TGS komponenti

2. TEORIJSKE OSNOVE

Bezbednosni mehanizmni korišćeni pri izradi projekta su autentifikacija, auditing, kriptovanje implementirano 3DES algoritmom u CBC modu.

Autentifikacija je proces u okviru koga korisnik ili izvor informacija dokazuju da su to za šta se predstavljaju. U projektu se identitet potvrđuje prvo *Windows authentication* protokolom koji je zasniva na *NTLM*-u (NT Lan Manager). Dodatno, klijent i servis potvrđuju svoj identitet po principu *challenge-response* protokola. Ovi protokoli omogućuju autentifikaciju bez slanja poverljivih podataka (šifre).
Kao deo obostrane autentifikacije, neophodno je podesiti identitet servisa koji klijent i servis očekuju prilikom uspostavljanja komunikacije sa DC komponentom. Za to je potrebno proslediti *EndpointIdentity* prilikom uspostavljanja komunikacije.

Auditing predstavlja vođenje istorijata aktivnosti. Ovim bezbednosnim mehanizmom se može ustanoviti u kom momentu i zašto je nastao problem u sistemu.

Kriptovanje algoritmom 3DES se koristi za poverljivost prilikom razmene kluča sesije kao i prilikom razmene podataka između klijenta i servisa. Triple-DES je poboljšana verzija DES algoritma koja podrazumeva kriptovanje podataka DES algoritmom u tri prolaza sa dva različita ključa od 64 bita. Uvođenjem dodatnog 64-bitnog ključa dobija se ukupna dužina ključa 128 bita (odnosno 112 bita efektivno), čime je prevaziđen problem *exhaustive key search* napada na DES.

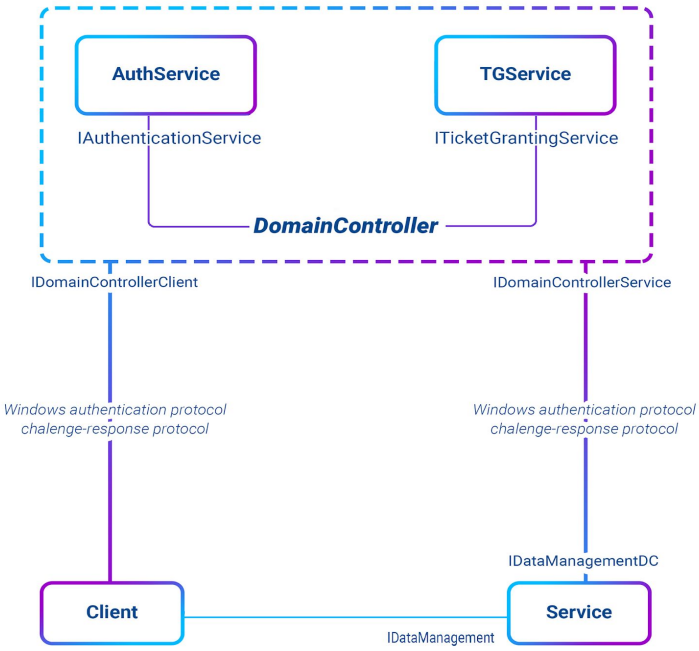
Cipher Block Chaining (CBC) mod se zasniva na randomizaciji ulazne poruke tako da isti blok podataka neće dati istu šifrovanu poruku. Naime, uvodi se inicijalni vektor (IV) tako da se nad prvim blokom podataka pre kriptovanja primenjuje XOR operacija sa vrednošću IV. Random vrednost kojom se XOR-uje svaki sledeći blok biće izlaz (šifrovana poruka) iz prethodnog bloka. Inicijalni vektor je random vrednost koja ne mora da se čuva u tajnosti, bitno je samo da vrednost bude nepredvidiva i da se svaki put koristi drugačija vrednost. Da bi poruka bila dekriptovana, potrebno je da strana koja dekriptuje, osim algoritma i tajnog ključa, zna i IV vrednost. Slanje inicijanog vektora je moguće na različite načine, npr. može se slati zajedno sa šifrovanom porukom s obzirom da IV ne mora da se čuva u tajnosti.

3. DIZAJN IMPLEMENTIRANOG SISTEMA

Implementacija rešenja se sastoji od sledećih komponenti:

- DomainController
- Service
- Client
- Contracts

Na slici 1 su prikazane komponente sistema zajedno sa interfejsima koje one implementiraju i čije servise izlažu. Takođe na slici su prikazani i načini komunikacije između svih komponenti.



