



UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U
NOVOM SADU



Fakultet Tehničkih nauka

Sigurnost i bezbednost u Smart Grid sistemima

Projektni zadatak 3 - Dokumentacija

Novi Sad, 2017

1. SISTEM

U ovom odeljku se nalazi kratak opis projektnog zadatka, dizajn sistema, kao i arhitektura sistema.

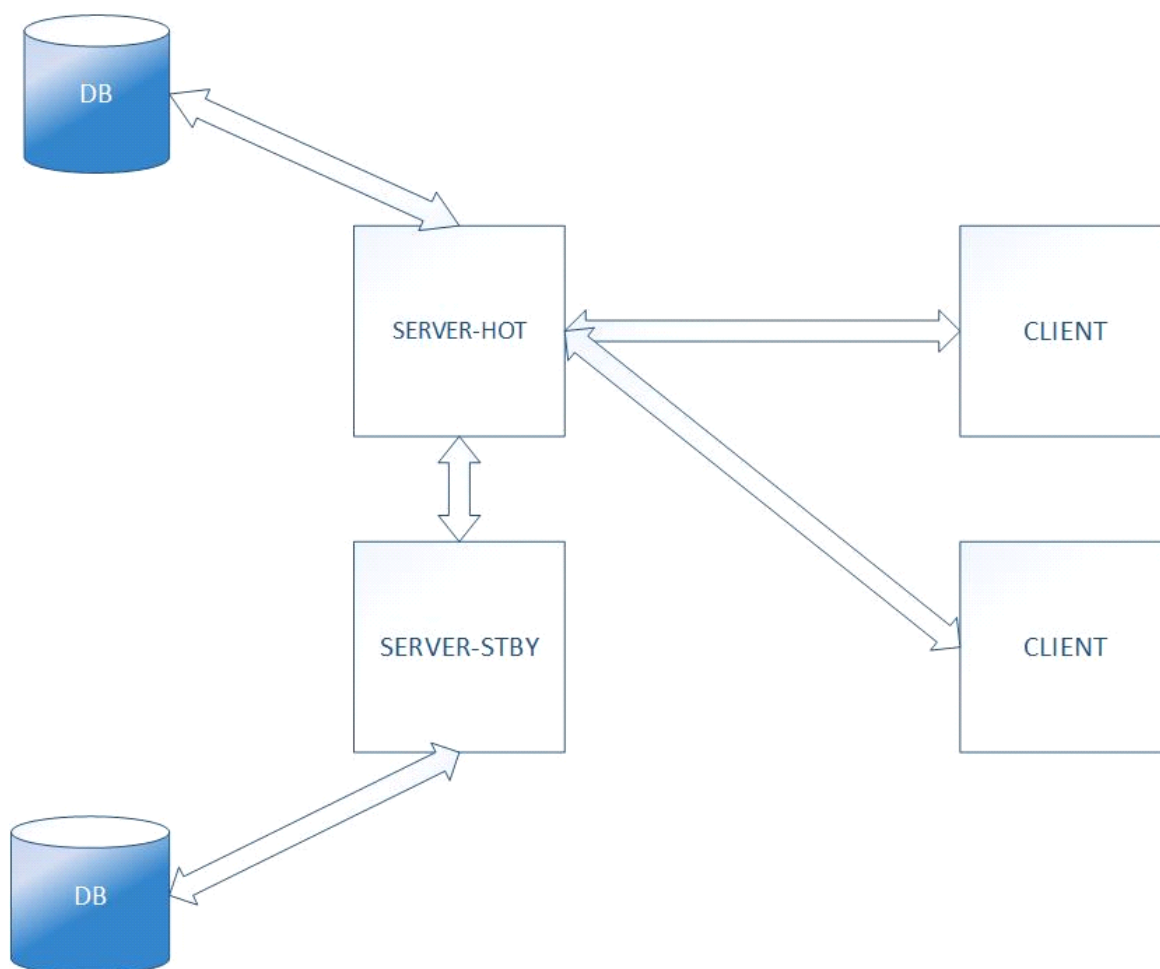
1.1 OPIS SISTEMA

Implementirati klijent – servis model koji simulira anti – malware.

- Anti – malware servis je komponenta kojoj klijenti pristupaju sa zahtevom za proveru potpisa određene aplikacije koja je izvršava. Autentifikacija između servera i klijenta se vrši pomoću sertifikata. Komunikacija se vrši preko TCP protokola.
- Klijentska aplikacija ima dve uloge :
 - Praćenje aplikacija i procesa u izvršavanju aplikacija. Kada se aplikacija pokreće, klijentska aplikacija treba da preuzme digitalni potpis aplikacije koja se pokreće i da pošalje Anti – malware servisu koji treba da vrati odgovor da li je aplikacija koja se pokreće bezbedna ili je zaražena.
 - Druga uloga je mogućnost prosleđivanja putanje do određenog fajla na disku koji klijentska aplikacija treba da proveri, da preuzme digitalni potpis i da pošalje Anti – malware servisu koji treba da vrati odgovor da li je aplikacija koja se pokreće bezbedna ili je zaražena.
- Prilikom slanja podataka od klijenta ka servisu podaci treba da budu kodirani 3DES algoritmom, tj. ne šalju se u obliku otvorenog teksta.
- Implementirati 3DES algoritam u CBC modu. Prilikom izrade zadatka voditi računa o mogućnostima paralelizacije operacija.
- Potpis aplikacije koja se proverava predstavlja heš vrednost, i kao takva heš vrednost se čuva u jedinstvenoj bazi podataka (MySQL, SQLite, Mongo...) kojoj pristupa servis.
- Sve akcije klijentske aplikacije treba da budu logovane u Windows Event Log-u, kao i odgovor anti – malware servisa o bezbednosti aplikacije koja je predmet analize.
- Dodatno, server treba da omogući Failover funkcionalnost. Potrebno je obezbediti redundantnu server komponentu koja u slučaju nedostupnosti primarnog servera može da preuzme njegovu ulogu :
 - Održavanje integriteta podataka između primarne i backup komponente.
- Mogućnost klijenta da komunicira sa novom server komponentom nakon failovera.

1.2 ARHITEKTURA SISTEMA

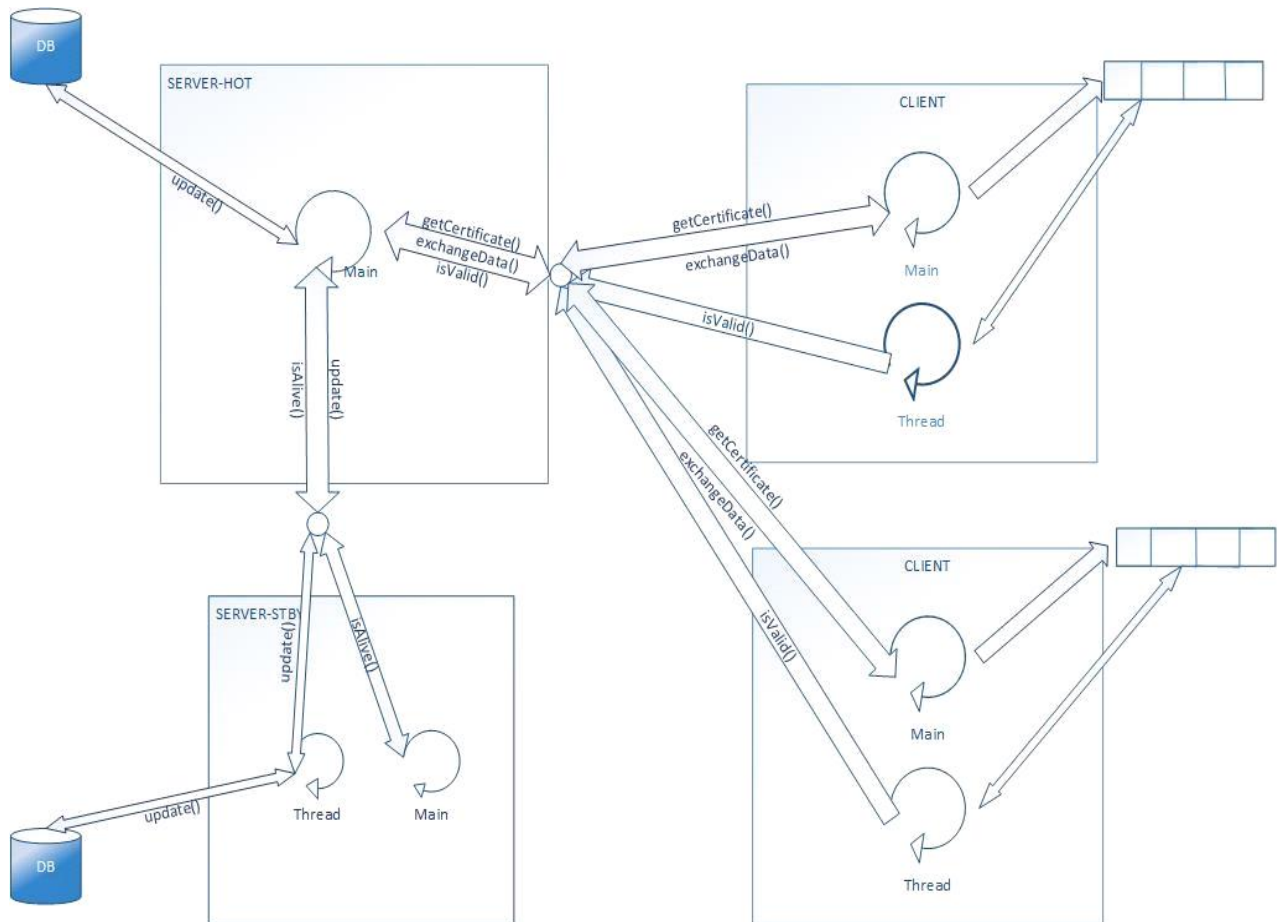
Arhitekturu sistema čine aktivni server (hot server), backup server (standby server), klijentske aplikacije, i baze podataka (po jedna baza za svaki server). U bazi podataka se nalazi tabela sa heševima aplikacija (procesa) koje nisu zaražene, tj. koje su bezbedne, kao i odgovarajuća imena aplikacija. Klijent može da unese putanju do određenog fajla na disku, ili da pokrene željeni proces na računaru, koji treba da se proveri da li je zaražen. U tom trenutku se proverava da li se heš vrednost aplikacije nalazi u white listi u bazi, tj. da li je aplikacija bezbedna. U zavisnosti od ishoda pretrage baze klijentu se javlja da li je aplikacija bezbedna ili nije. Na jedan server je moguće nakačiti proizvoljan broj klijenata. Naravno samo klijenti sa odgovarajućim sertifikatima mogu da se povežu sa servisom. Ukoliko iz nekog razloga aktivni server padne ili postane nedostupan, tada će njegovu ulogu da preuzme backup server, tj. učiniće se prevezivanje. Podržana je funkcionalnost održavanja integriteta podataka između aktivnog i backup servera. Na slici 1.1 je prikazana arhitektura sistema.



