IJCCS, Vol.9, No.2, July 2015, pp. 177~186
ISSN: 1978-1520

Covert Channel Pada Aliran Data Websocket untuk Komunikasi Messaging XMPP

Yoga Dwitya Pramudita*[1], Reza Pulungan[2]
[1]Prodi S2Ilmu Komputer, FMIPA UGM, Yogyakarta
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: *[1]**yoga.dp@gmail.com**, [2]reza.pulungan@ugm.ac.id

Abstrak

Layanan komunikasi Instant Messaging menyediakan berbagai fitur komunikasi yang bisa digunakan oleh pengguna, diantaranya adalah text messaging (pesan teks) baik online maupun offline. Salah satu standar protokol yang mendukung layanan ini adalah XMPP (Extensible Messaging and Presence Protocol). Aliran komunikasi XMPP menggunakan potongan dokumen XML, sehingga rentan terhadap serangan pasif monitoring konten paket komunikasi. Untuk mengatasi kelemahan ini solusinya adalah menggunakan komunikasi yang terenkripsi. Selain itu ada solusi lain yang coba ditawarkan dalam penelitian ini, yaitu penggunaan covert channel untuk mengirim pesan secara tersembunyi. Dalam penelitian ini akan dibuat sebuah aplikasi klien XMPP berbasis web browser yang mampu melakukan komunikasi XMPP dan juga menyediakan komunikasi covert channel. Komunikasi XMPP agarbisa berjalan diatas aplikasi berbasis web browser maka digunakanlah protokol WebSocket. Protokol inilah yang nantinya akan dieksploitasi pada sisi header, khususnya pada field masking-key untuk memuat pesan covert channel yang dikirimkan pada saat sesi komunikasi XMPP berlangsung. Dari hasil ujicoba, aplikasi klien covert channel mampu menghasilkan komunikasi dengan lebar data 3 byte perpaket. Aplikasi Klien juga mampu melakukan komunikasi covert channel pada kondisi link komunikasi dengan tingkat probabilitas packet loss dibawah 10%.

Kata kunci— *WebSocket, XMPP, masking-key, Covert Channel, aplikasi klien berbasis browser.*

Abstract

Instant Messaging communication services provide a variety of communication features that can be used by the user, such as text messaging (text messages) both online and offline. One of the standard protocol that supports this service is XMPP (Extensible Messaging and Presence Protocol). XMPP communication using XML documents, making it vulnerable to passive attacks monitoring content of communications. To overcome this drawback the solution is encrypted communications. The other solutions that try to offer in this research is the use of a covert channel to send hidden messages. In this research will create a browser based XMPP client application that is capable to deliver XMPP communication and also provide covert channel communication. XMPP communication can be built on a web-based application using WebSocket protocol. This protocol will exploit field masking-key to load the covert channel messages that is sent during the session XMPP communication takes place. From the test results, the client application is able to produce a covert channel communication with a data width of 3 bytes in each packet. The client application is also able to perform covert communication channel in a communication link with the condition of the probability of packet loss rate below 10%.

Keywords— *WebSocket, XMPP, masking-key, Covert Channel, browser based application.*

Received November 27th,2014; Revised January 14th, 2015; Accepted July 10th, 2015

1. PENDAHULUAN

Layanan komunikasi Instant Messaging menyediakan berbagai fitur komunikasi yang bisa digunakan oleh pengguna, diantaranya adalah text messaging (pesan teks) baik online maupun offline. Salah satu standar protokol yang mendukung layanan ini adalah XMPP (Extensible Messaging and Presence Protocol). Protokol XMPP bekerja dengan cara mengirimkan pesan berupa potongan-potongan (*subset dari child* tingkat pertama) dokumen XML (Extensible Markup Language) yang disebut sebagai stanza. Karena aliran komunikasi XMPP menggunakan potongan-potongan dokumen XML, maka komunikasi ini rentan terhadap serangan pasif dari keamanan komunikasi yaitu serangan oleh pihak ketiga dengan memonitor lalu-lintas data dengan tujuan untuk mengetahui konten informasi yang sedang dikirimkan.