Slika 1. - Komponente sistema.

DomainController predstavlja treću stranu od poverenja prilikom uspostavljanja bezbedne komunikacije *Client* i *Service* komponente. Implementira dva interfejsa – *IDomainControllerClient* koji je dostupan *Client* komponenti i *IDomainControllerService* koji je dostupan *Service* komponenti. *DomainController* je implementiran tako da čuva podatke *Client* tj. *Service* komponente u sesiji. Komunicira sa njima putem *Windows authentication protocol-a* i dodatnu potvrdu identiteta radi putem *challenge-response protocol-a.*

DomainController je implementiran tako da ujedno pokreće i *AuthService* i *TGService* – servisi za autentifikaciju *Client* i *Service* komponenti, validaciju *Service* komponenti i za generisanje ključa sesije koji će *Client i Service* dalje koristiti u međusobnoj komunikaciji.

DomainController čuva podatke neophodne za ceo proces autentifikacije i generisanje ključa sesije u sesiji.

Nakon primljenog zahteva za autentifikaciju, bilo od *Client* ili *Service* komponente, prosleđuje korisničko ime *AuthService* komponenti. Ukoliko *AuthService* čuva podatke za tog klijenta, tj. može da ga autentifikuje, vraća izgenerisani 16-bitni broj – *challenge* - koji će *DomaincController* da prosledi *Client* tj. *Service* komponenti i ujedno ga sačuvati u sesiji tog korisnika.

U drugom delu challenge-response protokola, *Client* tj. *Service* komponente šalju *response.* On se dobija tako što se spoji *hash* vrednost korisničke šifre i dobijeni *challenge* i nad tim nizom bajtova se odradi *hash* funkcija. Ovim se izbegava slanje korisničke šifre. *DomainController* tada prosleđuje korisničko ime, poslati *challenge* i primljeni *response* servisu za autentifikaciju. *AuthService* tada uzima iz baze podataka *hash* vrednost korisničke šifre i ponavlja isti proces kao i klijent kako bi izračunao *response.* Ukoliko se primljeni *response* i vrednost koju je

AuthService izračunao poklapaju, *AuthService* prosleđuje *DomainController* komponenti da je korisnik autentifikovan.

Sledeći koraci *DomainController* komponente se razlikuju u odnosu na to ko je poslao zahtev za autentifikaciju.

U slučaju *Service* komponente, *DomainController* treba da validira servis koji *Service* komponenta želi da pokrene. Tada *DomainController* komunicira sa *TGService* komponentom. *TGService* proverava da li zahtevani servis koji *Service* komponenta želi da pokrene postoji u DNS tableli. Ukoliko postoji *TGService* će vratiti punu adresu tog servisa, ukoliko ne postoji smatra se da *Service* komponenta želi da pokrene servis koji nije podržan tako da će validacija *Service* komponente biti neuspešna. U slučaju uspešne validacije *DomainController* će poslati još jedan zahtev *TGService* servisu – startovanje servisa. Ovaj zahtev dodaje validirani servis u evidenciju startovanih servisa. Time se završava validacija *Service* komponente i *DomainController* šalje potvrdu *Service* komponenti da je sve u redu.

U slučaju *Client* komponente, *DomainController* treba da proveri da li zahtevani servis postoji i da li je taj servis aktivan. *DomainController* će od *TGService* tražiti te informacije i ukoliko zahtevani servis postoji i startovan je *DomainController* će pokrenuti proces generisanja ključa sesije. Generisanje ključa sesije se obavlja u *TGService* komponenti. Kluč sesije je neophodno poslati prvo *Service* komponenti. Izgenerisani ključ sesije se enkriptuje 3DES algoritmom. Onaj koji se šalje *Client* komponenti se enkriptuje koristeći tajni ključ korisnika koji je pokrenuo *Client* komponentu (*hash* vrednost korisničke šifre), dok se onaj koji se šalje *Service* komponenti enkriptuje koristeći tajni ključ korisnika koji je pokrenuo *Service* komponentu (*hash* vrednost korisničke šifre). *DomainController* će prvo pokušati da pozove metodu za slanje ključa sesije *Service* komponenti iz *IDataManegementDC* interfejsa koji implementira *Service* komponenta. Ukoliko *DomainController* ne može da ostvari komunikaciju sa *Service* komponentom, smatraće se da taj servis više nije aktivan i ta informacija će se takođe proslediti i *Client* komponenti. Ukoliko su ključ sesije i korisničko ime klijenta za kojeg važi taj ključ uspešno poslati, *DomainController* će vratiti *Client* komponenti neophodne informacije za komunikaciju sa zahtevanim servisom – enkriptovani ključ sesije i adresu zahtevanog servisa.