Slika 1.1 Arhitektura sistema

1.3 DIZAJN SISTEMA

Dizajn sistema sadrži detaljan model, tj. osnovne komponente sistema, tip komunikacije, kao i tok podataka između njih, što je prikazano na slici 1.2.



Slika 1.2 Dizajn sistema

2. STRUKTURE PODATAKA

U ovom poglavlju su prikazane strukture podataka koje su korišćene za implementaciju sistema.

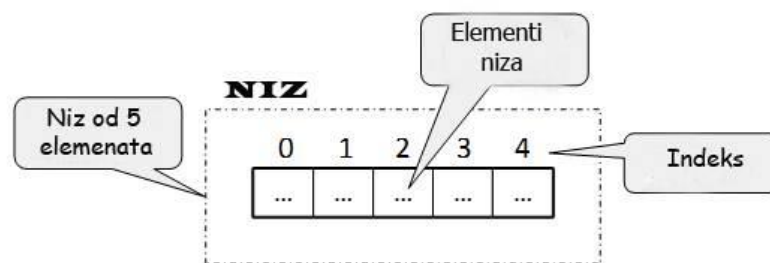
2.1 Dictionary

Dictionary je možda i najčešće korišćena kolekcija u C# koja predstavlja asocijativni kontejner. Asocijativna iz razloga što se prilikom skladištenja podacima dodeljuje ključ (key) koji služi za manipulisanje podacima u kolekciji. Dictionary važi za najbržu asocijativnu kolekciju jer u osnovi koristi HashTable strukturu. To znači da su vrednosti ključeva hash vrednosti, čime se znatno poboljšavaju performanse u radu sa ovom vrstom kolekcija. Razlika između HashTable i Dictionary kolekcije je u tome što je dictionary generički tip, što dictionary čini type safe strukturom, odnosno nije moguće dodati slučajan tip elementa u kolekciju, a samim tim nije potrebno kastrovanje podataka prilikom čitanja elemenata iz kolekcije. Vreme potrebno za dodavanje, uklanjanje i pretraga je relativno konstantno bez obzira na veličinu kolekcije.

Elementima dictionary-a se pristupa preko vrednosti ključa. Ukoliko nismo sigurni da li se ključ po kom pristupamo elementu dictionary-a zaista nalazi u dictionary-u, potrebno izvršiti odgovarajuću proveru pre pristupanja. Ta provera se najčešće izvršava koristeći metodu *ContainsKey*. Međutim, ukoliko postoji potreba za čestim pristupanjem elementima sa verovatnoćom da ključ ne postoji u dictionary-u, efikasnije je koristiti metodu *TryGetValue*. Ova metoda u pozadini proverava da li navedeni ključ postoji u kolekciji, a zatim vraća vrednosti elementa koji je dodeljen tom ključu. U slučaju da ključ ne postoji, biće vraćena default vrednost koja odgovara tipu elementa. Dakle, za pristupanje elementima preko ključa treba koristiti metodu *TryGetValue* kao mnogo pouzdaniji i efikasniji način pristupanja.

2.2 Nizovi

U rešavanju raznih problema javlja se potreba za postojanjem većeg broja podataka istog tipa koje predstavljaju jednu celinu. Zbog toga se u programskim jezicima uvodi pojam niza ili u opštem slučaju pojam polja. Ovo možemo predstaviti slikom.



Elementi niza su numerisani sa 0,1,2,...,n-1.

Ovi brojevi se nazivaju indeksima elemenata niza. Broj elemenata u nizu predstavlja njegovu dužinu. Nizovi mogu biti različitih dimenzija. Najčešće se koriste jednodimenzionalni nizovi ili vektori i dvodimenzionalni nizovi ili matrice.

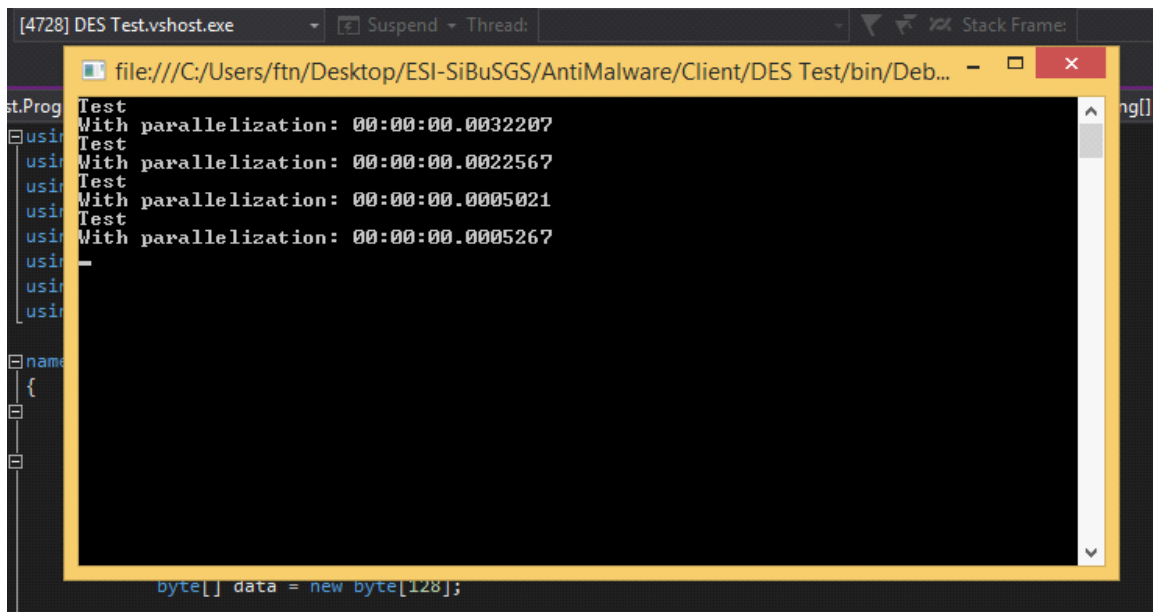
Niz predstavlja složeni tip podataka, sačinjen od nekolicine drugih podataka istog ili različitog tipa. Svaki podatak u nizu se naziva njegovim elementom, a svaki element ima svoj indeks, preko kojeg pristupamo tom elementu u nizu.

Nizovi u okviru našeg sistema se koriste u dosta slučajeva. Uglavnom se koriste za podatke, pre svega nizove bajtova podataka koji se enkriptuju i dekriptuju, kao i ključeve i inicijalni vektor koji su predstavljeni kao nizovi bajtova. Takođe nizovi se dosta koriste za implementaciju 3DES algoritma.

3. TESTIRANJE

Vršeno je više testiranja koja su opisana u daljem tekstu. Testovi pokrivaju sve pozitivne i negativne testne slučajeve (npr. uspešna i neuspešna autentifikacija).

Testirana je mogućnost ubrzanja programa pomoću paralelizacije. Paralelizacija je odrađena na dekripciji kod 3DES algoritma. Slika 3.1 prikazuje kojom brzinom se izvršava 3DES enkripcija i dekripcija sa paralelizovanom dekripcijom. Enkripcija i dekripcija su pozivani više puta. Na slici 3.2 nalazi se prikaz enkripcije i dekripcije bez paralelizacije. Na osnovu testova, zaključili smo da je algoritam bez paralelizacije znatno brži pri inicijalnom pokretanju, a posle su rezultati slični.

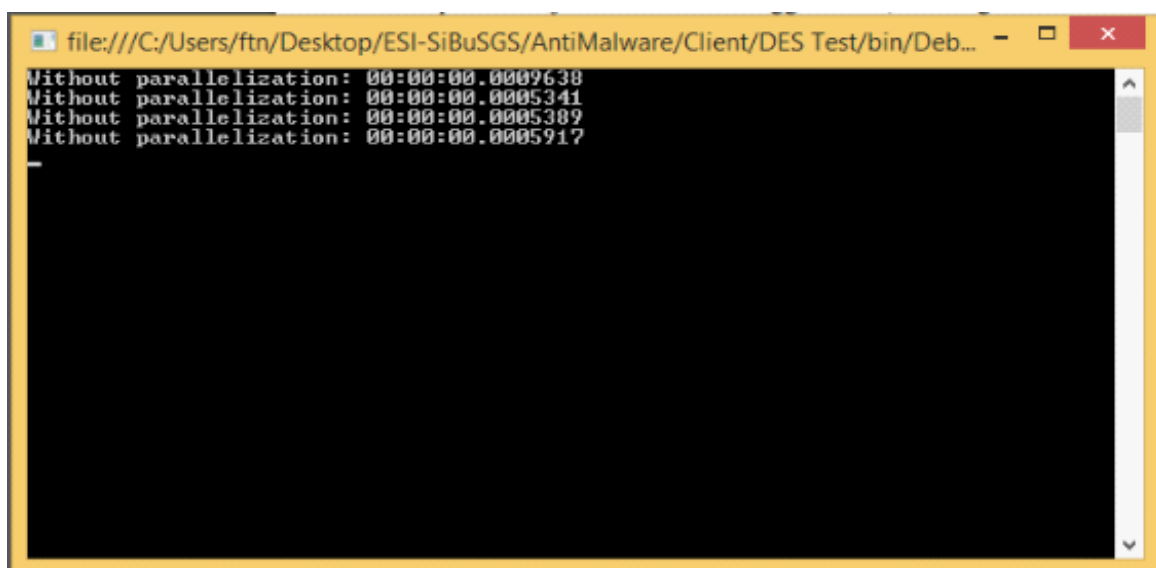


The screenshot shows a debugger window for 'DES Test.vshost.exe'. A console window is open, displaying the following output:

```
Test  
With parallelization: 00:00:00.0032207  
Test  
With parallelization: 00:00:00.0022567  
Test  
With parallelization: 00:00:00.0005021  
Test  
With parallelization: 00:00:00.0005267  
Test  
Test  
Test  
Test
```