Untuk merahasiakan konten pesan pada komunikasi XMPP, maka secara umum bisa menggunakan enkripsi pada semua koneksi XMPP yang sedang berlangsung. Pendekatan lain yang bisa digunakan untuk menghalangi pihak ketiga memonitor konten informasi yang sedang dikirim adalah dengan menyembunyikan konten pesan dengan menggunakan *covert channel*.

Covert channel itu sendiri bisa diartikan sebagai sebuah serangan dimana dua entitas dapat berkomunikasi dengan memanipulasi sumber daya bersama dengan cara-cara yang tidak diinginkan, dan bisa membahayakan aset penting (dalam hal ini adalah informasi) [1]. *Covert channel* bisa dibangun menggunakan *header* protokol sebagai pembawa pesan, salah satunya adalah penggunaan protokol TCP, IP, dan ARP [2], [3], [4], [5], [6], [7], [8], penggunaan HTTP untuk membangun komunikasi covert channel juga mungkin dilakukan [9]. Pemanfaatan covert channel untuk FTP (*File Transfer Protocol*) juga telah diteliti [6], dimana mereka membuat aplikasi FTP berbasis covert channel dengan memanfaatkan field IP record route.

Pada dasarnya protokol komunkasi XMPP bisa berdiri sendiri. Tapi untuk mengakomodir komunikasi dengan menggunakan aplikasi berbasis *browser* maka digunakanlah protokol *WebSocket.* Struktur framing pada WebSocket terdiri dari header dan payload yang nantinya membawa message dari klien ke server atau sebaliknya. Draft IETF merekomendasikan penggunaan frame masking pada komunikasi klien ke server menggunakan WebSocket. Masking-key (4 byte data pada field header) dibuat pada sisi klien pada saat pesan akan dikirim dan nilainya bisa berubah. Disatu sisi penggunaan *masking-key* bisa menutupi celah keamanan yang diakibatkan oleh *cache poisoning* karena menghasilkan payload yang selalu berubah, disisi lain hal ini berpotensi digunakan untuk membangun *covert channel*. Dalam penelitian ini *covert channel* digunakan untuk mengirimkan pesan tersembunyi pada komunikasi XMPP, tanpa merubah struktur protokol XMPP yang digunakan saat komunikasi berlangsung. Covert channel yang diajukan dalam penelitian ini menggunakan pendekatan memanipulasi *header* pada protokol komunikasi websocket.

2. METODE PENELITIAN

2.1 Analisa

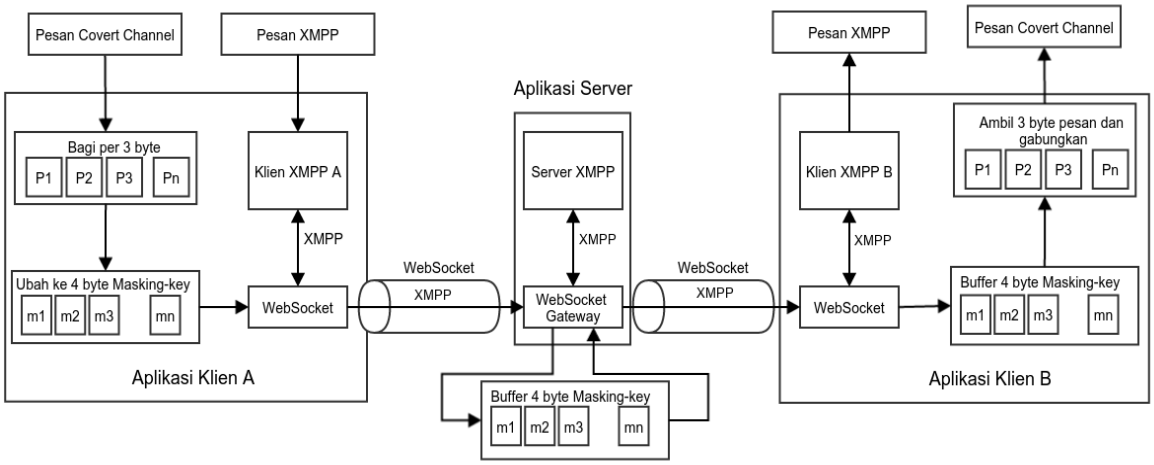
Di dalam membangun *covert channel* ada beberapa permasalahan yang muncul, baik dari segi pesan yang akan dikirimkan, maupun dari bagaimana mengirimkan pesan dari klien ke server dan dari server ke klien tujuan. Pesan yang nantinya dikirim secara tersembunyi melalui covert channel tidak terbatas pada pesan teks saja, tapi bagaimana pesan dalam bentuk file bisa juga dikirimkan. Selain dari tipe pesan yang akan dikirimkan, ukuran pesan juga menjadi kendala tersendiri, hal ini dikarenakan lebar *field masking-key* pada protokol *WebSocket* hanya 4 byte (32 bit), sedangkan ukuran pesan yang dikirimkan nanti akan lebihdari 4 byte.Pengirimanpesan melalui covert channel pada protokol *Websocket* harus mempunyai pola komunikasi yang bisa dikenal oleh semua anggota yang terlibat dalamkomunikasi *covert*