DomainController komponenta pored ispisa svakog koraka na konzoli takođe loguje u specifičnom *Windows event log-u* bezbednosne događaje koji se odnose na uspostavljanje komunikacije – uspešna ili neuspešna autentifikacija *Client* tj. *Service* komponente kao i uspešna ili neuspešna validacija servisa.

Service je komponenta koja je prvo u ulozi klijenta u odnosu na *DomainController* komponentu. Šalje zahtev za autentifikaciju u kome takođe šalje ime servisa koji želi da pokrene. Komuniciraju putem *Windows authentication protocol-a.* Dodatna potvrda identiteta se obavlja putem *challenge-response* protokola, gde *DomainController* šalje *Service* komponenti *challenge* ukoliko se username servisa nalazi u bazi *AuthenticationService* komponente. Ukoliko je *Service* u sledećem koraku *challenge-response* protokola poslao validne podatke, smatra se da je autentifikovan.

U daljoj komunikaciji *Service* komponenta je u ulozi servera u odnosu na *DomainController*. *Service* komponenta implementira interfejs *IDataManagementDC* koji je dostupan *DomainController* komponenti. Nakon što je servis autentifikovan i startovan, *Ser*vice može da

prima zahteve od *Client* komponente, ali pre toga je neophodno proslediti ključ sesije koji važi za određenog klijenta. Kada se pojavi klijent koji želi da stupi u kontakt sa *Service* komponentom, nakon njegove uspešne autentifikacije, *DomainController* će poslati ključ sesije i ime klijenta za kojeg važi taj ključ *Service* komponenti. Ukoliko *DomainController* ne može da stupi u kontakt sa *Service* komponentom smatraće da *Service* više nije aktivan i neće mu prosleđivati ključeve sesija dok se *Service* ponovo ne pokrene i autentifikuje.

Service komponenta implementira *IDataManagement* interfejs u kojem se nalaze metode dostupne *Client* komponenti. Nude se dve metode – *Read()* i *Write()* koje služe za rad sa bazom podataka. *Service* sa *Client* komponentom komunicira takođe preko *Windows authentication protocol-a* i dodatno koristi *3DES* algoritam enkripcije u CBC modu. *Service* komponenta čuva ključ sesije za svakog klijenta tako da će, pre nego što odradi akcije zahtevane od *Client* komponente, proveriti da li ima ključ sesije od klijenta koji se javio. Ukoliko ima nastavlja komunikaciju sa tim klijentom.

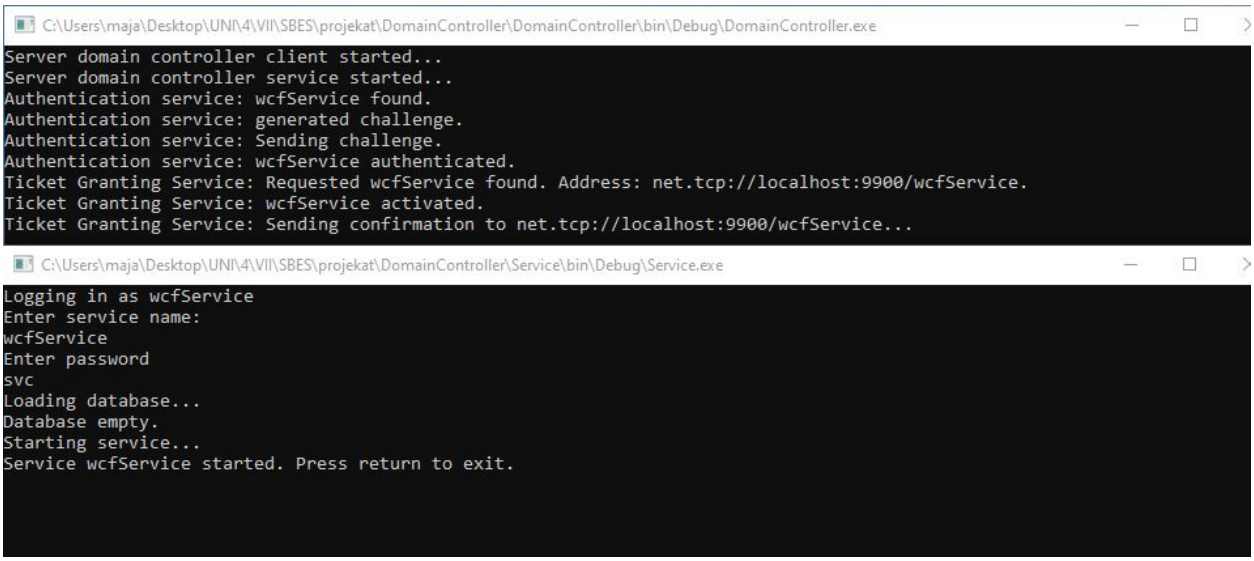
Client je komponenta koja komunicira sa *DomainController* i *Service* komponentama. *Client* šalje zahtev za autentifikaciju *DomainController* komponenti i ujedno šalje i ime servisa. Nakon uspešne autentifikacije i provere servisa na *DomainController* komponenti, koristi dobijenu adresu servisa i ključ sesije za dalju komunikaciju sa *Service* komponentom. *Client* poziva metode za rad sa bazom podataka koje su implementirane na *Service* komponenti preko *IDataManagement* interfejsa. U okviru ovih metoda podaci se prilikom razmene kriptuju primenom 3DES algoritma u CBC modu koristeći dobijeni ključ sesije kao tajni ključ. *Client* komponenta veruje *Service* komponenti jer je adresu servisa dobila od *DomainController* komponente. Takođe je implemetiran meni na strani klijenta tako da korisnik koji je pokrenuo *Client* komponentu može da bira metode – pritisak *w* dugmeta označava pozivanje metode *Write(),* pritisak *r* dugmeta označava pozivanje metode *Read(),* dok se pritiskom dugmeta *x* prekida rad konzolne aplikacije.