The background code in the debugger shows a loop with 'using' statements and a variable declaration: `byte[] data = new byte[128];`

Slika 3.1 Prikaz rada programa sa paralelizacijom

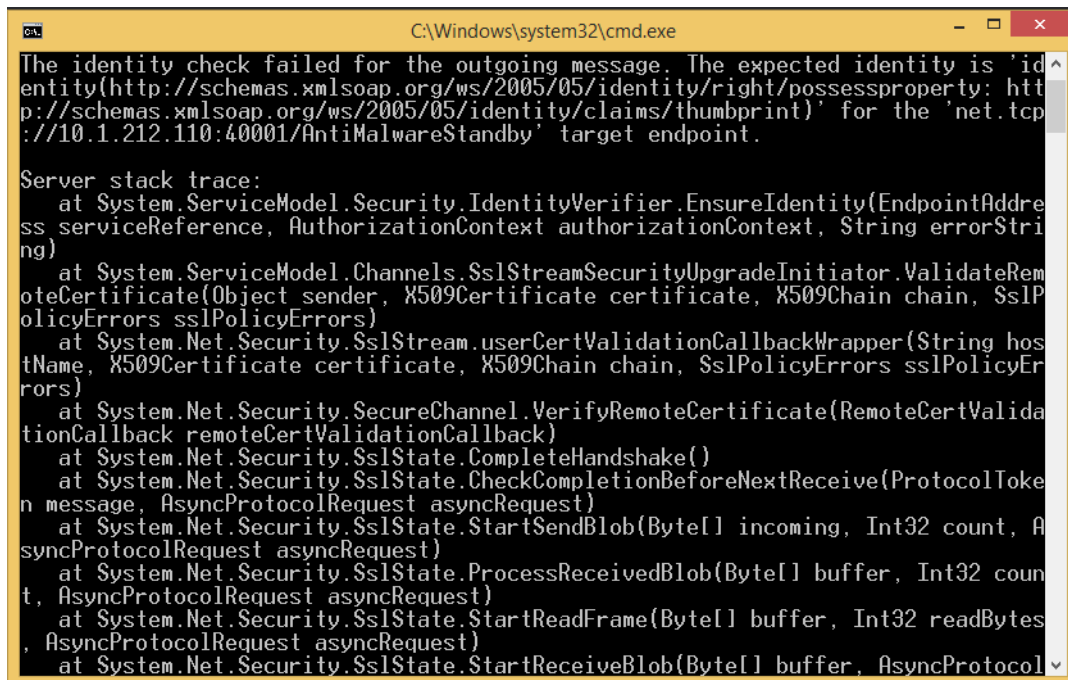


The screenshot shows a debugger window for 'DES Test.vshost.exe'. A console window is open, displaying the following output:

```
Without parallelization: 00:00:00.0009638  
Without parallelization: 00:00:00.0005341  
Without parallelization: 00:00:00.0005309  
Without parallelization: 00:00:00.0005917
```

Slika 3.2 Prikaz rada programa bez paralelizacije

Ponašanje programa kada klijent poseduje nevalidan sertifikat i pokušava da koristi usluge sistema je prikazano na slici 3.3. Kao što se na slici može videti klijent ne uspeva da se poveže sa servisom, jer nema validan sertifikat za autentifikaciju.



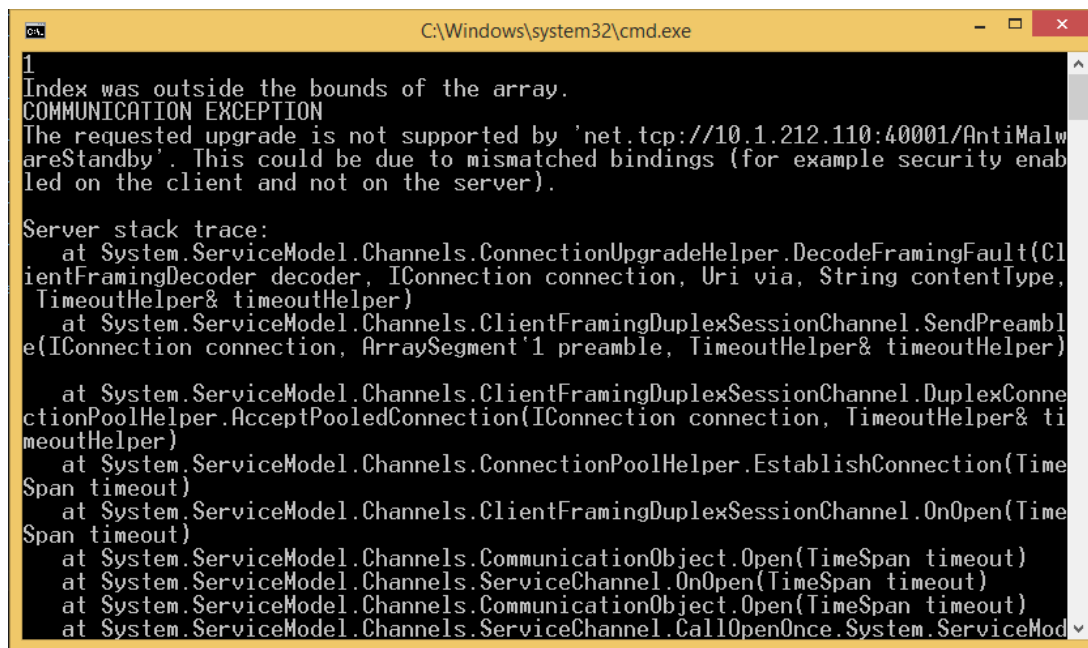
```
C:\Windows\system32\cmd.exe

The identity check failed for the outgoing message. The expected identity is 'identity(http://schemas.xmlsoap.org/ws/2005/05/identity/right/possessproperty: http://schemas.xmlsoap.org/ws/2005/05/identity/claims/thumbprint)' for the 'net.tcp://10.1.212.110:40001/AntiMalwareStandby' target endpoint.

Server stack trace:
   at System.ServiceModel.Security.IdentityVerifier.EnsureIdentity(EndpointAddress serviceReference, AuthorizationContext authorizationContext, String errorString)
   at System.ServiceModel.Channels.SslStreamSecurityUpgradeInitiator.ValidateRemoteCertificate(Object sender, X509Certificate certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors)
   at System.Net.Security.SslStream.UserCertValidationCallbackWrapper(String hostName, X509Certificate certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors)
   at System.Net.Security.SecureChannel.VerifyRemoteCertificate(RemoteCertificateValidationCallback remoteCertValidationCallback)
   at System.Net.Security.SslState.CompleteHandshake()
   at System.Net.Security.SslState.CheckCompletionBeforeNextReceive(ProtocolToken message, AsyncProtocolRequest asyncRequest)
   at System.Net.Security.SslState.StartSendBlob(Byte[] incoming, Int32 count, AsyncProtocolRequest asyncRequest)
   at System.Net.Security.SslState.ProcessReceivedBlob(Byte[] buffer, Int32 count, AsyncProtocolRequest asyncRequest)
   at System.Net.Security.SslState.StartReadFrame(Byte[] buffer, Int32 readBytes, AsyncProtocolRequest asyncRequest)
   at System.Net.Security.SslState.StartReceiveBlob(Byte[] buffer, AsyncProtocolRequest asyncRequest)
```

Slika 3.3 Ponašanje programa sa nevalidnim klijentskim sertifikatom

Na slici 3.4 nalazi se prikaz ponašanja programa ako klijent nema sertifikat, a pokušava da komunicira sa serverom. Kao što se može videti na slici u ovoj situaciji dolazi do communication exception-a, jer klijent ne poseduje nijedan sertifikat za autentifikaciju, a definisana je autentifikacija sertifikatima.



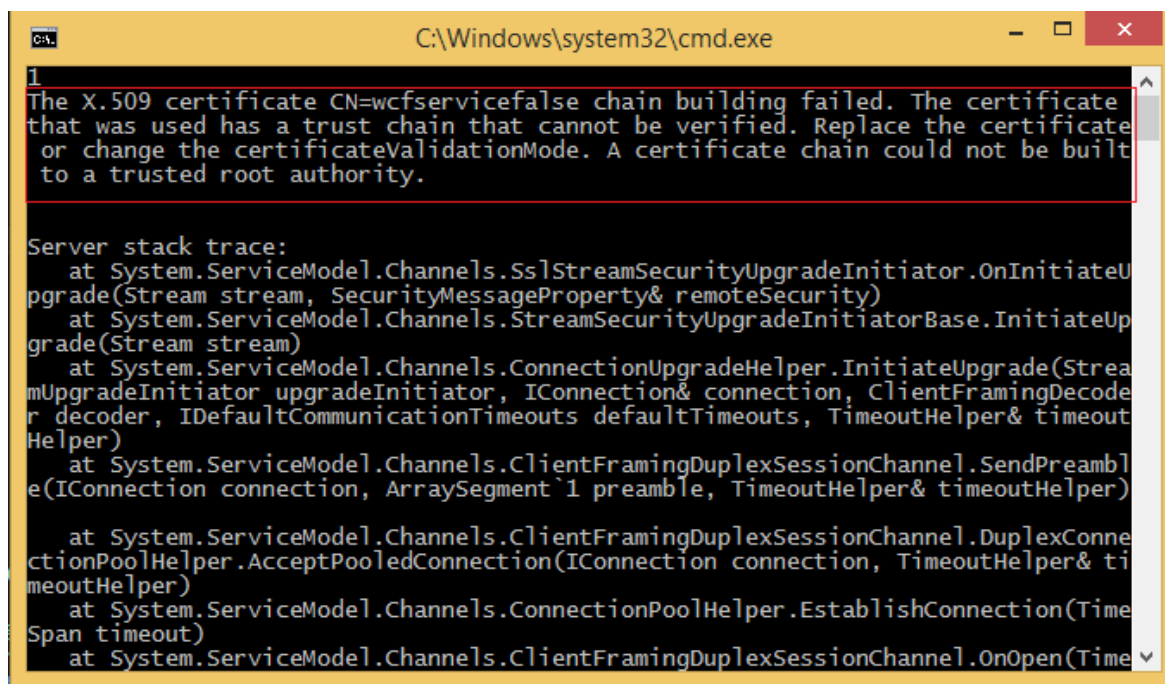
```
C:\Windows\system32\cmd.exe

1
Index was outside the bounds of the array.
COMMUNICATION EXCEPTION
The requested upgrade is not supported by 'net.tcp://10.1.212.110:40001/AntiMalwareStandby'. This could be due to mismatched bindings (for example security enabled on the client and not on the server).

Server stack trace:
   at System.ServiceModel.Channels.ConnectionUpgradeHelper.DecodeFramingFault(ClientFramingDecoder decoder, IConnection connection, Uri via, String contentType, TimeoutHelper& timeoutHelper)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.SendPreamble(IConnection connection, ArraySegment`1 preamble, TimeoutHelper& timeoutHelper)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.DuplexConnectionPoolHelper.AcceptPooledConnection(IConnection connection, TimeoutHelper& timeoutHelper)
   at System.ServiceModel.Channels.ConnectionPoolHelper.EstablishConnection(TimeSpan timeout)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.OnOpen(TimeSpan timeout)
   at System.ServiceModel.Channels.CommunicationObject.Open(TimeSpan timeout)
   at System.ServiceModel.Channels.ServiceChannel.OnOpen(TimeSpan timeout)
   at System.ServiceModel.Channels.CommunicationObject.Open(TimeSpan timeout)
   at System.ServiceModel.Channels.ServiceChannel.CallOpenOnce.System.ServiceModel.Channels.ICommunicationObject.Open(TimeSpan timeout)
```