channel, baik dari segi bagaimana komunikasi dilakukan, dan bagaimana mengenali bahwa paket tersebut adalah paket yang membawa informasitersembunyi.

Untuk menjawab beberapa permasalahan diatas maka perlu adanya suatu rancangan mengenai arsitektur komunikasi menggunakan *covert channel* dan bagaimana komunikasi tersebut dilakukan oleh semua pihak yang menjadi bagian dari skenario komunikasi covert channel.

2. 2 Rancangan Covert Channel

Dalam penelitian ini akan dikembangkan komunikasi tersembunyi melalui *covert channel* dengan mengeksploitasi header paket WebSocket. Penggunaan protokol WebSocket akan dimanfaatkan untuk mengirimkan pesan tersembunyi seoalah-olah paket yang dikirim adalah paket komuniksai normal pada umumnya. Untuk bisa berkomunikasi dengan menggunakan *covert channel* maka semua pihak yang terlibat komunikasi akan dieksploitasi oleh proses yang melayani *covert channel*, baik klien maupun server XMPP.



Gambar 1 Rancangan Covert Channel Pada Komunikasi XMPP

Pada Gambar 1 terdapat dua bagian utama dari desain sistem komunikasi *covert channel.*

1. Pertama adalah klien A dan B yang nantinya sebagai pihak pengirim pesan dan penerima pesan tersembunyi. Sebelum membangun covert channel klien A akan terkoneksi terlebih dahulu ke server XMPP yang sudah mendukung protokol WebSocket.
2. Kedua adalah server XMPP yang sudah mendukung protokol WebSocket. Server disini mempunyai peranan penting dimana selain melakukan routing protokol XMPP dari klien ke klien, server juga berperan penting dalam proses pengiriman dan penerimaan paket-paket data yang didalamnya terdapat informasi tersembunyi.

Pada sisi pengirim, pengguna bisa mengirimkan informasi tersembunyi berupa file maupun pesan teks. Hal ini dilakukan pada saat sesi komunikasi dengan menggunakan XMPP sedang berlangsung. Pada sisi penerima selain menerima pesan XMPP juga akan mencoba mengenali tiap paket komunikasi yang masuk ke dalam sistem untuk mengetahui apakah ada pesan tersembunyi dalam tiap paket yang diterimanya. Pada sisi server akan dilakukan proses penerimaan dan pengiriman kembali paket-paket tertentu yang berisi pesan tersembunyi.