Contracts je biblioteka sa interfejsima i pomoćnim klasama. Svi interfejsi koji se koriste za komunikaciju između *DomainController* komponente i *Client* komponente, *DomainController* i *Service* komponente kao i *Client* komponente i *Service* komponente se nalaze u ovoj biblioteci. Od pomoćnih klasa sadrži *Formatter* klasu koja služi za parsiranje korisničkog imena korisnika koji je pozvao određenu metodu nekog od servisa. Ovde se nalaze i klase koje se koriste za kriptovanje - *ChallengeResponse* klasa služi za kreiranje *response* dela u *challenge-response protocol-u,* dok *3DESAlgorithm* klasa nudi metode šifrovanja i dešifrovanja podataka.

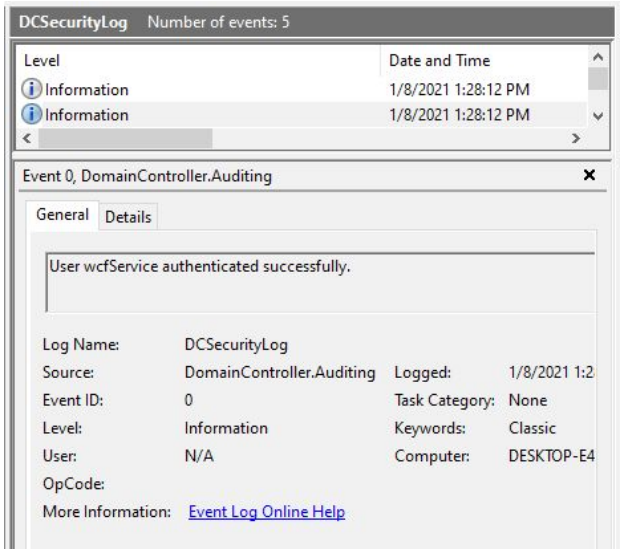
4. TESTIRANJE SISTEMA

Pozitivni test scenariji:

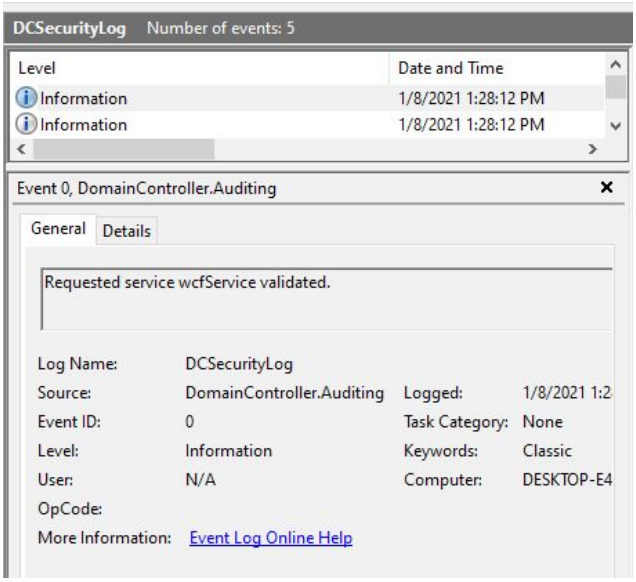
- Uspešna autentifikacija servisa i uspešna validacija servisa, slika 2.1, a potom upisana poruka u DCSecurityLog, slika 2.2 i slika 2.3.