Slika 3.4 Ponašanje programa kada klijent nema sertifikat

Ponašanje programa kada server ima nevalidan sertifikat prikazano je na slici 3.5. U okviru poruke jasno se zaključuje da serverski sertifikat nije validan, tj. da sertifikat koristi trust chain koji se ne može proveriti, tako da dalje izvršavanje programa nije moguće bez validnog sertifikata.



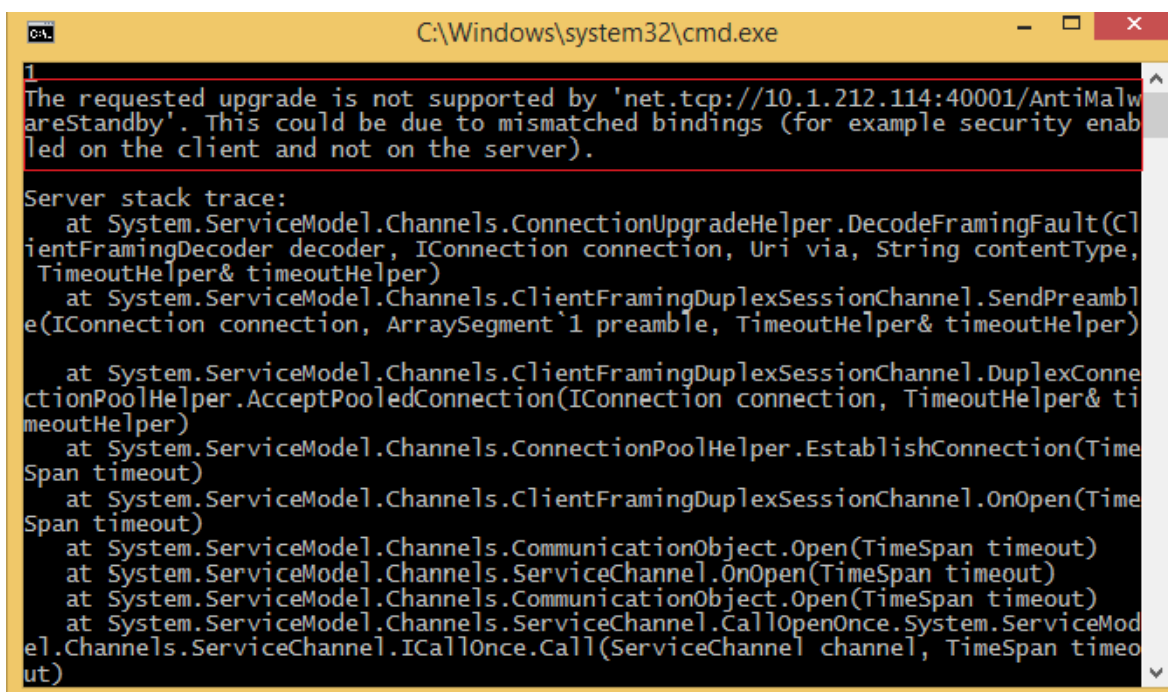
```
C:\Windows\system32\cmd.exe

1
The X.509 certificate CN=wcfservicefalse chain building failed. The certificate
that was used has a trust chain that cannot be verified. Replace the certificate
or change the certificateValidationMode. A certificate chain could not be built
to a trusted root authority.

Server stack trace:
   at System.ServiceModel.Channels.SslStreamSecurityUpgradeInitiator.OnInitiateUp
pgrade(Stream stream, SecurityMessageProperty& remoteSecurity)
   at System.ServiceModel.Channels.StreamSecurityUpgradeInitiatorBase.InitiateUp
grade(Stream stream)
   at System.ServiceModel.Channels.ConnectionUpgradeHelper.InitiateUpgrade(Strea
mUpgradeInitiator upgradeInitiator, IConnection& connection, ClientFramingDecode
r decoder, IDefaultCommunicationTimeouts defaultTimeouts, TimeoutHelper& timeout
Helper)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.SendPreambl
e(IConnection connection, ArraySegment`1 preamble, TimeoutHelper& timeoutHelper)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.DuplexConne
ctionPoolHelper.AcceptPooledConnection(IConnection connection, TimeoutHelper& ti
meoutHelper)
   at System.ServiceModel.Channels.ConnectionPoolHelper.EstablishConnection(Time
Span timeout)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.OnOpen(Time
```

Slika 3.5 Ponašanje programa kada server ima nevalidan sertifikat

Na slici 3.6 prikazano je kako se program ponaša kada server ne poseduje sertifikat. Dalje izvršavanje programa nije moguće usled nedostatka validnog serverskog sertifikata.



```
C:\Windows\system32\cmd.exe

1
The requested upgrade is not supported by 'net.tcp://10.1.212.114:40001/AntiMalw
areStandby'. This could be due to mismatched bindings (for example security enab
led on the client and not on the server).

Server stack trace:
   at System.ServiceModel.Channels.ConnectionUpgradeHelper.DecodeFramingFault(Cl
ientFramingDecoder decoder, IConnection connection, Uri via, String contentType,
TimeoutHelper& timeoutHelper)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.SendPreambl
e(IConnection connection, ArraySegment`1 preamble, TimeoutHelper& timeoutHelper)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.DuplexConne
ctionPoolHelper.AcceptPooledConnection(IConnection connection, TimeoutHelper& ti
meoutHelper)
   at System.ServiceModel.Channels.ConnectionPoolHelper.EstablishConnection(Time
Span timeout)
   at System.ServiceModel.Channels.ClientFramingDuplexSessionChannel.OnOpen(Time
Span timeout)
   at System.ServiceModel.Channels.CommunicationObject.Open(TimeSpan timeout)
   at System.ServiceModel.Channels.ServiceChannel.OnOpen(TimeSpan timeout)
   at System.ServiceModel.Channels.CommunicationObject.Open(TimeSpan timeout)
   at System.ServiceModel.Channels.ServiceChannel.CallOpenOnce.System.ServiceMod
el.Channels.ServiceChannel.ICallOnce.Call(ServiceChannel channel, TimeSpan timeo
ut)
```

Slika 3.6 Ponašanje programa kada server ne poseduje sertifikat

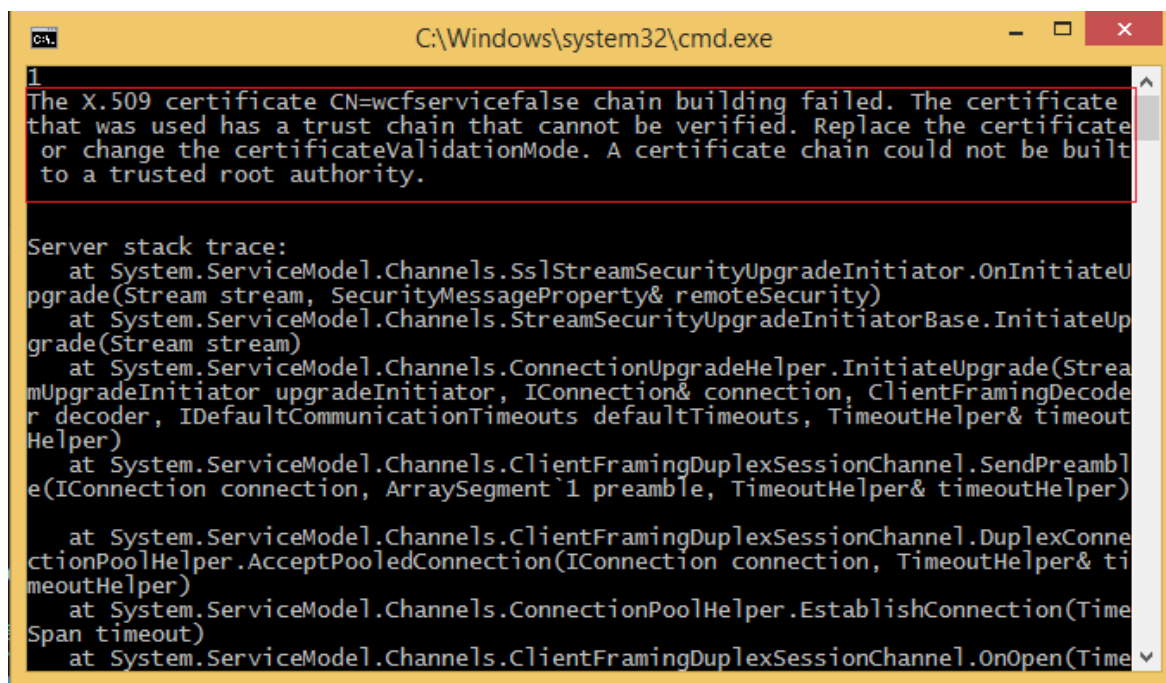
Za kriptovanje poruka korišćen je 3DES algoritam. Na slici 3.7 se može videti da su rezultati kriptovanja isti primenom našeg 3DES algoritma i 3DES algoritma korišćenjem *TripleDESCryptoServiceProvider* klase.

```
Homemade triple DES encrypted text:
6HtaITvLUeh4SWCq4Xzg+cXjcbvxUqWIua9AcrUAXr8zYr32wUJU08s97CRzkgoYf2ZK3TIUaESMqrFU2YMGRg==

C# triple DES encrypted text:
6HtaITvLUeh4SWCq4Xzg+cXjcbvxUqWIua9AcrUAXr8zYr32wUJU08s97CRzkgoYf2ZK3TIUaESMqrFU2YMGRg==
Press any key to continue . . .
```