Proses pengiriman pesan tersembunyi akan memanfaatkan paket ping dari klien ke server. Sedangkan untuk mengirim pesan rahasia dari server ke klien memanfaatkan paket pesan teks. Proses pengiriman tidak dilakukan secara otomatis tetapi dengan melakukan pengunduhan secara manual oleh klien dengan cara mengirim paket pesan teks kosong (sebagai pemicu) ke

server untuk kemudian akan dibalas dengan paket teks kosong juga oleh server, dimana paket yang dikirim ke klien akan disisipkan informasi rahasia pada *field masking-key.*

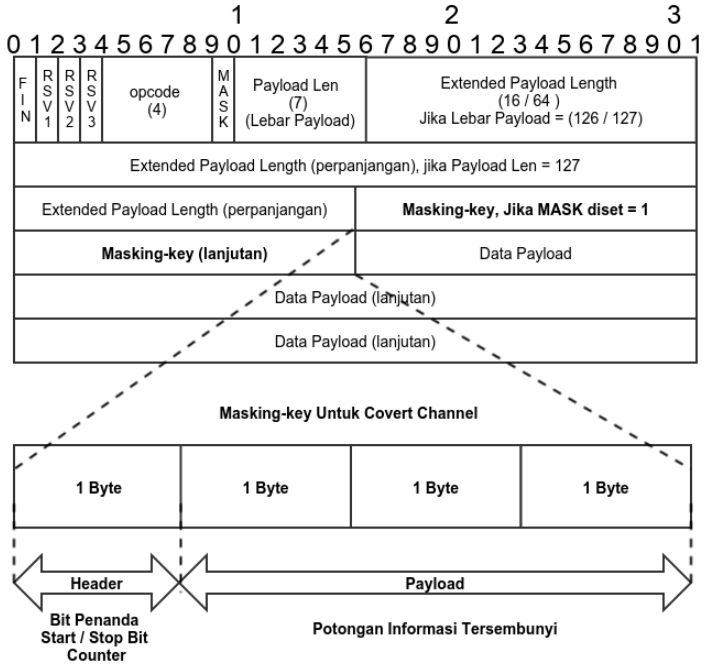
2. 2.1 Rancangan Algoritma

Secara umum proses *covert channel* dibagi menjadi dua, proses pada sisi klien, dan proses pada sisi server. Pada sisi Klien terdiri dari proses kirim pesan dan unduh pesan *covert channel*. Sedangkan pada sisi server terdiri dari proses terima pesan dan kirim pesan *covert channel*. Untuk proses kirim pesan dari klien ke server memanfaatkan paket ping sedangkan dari server ke klien memanfaatkan paket teks dengan *payload null.*

2. 2.2 Algoritma Klien

Proses komunikasi *covert channel* diawali dari klien pengirim pesan tersembunyi. Pesan rahasia yang akan dikirim terlebih dahulu akan diperiksa, apakah pesan tersebut adalah pesan teks atau pesan dalam bentuk file. Jika masih dalam bentuk file, maka harus dirubah terlebih dahulu ke bentuk teks dengan menggunakan algoritma base64, setelah itu lakukan pemotongan setiap 3 byte terhadap pesan yang akan dikirim.

Ubah potongan 3 byte pesan kedalam bentuk desimal menggunakan kode ASCII (merubah menjadi desimal bertujuan mempermudah konversi pesan ke bentuk 4 byte data masking-key). Buat buffer (buffer merupakan sebuah class mirip dengan array dari data integer tetapi berkaitan langsung dengan alokasi *raw memory* diluar heap v8) *masking-key* menggunakan potongan 3 byte pesan dengan menambahkan 1 byte informasi diawal potongan guna keperluan pengiriman dan penerimaan paket pesan, seperti tampak pada Gambar 2. Buat buffer paket ping dengan menambahkan *opcode*(semacam flag yang menandakan bahwa itu adalah paket kontrol pong) pong diikuti dengan *buffer masking-key*. Kirim sebanyak potongan 3 byte pesan yang dihasilkan.