Slika 2.1 – Ispis na konzoli DomainController komponente i Service komponente, nakon uspešne autentifikacije i validacije servisa.

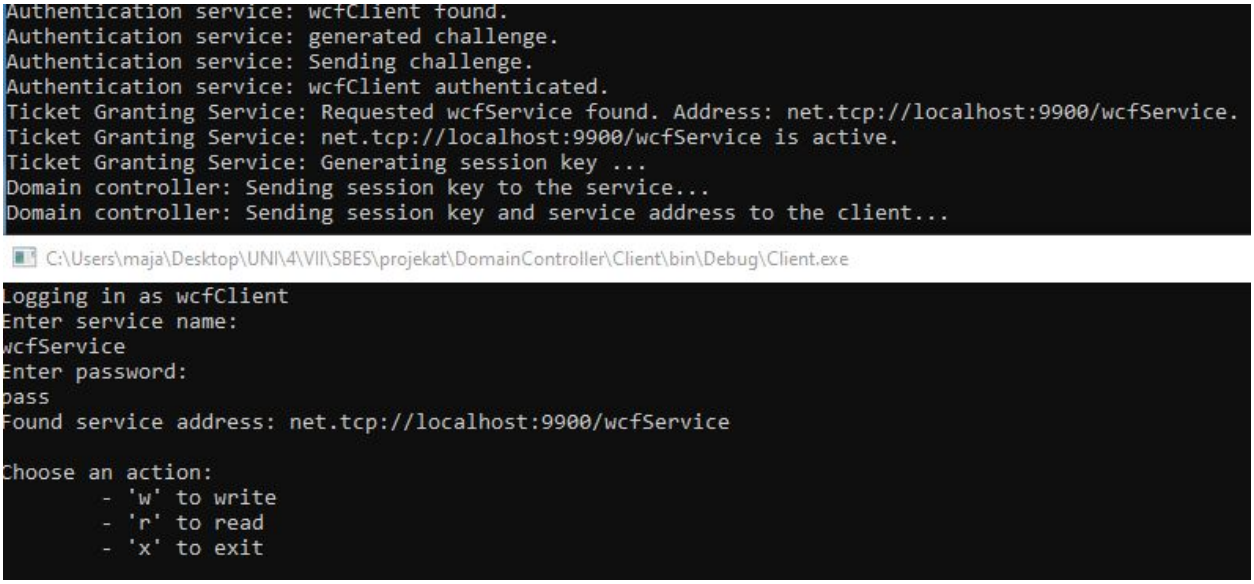


Slika 2.2 – Prikaz zapisa u DCSecuritylog-u u slučaju uspešne autentifikacije servisa.

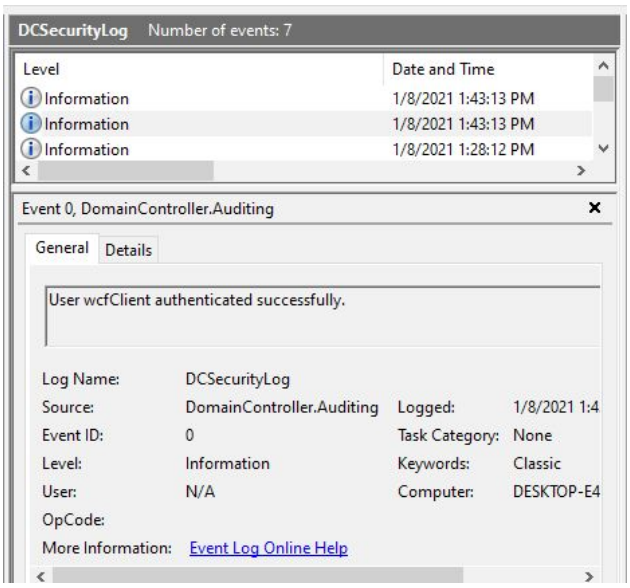


Slika 2.3 – Prikaz upisa u DCSecurityLog u slučaju upešne validacije servisa.