Slika 3.7 Rezultat testa 3des algoritma

Sledeći test je primer simulacije *Man in the middle* napada. Server se podigne sa lažnim (nevalidnim) sertifikatom, i ideja je kada klijent njemu pošalje poruku, on tu poruku ukrade, izmeni i prosledi dalje. Pošto taj server koji je podignut uopšte nema validan sertifikat, uopšte taj naš klijent neće moći da stupi u komunikaciju sa njim. Rezultat ovog testa prikazan je na slici 3.8.



Slika 3.8 Rezultat testa MITM napada

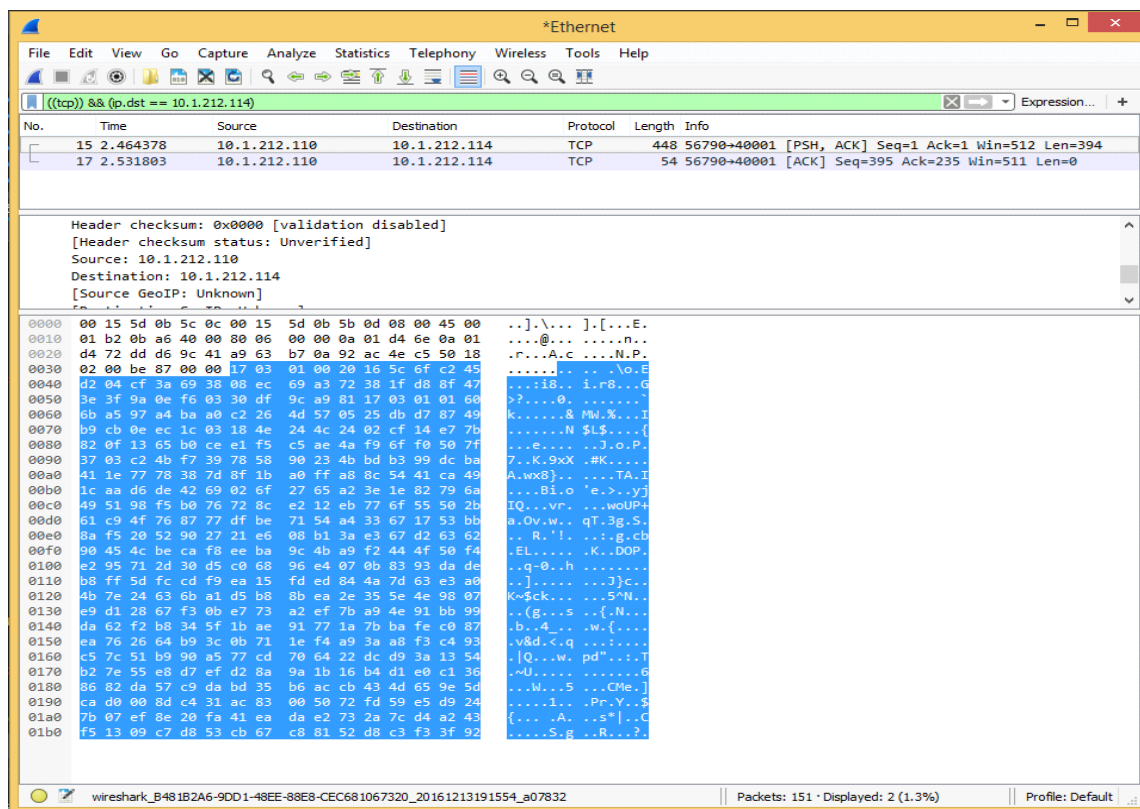
3.1 Testiranje ispravnosti sigurnosnih protokola kroz alat Wireshark

U okviru ovog poglavlja biće izvršeno testiranje ispravnosti sigurnosnih protokola putem alata *Wireshark*.

Wireshark je besplatan paket analizator, otvorenog koda. Koristi se za mrežno rešavanje problema, analizu, razvoj softvera, i komunikacionog protokola. Ovo je softver koji razume različite mrežne protokole. Može analizirati i prikazivati polja, zajedno sa njihovim značenjima kao što je navedeno od strane različitih mrežnih protokola.

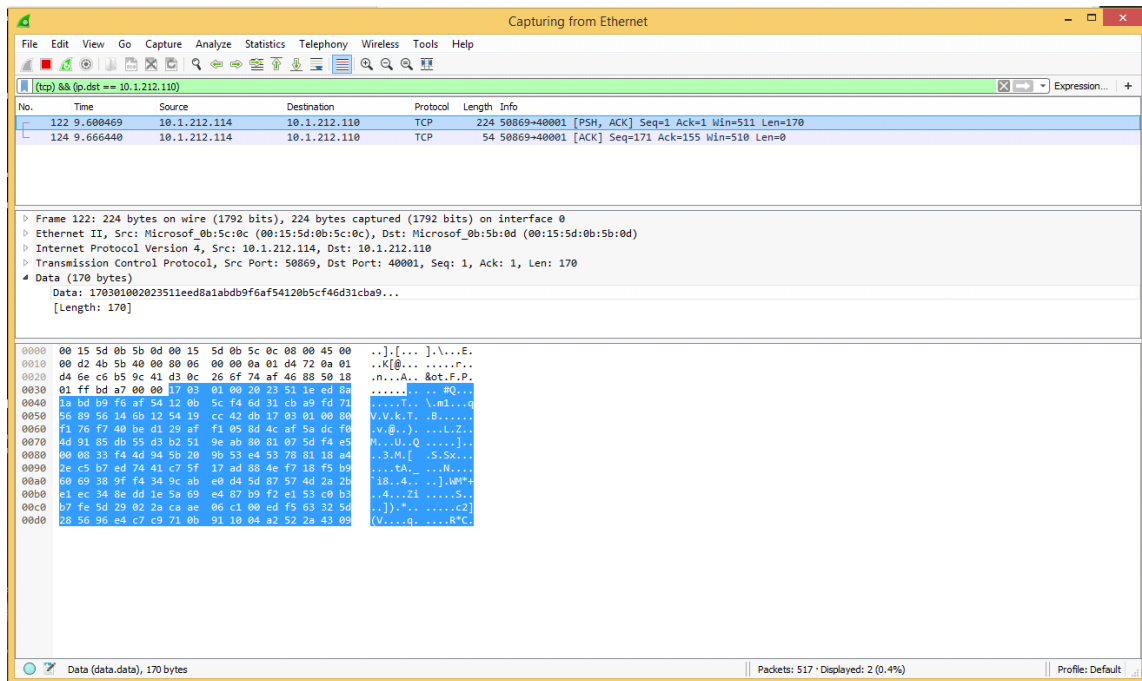
Da bi ovaj test bio uspešan potrebno je videti da li su podaci prilikom razmena poruka enkriptovani. Kao što se na narednim slikama može videti podaci prilikom razmena poruka su enkriptovani, tj. ne šalju se u čistom obliku (*plain text-u*). Dalje je potrebno pokazati da enkriptovane poruke klijentovog odgovora ka servisu, kao i poruke klijenta ka servisu koje nisu enkriptovane, nisu iste. To se može videti na slikama 3.7 i 3.8 ukoliko se uporede odgovori.

Slika 3.9 prikazuje klijentov zahtev ka servisu koji nije kriptovan.



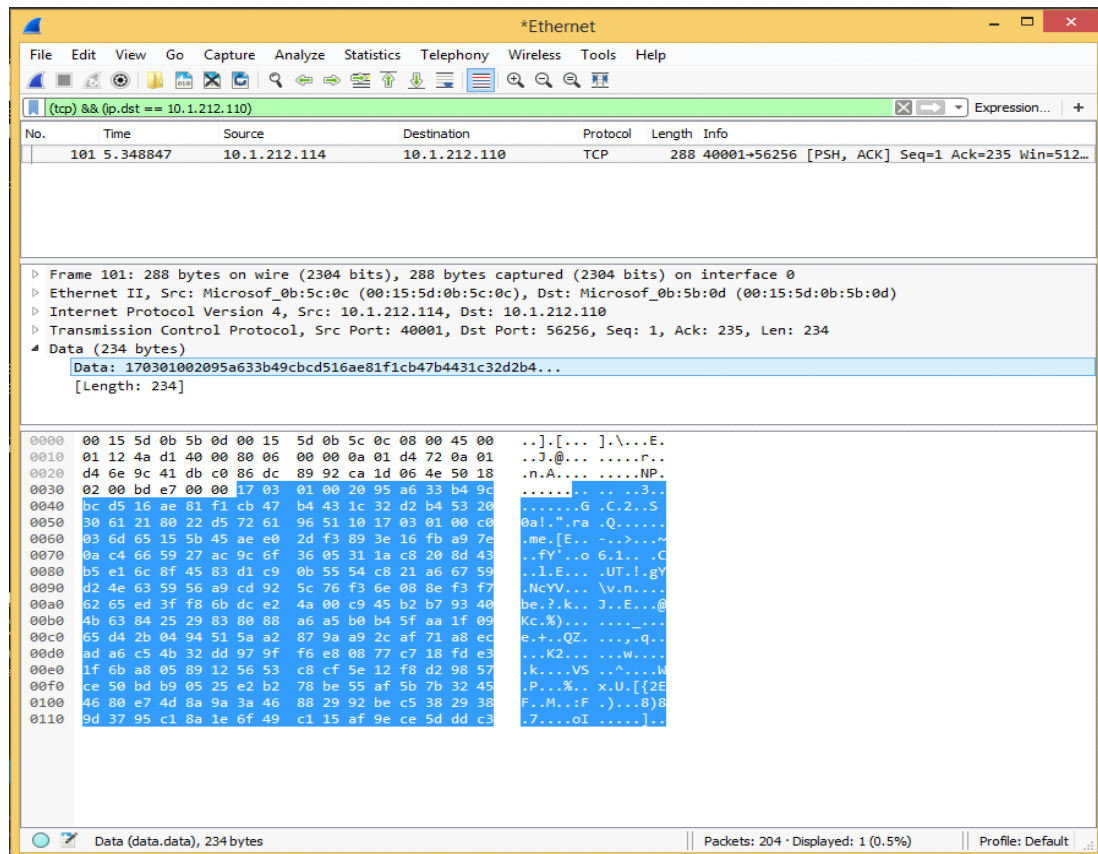
Slika 3.9 Prikaz klijentovog odgovora ka servisu koji nije kriptovan

Na slici 3.10 nalazi se prikaz kriptovanog klijentskog odgovora ka servisu.