Gambar 2 Struktur Penggunaan Masking-key Pada Covert Channel

Pada saat klien mengirim pesan, yang terjadi adalah :

1. Pesan bisa berupa teks atau file.
2. Jika pesan berupa teks langsung maka ada beberapa langkah yang dilakukan :

a. Pesan akan dibagi menjadi beberapa potongan pesan dengan ukuran 3 byte tiap potongan.
b. Jika pesan ada yang kurang dari 3 byte, maka akan ditambahkan data null dibelakangnya.
   Contoh : apa kabarmu = (a,p,a),(,k,a),(b,a,r),(m,u,)
c. Kemudian bit pesan akan dikonversi ke nilai desimal dengan menggunakan table ascii.
   Contoh : apa kabarmu = (97,112,97) , (32,107,97) , (98,97,114) , (109,117,0)
d. Kemudian menambahkan header pada masing-masing potongan informasi.
   = (128+64+0,97,112,97) , (128+64+1,32,107,97) , (128+64+2,98,97,114 (128+0+3,109,117,0)
   = (192,97,112,97) , (193,32,107,97) , (194,98,97,114) , (131,109,117,0)
e. Setelah data selesai dirubah ke desimal, kemudian data dirubah ke biner untuk kemudian disisipkan ke header WebSocket sebagai masking-key dari paket pong yang nantinya akan dikirimkan oleh klien pengirim (A).
   Contoh : (192,97,112,97) = 11000000011000010111000001100001 (masking-key baru yang berisi potongan pesan tersembunyi).

3. Jika pesan dalam bentuk file maka ada beberapa langkah yang dilakukan :
   a. Konversi dari file ke mime base64.
      Contoh : data.txt = YXBhIGthYmFybXUK
   b. Bagi menjadi per 3 byte pesan
      Contoh : = (Y,X,B),(h,I,G),(t,h,Y),(m,F,y),(b,X,U),(K,,)
   c. Ubah ke nilai desimal dengan menggunakan table ascii.
      Contoh : = (89,88,66) , (104,73,71) , (116,104,89) , (109,70,121) , (98,88,85) , (75,0,0)
   d. Kemudian menambahkan header pada masing-masing potongan informasi.
      Contoh : = (192,89,88,66) , (193,104,73,71) , (194,116,104,89) , (195,109,70,121) , (196,98,88,85) , (133,75,0,0)
   e. Data dirubah ke biner kemudian disisipkan pada header paket WebSocket pada bagian masking-key dan dikirimkan ke server.
      Contoh : (192,89,88,66) = 11000000010110010101100001000010 (masking-key baru yang berisi potongan pesan tersembunyi).

Ketika *masking-key* baru yang berisi potongan pesan *covert channel* sudah disiapkan, maka hal yang dilakukan berikutnya adalah membuat paket pong untuk membawa masking-key tadi untuk sampai ke server. Paket pong yang akan dibuat terdiri dari field FIN dengan nilai bit 1, diikuti oleh 3 bit 0 field RSV1,2,3, kemudian nilai opcode untuk pong 0xA (1010), diikuti oleh field mask dengan nilai bit 1, diikuti field payload length (payload null) dengan nilai bit 0000000, diikuti dengan masking- key (192,97,112,97 = 11000000 01100001 01110000 01100001), diikuti dengan payload (null untuk paket ping pong yang digunakan), jadi paket pong baru yang akan dikirim ke server adalah 10001010 10000000 11000000 01011001 01011000 01000010, atau jika dalam bentuk heksadesimal 8A80C0617061.

Pada sisi penerima jika ingin menerima pesan rahasi maka yang harus dilakukan adalah dengan mendownload pesan rahasia dengan menggunakan kode berupa pesan kosong kepada server, adapun detail langkah yang dilakukan adalah:

1. Untuk mengunduh potongan-potongan pesan rahasia klien akan mengirimkan sebuah pesan kosong ke server sebagai trigger, kemudian akan dijawab oleh server dengan pesan kosong sejumlah banyaknya masking-key yang membawa potongan informasi yang tersimpan di server.
2. Paket yang membawa pesan kosong tadi akan diambil data masking-key-nya untuk kemudian diolah menjadi informasi utuh di sisi klien penerima.