- Primer uspešne autentifikacije klijenta, klijent dobija adresu servisa i ključ sesije, slika 2.4. Prikaz upisa u DCSecurityLog nakon uspešne autentifikacije klijenta je na slici 2.5.

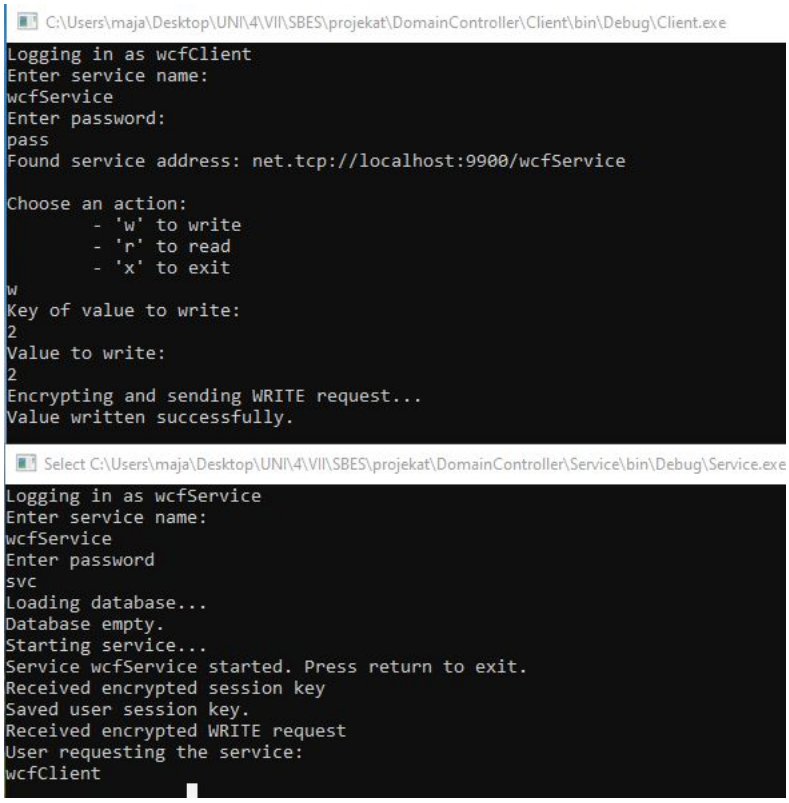


Slika 2.4 – Uspešna autentifikacija klijenta, uspešno primljena adresa zahtevanog servisa i ključ sesije.

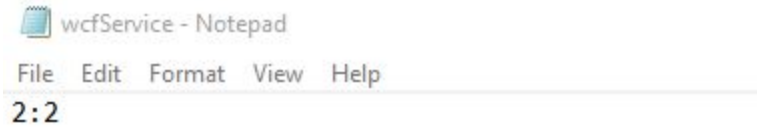


Slika 2.5 – Prikaz uspešnog upisa u DCSecurityLog nakon uspešne autentifikacije klijenta.

- Prikaz uspešne komunikacije autentifikovanog klijenta i autentifikovanog servisa. Klijent šalje zahtev za upis podataka, slika 2.6. Servis pristigle podatke dekriptuje odgovarajućim ključem sesije i upisuje ih u bazu podataka, slika 2.7.



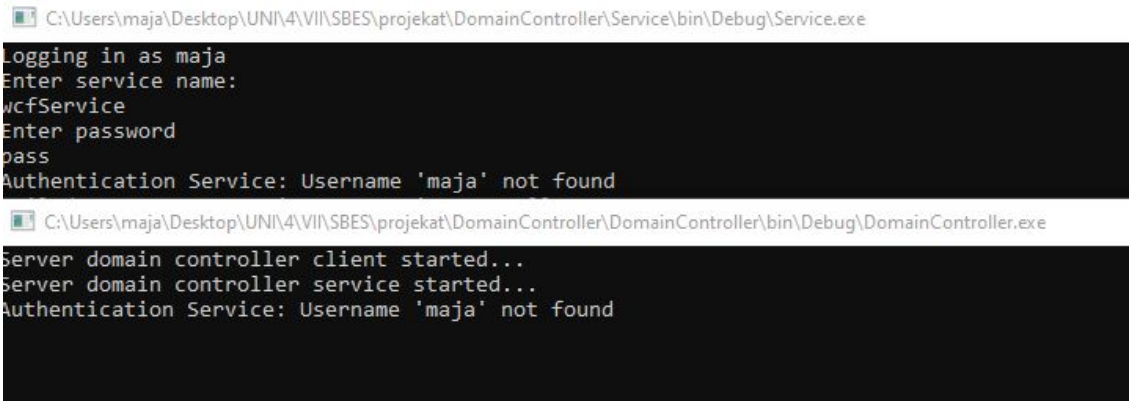
Slika 2.6 – Prikaz klijentskog poziva *Write* metode servisa. Klijent i servis su prethodno autentifikovani.