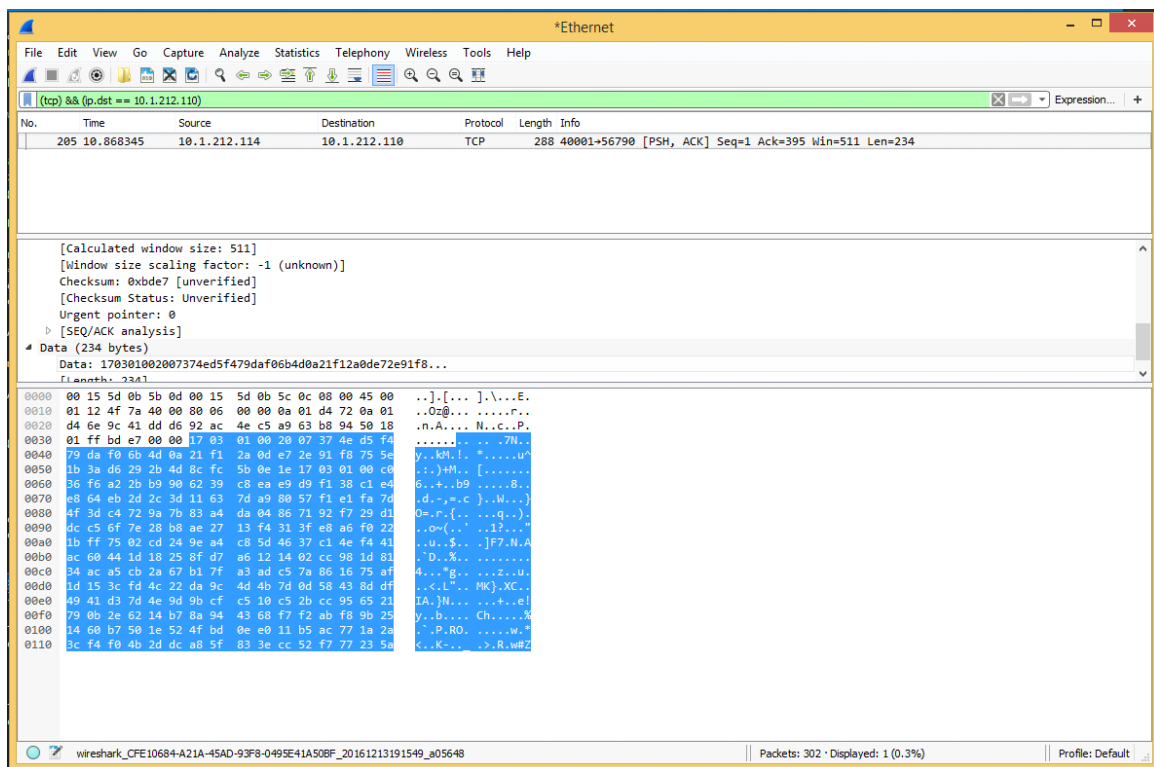
Slika 3.10 Prikaz kriptovanog klijentskog odgovora ka servisu

Na slici 3.11 nalazi se prikaz kriptovanog odgovora servisa ka klijentu.



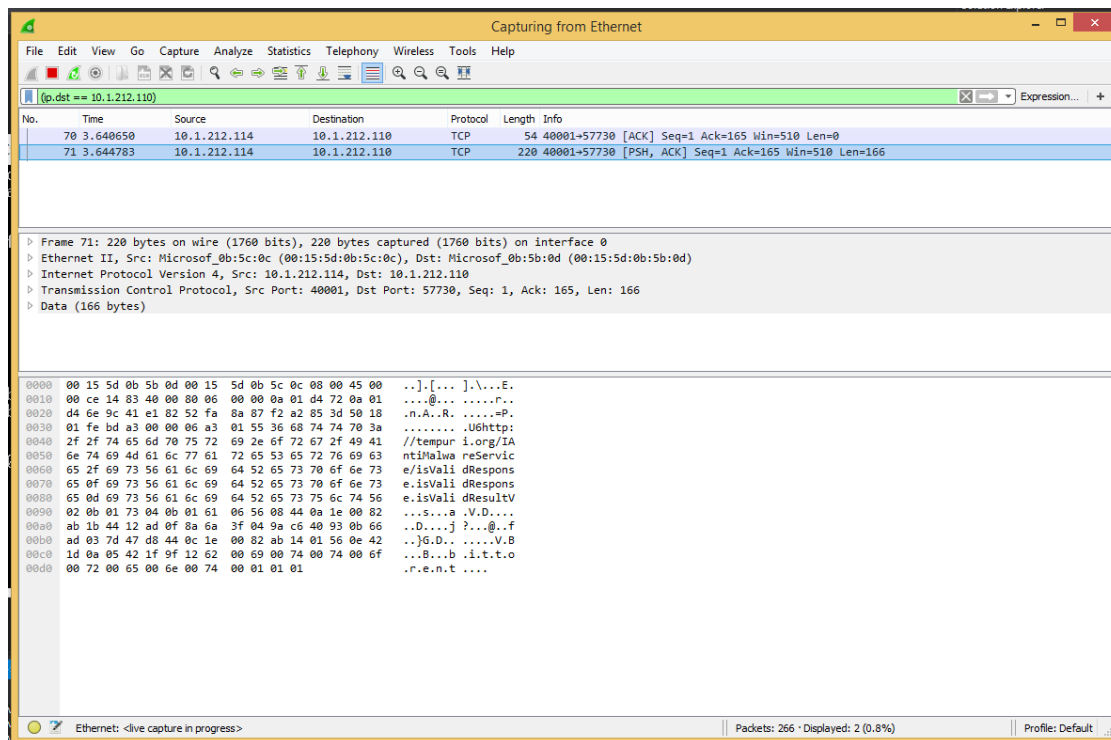
Slika 3.11 Prikaz kriptovanog odgovora servisa ka klijentu

Na slici 3.12 nalazi se odgovor servisa klijentu bez TripleDES šifrovanja.



Slika 3.12 Odgovor servisa klijentu bez TripleDES šifrovanja

Na slici 3.13 se nalazi ne kriptovana poruka klijenta prikayana u alatu wireshark. Kao što se na slici može videti odgovor nije kriptovan i može se pročitati (*plain text*).

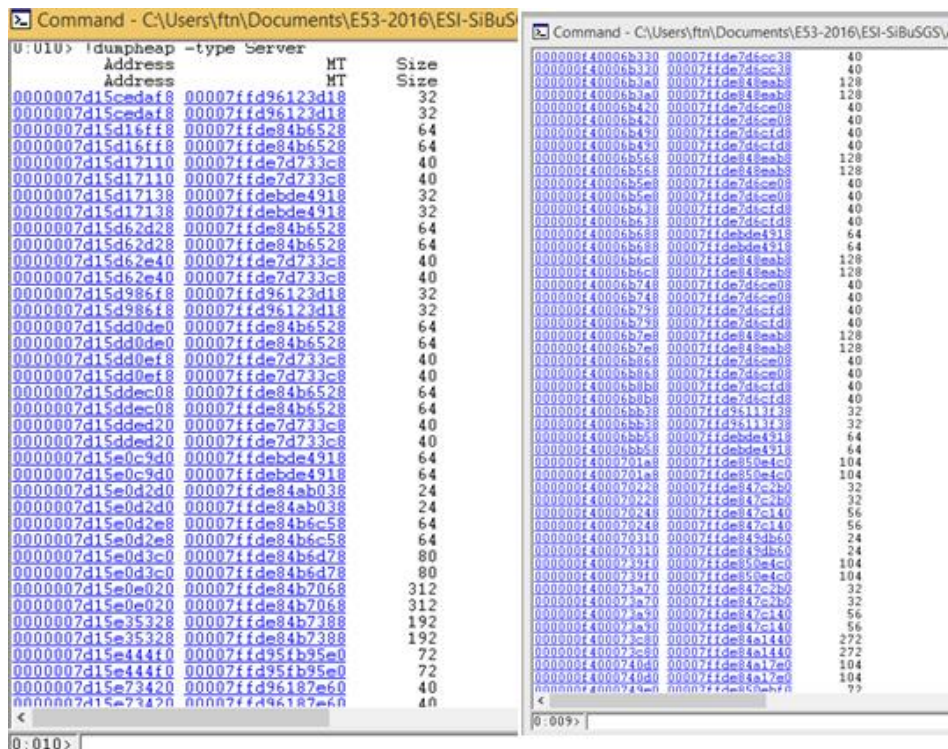


Slika 3.13 Prikaz ne kriptovane poruke

3.2 Testiranje curenja memorije putem alata WinDbg

Debugging je tehnika za otkrivanje problema i grešaka (bugova) u programskom kodu koje se mogu ispoljavati stalno ili povremeno. Stavljanje breakpoint-a u izvornom kodu koristeći Visual Studio je jedan od načina za debugging, ali to nije uvek moguće, jer izvorni kod ne mora uvek biti dostupan. U tom slučaju, potrebno je koristiti debugging alate kako bi se analizirali različiti problemi kao što je pronalaženje uzroka rušenja aplikacije, problemi sa performansama (performance issues), curenje memorije (memory leak), i slično. Glavna prednost ovog alata je GUI komponenta koja znatno olakšava proces debugovanja.

Na slici 3.14 prikazan je dump heap kada se klijent konektuje na server, prvi deo



The image shows two command windows from WinDbg. The left window displays the output of the `!dumpheap -type Server` command, showing a list of memory blocks with their addresses, MT (Memory Tag), and sizes. The right window shows a continuation of the same data, displaying the hex values of the memory blocks.

Address	MT	Size
0000007d15cedaf8	00007fde96123d18	32
0000007d15cedaf8	00007fde96123d18	32
0000007d15d16ff8	00007fde84b6528	64
0000007d15d16ff8	00007fde84b6528	64
0000007d15d17110	00007fde7d733c8	40
0000007d15d17110	00007fde7d733c8	40
0000007d15d17138	00007fdebde4918	32
0000007d15d17138	00007fdebde4918	32
0000007d15d62d28	00007fde84b6528	64
0000007d15d62d28	00007fde84b6528	64
0000007d15d62e40	00007fde7d733c8	40
0000007d15d62e40	00007fde7d733c8	40
0000007d15d986f8	00007fde96123d18	32
0000007d15d986f8	00007fde96123d18	32
0000007d15dd0d00	00007fde84b6528	64
0000007d15dd0d00	00007fde84b6528	64
0000007d15dd0e18	00007fde7d733c8	40
0000007d15dd0e18	00007fde7d733c8	40
0000007d15dd0e18	00007fde7d733c8	40
0000007d15dd0e08	00007fde84b6528	64
0000007d15dd0e08	00007fde84b6528	64
0000007d15dd0e20	00007fde7d733c8	40
0000007d15dd0e20	00007fde7d733c8	40
0000007d15e0c9d0	00007fdebde4918	64
0000007d15e0c9d0	00007fdebde4918	64
0000007d15e0d2d0	00007fde84b6038	24
0000007d15e0d2d0	00007fde84b6038	24
0000007d15e0d2e8	00007fde84b6c58	64
0000007d15e0d2e8	00007fde84b6c58	64
0000007d15e0d3c0	00007fde84b6d78	80
0000007d15e0d3c0	00007fde84b6d78	80
0000007d15e0e020	00007fde84b7068	312
0000007d15e0e020	00007fde84b7068	312
0000007d15e35328	00007fde84b7388	192
0000007d15e35328	00007fde84b7388	192
0000007d15e444f0	00007fde95fb95e0	72
0000007d15e444f0	00007fde95fb95e0	72
0000007d15e73420	00007fde96187e60	40
0000007d15e73420	00007fde96187e60	40

Slika 3.14 Prikaz dump heap-a kada se klijent konektuje na server

Slika 3.15 prikazuje dump heap kada se klijent konektuje drugi deo

```

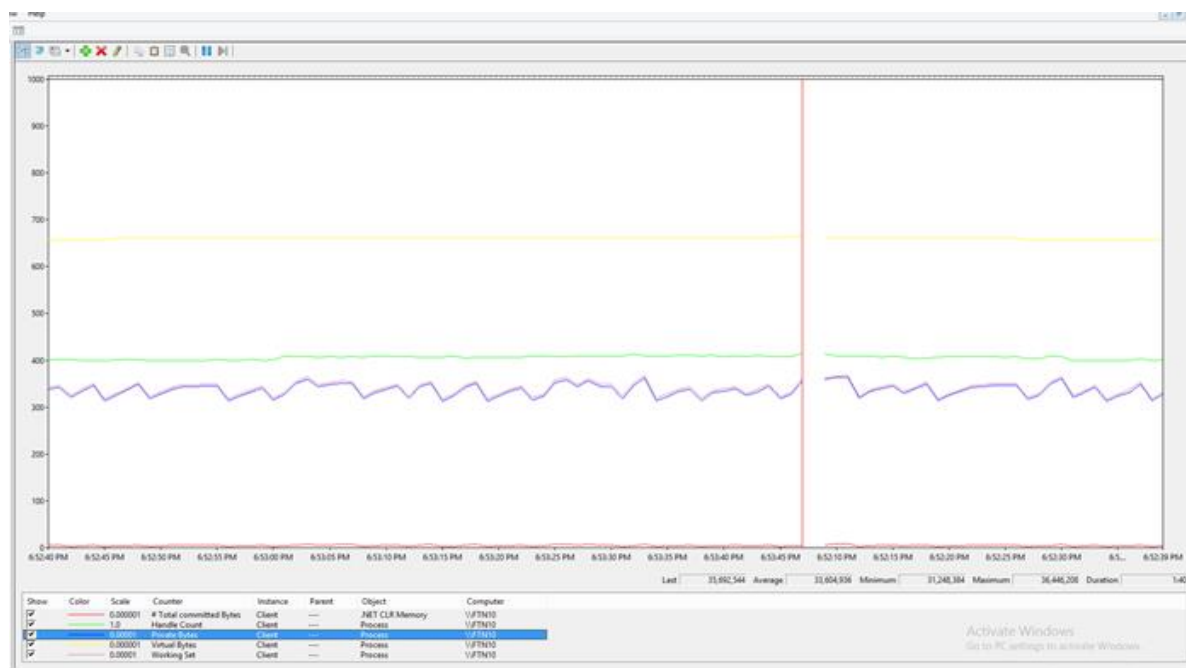
C:\Users\ftn\Documents\E53-2016\ESI-SiBuSGS\AntiMalware\Client\Ser...
0000007d15e73330 00007ffde8b6e728 32
0000007d15e73448 00007ffde8b6e4918 32
0000007d15e9dce8 00007ffdf7dd8d8d0 64
0000007d15e9dce8 00007ffdf7dd8d8d0 64
0000007d15f211a0 00007ffdf96128c18 40
0000007d15f211a0 00007ffdf96128c18 40

Statistics:
Statistics:
MT      Count  TotalSize Class Name
MT      Count  TotalSize Class Name
00007ffde84ab038 1 24 00007ffde84ab038 1 24 System
System.Collections.Generic.ObjectEqualityComparer`1[System.ServiceModel.Channels.In
00007ffdf96187e60 1 40 00007ffdf96187e60 1 40 System
System.Collections.Generic.List`1[MySql.Data.MySqlClient.Replication.ReplicationSer
00007ffdf96128c18 1 40 00007ffdf96128c18 1 40 Server
Server.ServiceData
00007ffdf7dd8d8d0 1 64 00007ffdf7dd8d8d0 1 64 System
System.Management.WmiNetUtilsHelper.ConnectServerWmi
00007ffde84b6c58 1 64 00007ffde84b6c58 1 64 System
System.ServiceModel.Channels.ServerSessionPreambleCallback
00007ffdf96123d18 2 64 00007ffdf96123d18 2 64 Micros
Microsoft.VisualStudio.Diagnostics.ServiceModelSink.StubServerEventSink
00007ffdf95fb95e0 1 72 00007ffdf95fb95e0 1 72 Server
Server.Service
00007ffde84b6d78 1 80 00007ffde84b6d78 1 80 System
System.ServiceModel.Channels.ServerSessionDecoder
00007ffde8b6e4918 3 128 00007ffde8b6e4918 3 128 System
System.Object[]
00007ffde7d733c8 4 160 00007ffde7d733c8 4 160 System
System.Collections.Generic.List`1[System.ServiceModel.Channels.InitialServerConnect
00007ffde84b7388 1 192 00007ffde84b7388 1 192 System
System.ServiceModel.Channels.ServerSessionPreambleConnectionReader+ServerFramingDupl
00007ffde84b6528 4 256 00007ffde84b6528 4 256 System
System.ServiceModel.Channels.ServerSessionPreambleDeauxCallback
00007ffde84b7068 1 312 00007ffde84b7068 1 312 System
System.ServiceModel.Channels.ServerSessionPreambleConnectionReader+ServerFramingDupl
Total 22 objects
Total 22 objects
0:010>

```

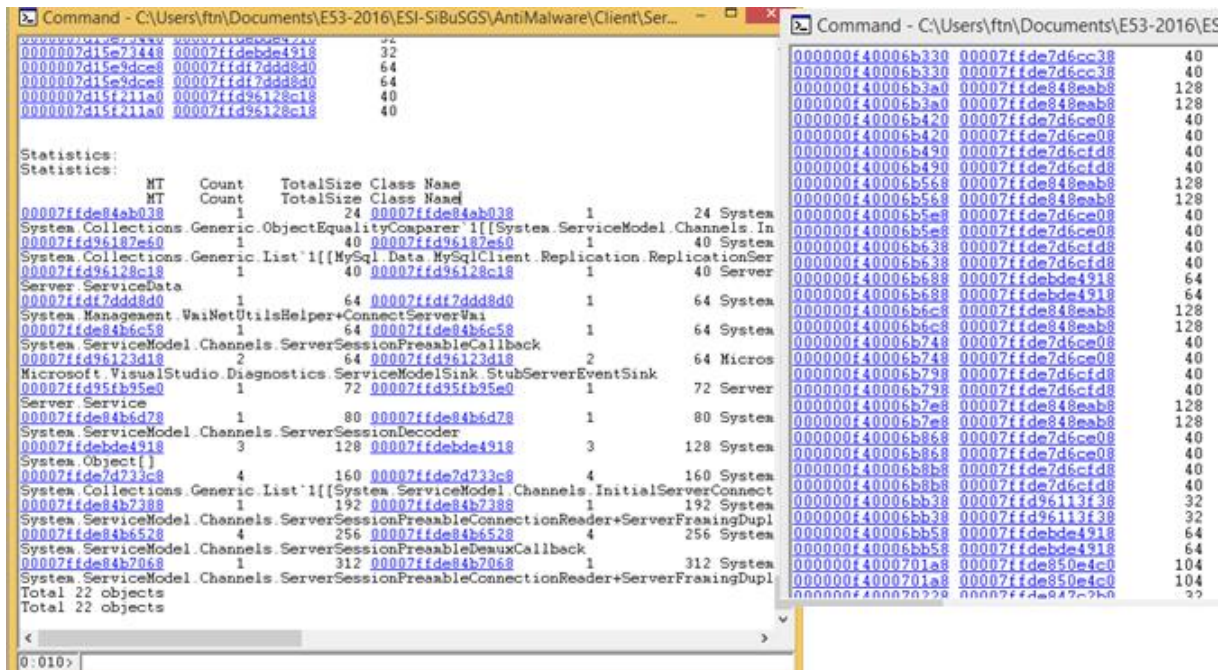
Slika 3.15 Prikaz dump heap-a kada se klijent konektuje

Slika 3.16 prikazuje kako se program ponaša kada je izložen stres testu. Na monitoru su prikazani sledeći podaci: *total committed bytes*, *handle count*, *private bytes*, *virtual bytes* i *working set*.