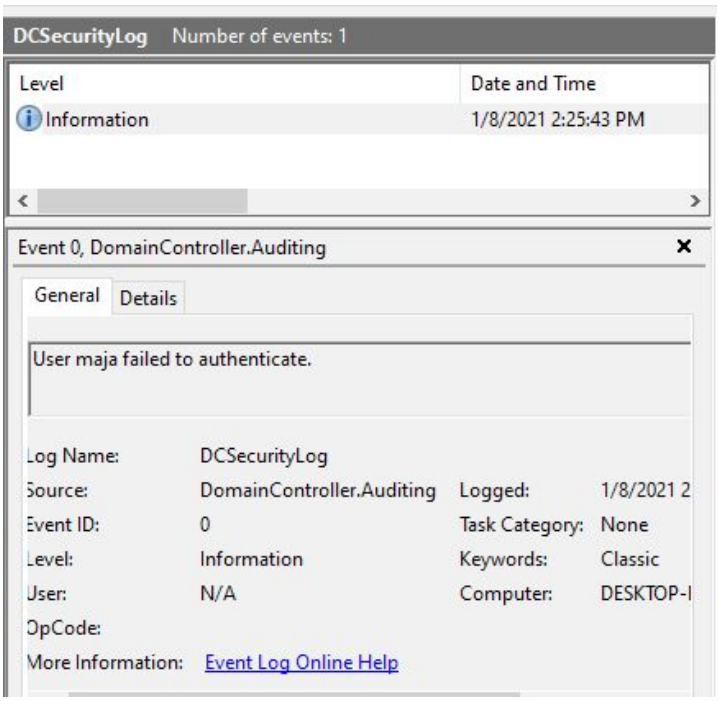
Slika 2.7 – Podaci su uspešno dešifrovani u upisani u tekstualni fajl na serveru.

Negativni test scenariji:

- Neuspešna autentifikacija servisa. Servis se pokreće preko naloga koji ne postoji u bazi koja se nalazi u AuthenticationService komponenti, slika 2.8. DC uspešno upisuje u DCSecurityLog neuspeli pokušaj autentifikacije, slika 2.9.

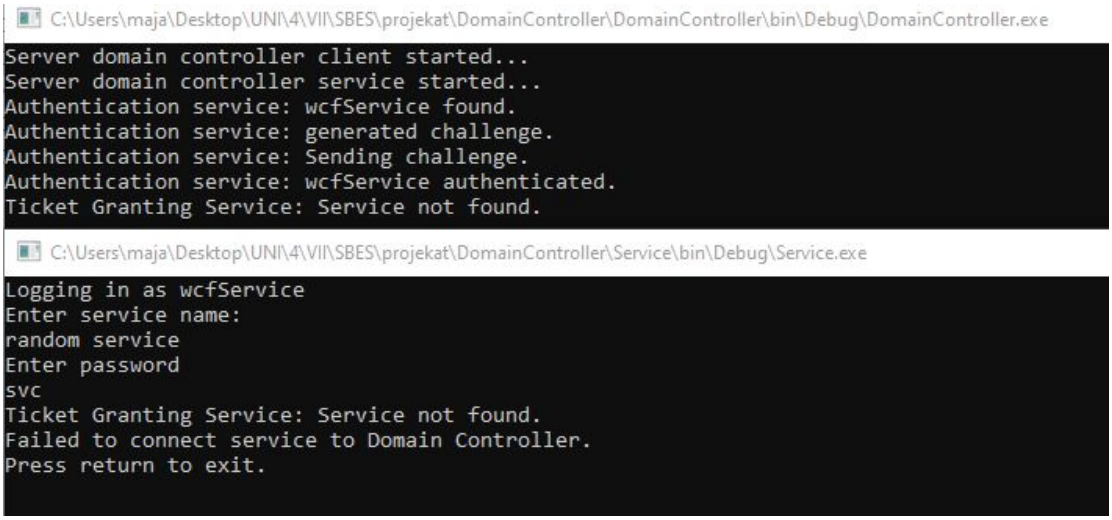


Slika 2.8 – Neuspešna autentifikacija servisa.

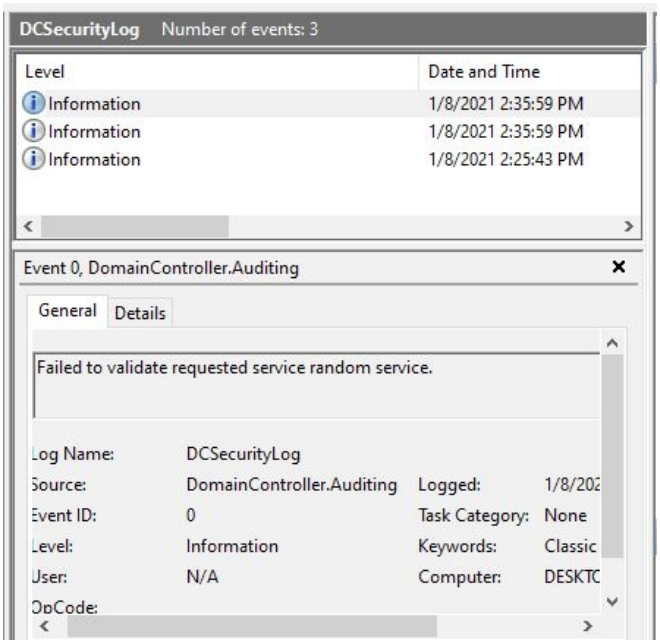


Slika 2.9 - Logovanje neuspešnog pokušaja autentifikacije u DCSecurityLog-u.

- Neuspešna validacija servisa. Autentifikovani servis traži od DomenController komponente validaciju servisa – random service. Validacija nije uspešna kao što može da se vidi na slici 2.10. Takođe je prikazano logovanje neuspešne validacijie servisa u DCSecurityLog-u, slika 2.11.

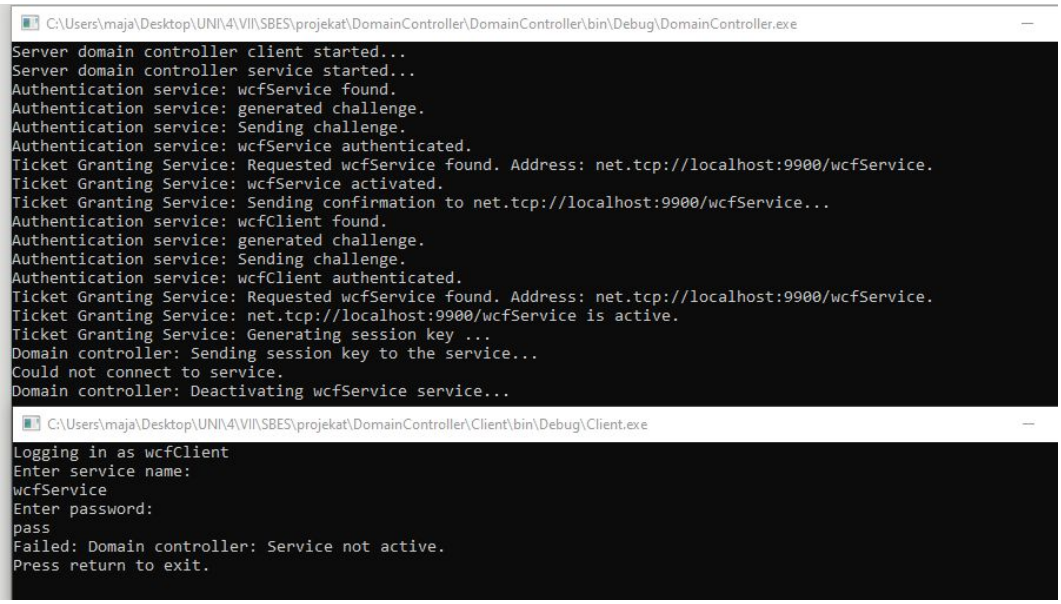


Slika 2.10 – Neuspešna validacija servisa.



Slika 2.11 – Neuspešna validacija servisa upisana u DCSecurityLog.

- Servis je uspešno autentifikovan i validiran ali se u međuvremenu ugasi. DomainController komponenta ne zna da je servis ugašen dok se ne pojavi klijent koji zahteva taj servis. Na slici 2.12 je prikazano da će DomainController pokušati da pošalje ključ sesije i da će ga označiti kao servis koji više nije aktivan.



Slika 2.12 - Prikaz ispada autentifikovanog i validiranog servisa.