Slika 3.16 Prikaz ponašanja programa kada je izložen stres testu

Slika 3.17 prikazuje dump heap servera kada obradi nekoliko hiljada klijentskih zahteva



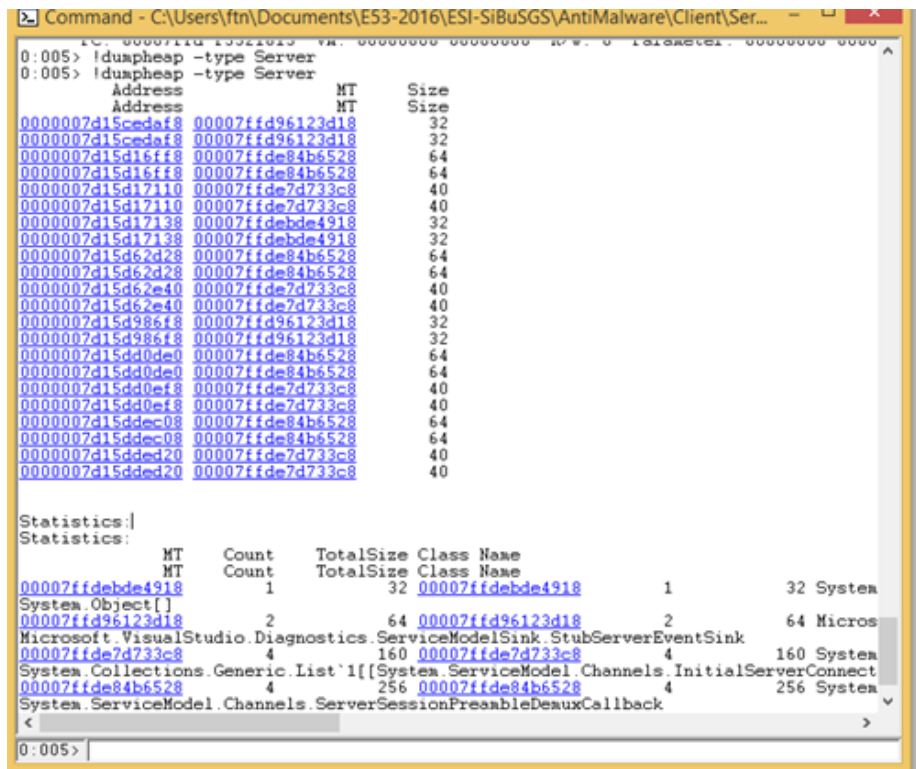
```
Command - C:\Users\ftn\Documents\E53-2016\ESI-SiBuSGS\AntiMalware\Client\Ser...
0000007d15e21448 00007ffde4918 32
0000007d15e2dce8 00007ffdf7dd8d40 64
0000007d15e2dce8 00007ffdf7dd8d40 64
0000007d15f211a0 00007ffdf96128c18 40
0000007d15f211a0 00007ffdf96128c18 40

Statistics:
Statistics:
MT Count TotalSize Class Name
MT Count TotalSize Class Name
00007ffde84ab038 1 24 00007ffde84ab038 1 24 System
System.Collections.Generic.ObjectEqualityComparer`1[System.ServiceModel.Channels.In
00007ffdf96187e60 1 40 00007ffdf96187e60 1 40 System
System.Collections.Generic.List`1[MySql.Data.MySqlClient.Replication.ReplicationSer
00007ffdf96128c18 1 40 00007ffdf96128c18 1 40 Server
Server.ServiceData
00007ffdf7ddd8d0 1 64 00007ffdf7ddd8d0 1 64 System
System.Management.VaiNetUtilsHelper.ConnectServerVai
00007ffde84b6c58 1 64 00007ffde84b6c58 1 64 System
System.ServiceModel.Channels.ServerSessionPreambleCallback
00007ffdf96123d18 2 64 00007ffdf96123d18 2 64 Micros
Microsoft.VisualStudio.Diagnostics.ServiceModelSink.StubServerEventSink
00007ffdf961b95e0 1 72 00007ffdf961b95e0 1 72 Server
Server.Service
00007ffde84b6d78 1 80 00007ffde84b6d78 1 80 System
System.ServiceModel.Channels.ServerSessionDecoder
00007ffde4918 3 128 00007ffde4918 3 128 System
System.Object[]
00007ffde7d233c8 4 160 00007ffde7d233c8 4 160 System
System.Collections.Generic.List`1[System.ServiceModel.Channels.InitialServerConnect
00007ffde84b7388 1 192 00007ffde84b7388 1 192 System
System.ServiceModel.Channels.ServerSessionPreambleConnectionReader+ServerFrainingDupl
00007ffde84b6528 4 256 00007ffde84b6528 4 256 System
System.ServiceModel.Channels.ServerSessionPreambleDeauxCallback
00007ffde84b7068 1 312 00007ffde84b7068 1 312 System
System.ServiceModel.Channels.ServerSessionPreambleConnectionReader+ServerFrainingDupl
Total 22 objects
Total 22 objects

Command - C:\Users\ftn\Documents\E53-2016\ESI-SiBuSGS\AntiMalware\Client\Ser...
000000f40006b330 00007ffde7d6cc38 40
000000f40006b330 00007ffde7d6cc38 40
000000f40006b3a0 00007ffde848eab8 128
000000f40006b3a0 00007ffde848eab8 128
000000f40006b420 00007ffde7d6cc08 40
000000f40006b420 00007ffde7d6cc08 40
000000f40006b490 00007ffde7d6cc08 40
000000f40006b490 00007ffde7d6cc08 40
000000f40006b568 00007ffde848eab8 128
000000f40006b568 00007ffde848eab8 128
000000f40006b5e8 00007ffde7d6cc08 40
000000f40006b5e8 00007ffde7d6cc08 40
000000f40006b638 00007ffde7d6cc08 40
000000f40006b638 00007ffde7d6cc08 40
000000f40006b688 00007ffde4918 64
000000f40006b688 00007ffde4918 64
000000f40006b6c8 00007ffde848eab8 128
000000f40006b6c8 00007ffde848eab8 128
000000f40006b748 00007ffde7d6cc08 40
000000f40006b748 00007ffde7d6cc08 40
000000f40006b798 00007ffde7d6cc08 40
000000f40006b798 00007ffde7d6cc08 40
000000f40006b7e8 00007ffde848eab8 128
000000f40006b7e8 00007ffde848eab8 128
000000f40006b868 00007ffde7d6cc08 40
000000f40006b868 00007ffde7d6cc08 40
000000f40006b8b8 00007ffde7d6cc08 40
000000f40006b8b8 00007ffde7d6cc08 40
000000f40006bb38 00007ffdf96113f38 32
000000f40006bb38 00007ffdf96113f38 32
000000f40006bb58 00007ffde4918 64
000000f40006bb58 00007ffde4918 64
000000f40006bbf8 00007ffde848eab8 128
000000f40006bbf8 00007ffde848eab8 128
000000f4000701a8 00007ffde850e4c0 104
000000f4000701a8 00007ffde850e4c0 104
000000f400070720 00007ffde848eab8 128
```

Slika 3.17 Prikaz dump heap-a servera kada obradi nekoliko hiljada klijentskih zahteva

Slika 3.18 prikazuje dump heapa kada se server startuje



```
Command - C:\Users\ftn\Documents\E53-2016\ESI-SiBuSGS\AntiMalware\Client\Ser...
0:005> !dumpheap -type Server
0:005> !dumpheap -type Server
Address MT Size
Address MT Size
0000007d15cedaf8 00007ffdf96123d18 32
0000007d15cedaf8 00007ffdf96123d18 32
0000007d15d16ff8 00007ffde84b6528 64
0000007d15d16ff8 00007ffde84b6528 64
0000007d15d17110 00007ffde7d733c8 40
0000007d15d17110 00007ffde7d733c8 40
0000007d15d17138 00007ffde4918 32
0000007d15d17138 00007ffde4918 32
0000007d15d62d28 00007ffde84b6528 64
0000007d15d62d28 00007ffde84b6528 64
0000007d15d62e40 00007ffde7d733c8 40
0000007d15d62e40 00007ffde7d733c8 40
0000007d15d986f8 00007ffdf96123d18 32
0000007d15d986f8 00007ffdf96123d18 32
0000007d15dd0de0 00007ffde84b6528 64
0000007d15dd0de0 00007ffde84b6528 64
0000007d15dd0ef8 00007ffde7d733c8 40
0000007d15dd0ef8 00007ffde7d733c8 40
0000007d15dd0e08 00007ffde84b6528 64
0000007d15dd0e08 00007ffde84b6528 64
0000007d15dd0e20 00007ffde7d733c8 40
0000007d15dd0e20 00007ffde7d733c8 40

Statistics:
Statistics:
MT Count TotalSize Class Name
MT Count TotalSize Class Name
00007ffde4918 1 32 00007ffde4918 1 32 System
System.Object[]
00007ffdf96123d18 2 64 00007ffdf96123d18 2 64 Micros
Microsoft.VisualStudio.Diagnostics.ServiceModelSink.StubServerEventSink
00007ffde7d733c8 4 160 00007ffde7d733c8 4 160 System
System.Collections.Generic.List`1[System.ServiceModel.Channels.InitialServerConnect
00007ffde84b6528 4 256 00007ffde84b6528 4 256 System
System.ServiceModel.Channels.ServerSessionPreambleDeauxCallback
```

Slika 3.18 Prikaz dump heapa kada se server startuje