contoh : masking-key = 11000000011000010111000001100001

3. Masking-key akan di buffer, kemudian diurutkan sesuai counter, dan selanjutnya diambil payload-nya (3 byte terakhir dari masking-key).
   Contoh :
   masking-key = 11000000 01100001 01110000 01100001
   masking-key = (192,97,112,97) dalam bentuk desimal
   potongan pesan = (192-192,97,112,97) = (0,97,112,97) hasil menunjukkan potongan pertama dari 3 byte pesan *covert channel*.
   potongan pesan = (97,112,97) = (a,p,a) dalam bentuk ascii.
4. Kemudian potongan informasi tadi dirangkai kembali dan dilakukan proses decode dari data stream biner ke karakter ascii.
5. Untuk informasi hasil bisa berupa pesan teks atau pesan encode base64.
6. Untuk pesan *encode* base64 perlu dilakukan *decode* lagi untuk mengetahui informasi yang di terima.

2. 2.3 Algoritma Server

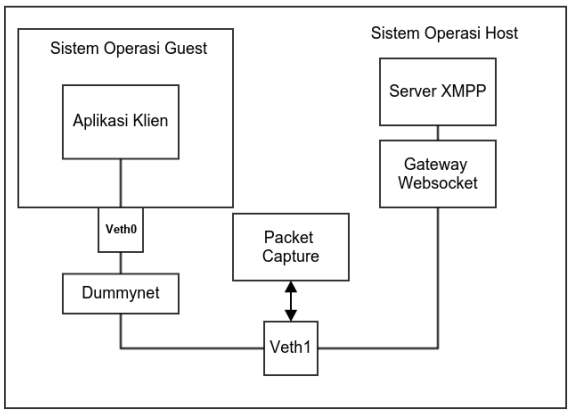
Server dalam komunikasi covert channel berperan sebagai media penyimpan sementara pesan tersembunyi dari klien pengirim ke pada klien penerima. Yang dilakukan pada sisi server ketika ada pesan rahasia yang masuk adalah :

1. Melakukan pengecekan terhadap tiap-tiap paket yang masuk, apakah masking-key yang ada pada paket mempunyai header dengan nilai diatas 127, jika ya akan dilakukan buffer.
2. Ketika klien mengirimkan paket dengan payload kosong, server akan membalas dengan mengirimkan paket teks menggunakan masking-key yang sudah ter-buffer sebelumnya.
3. Pesan yang ter-buffer di server masih akan terus tersimpan sampai server melakukan buffer terhadap pesan baru.

2. 3 Skenario Ujicoba

Ujicoba akan dilakukan dengan menggunakan beberapa skenario link komunikasi menggunakan *packet delay*, *packet loss*, dan *multipath.* Penggunaan *packet delay* disini bertujuan untuk mengetahui apakah link dengan nilai delay berbeda akan mempengaruhi waktu yang dibutuhkan untuk mengirim semua potongan pesan covert channel dari pengirim ke penerima (klien ke server atau dari server ke klien). Penggunaan packet loss bertujuan untuk mensimulasikan terjadinya packet retransmisi dimana terjadi pengiriman packet yang sama secara berulang, dan apakah kondisi ini mempengaruhi penerimaan pesan covert channel secara keseluruhan, apakah pesan bisa diterima dengan utuh jika ada paket yang tidak sampai ke tujuan. Penggunaan multipath (multilink) bertujuan menciptakan situasi dimana paket dirimkan melewati link yang berbeda, sehingga muncul kondisi dimana paket yang datang urutannya berbeda pada saat paket tersebut dikirimkan, dan apakah jika ada potongan pesan yang out of order di sisi penerima akan mempengaruhi penerimaan pesan covert channel secara keseluruhan, apakah nantinya urutan pesan akan menjadi terbalik atau tidak. Untuk menyediakan link komunikasi dengan sekenario berbeda maka penggunaan network emulator menjadi salah satu solusi yang bisa digunakan.

Untuk kebutuhan ujicoba, maka yang dipantau adalah komunikasi dari klien A ke server baik untuk komunikasi kirim pesan tersembunyi maupun terima pesan tersembunyi. Ketika ujicoba komunikasi dilakukan dengan menggunakan virtulaisasi, maka secara fisik kedua titik komunikasi tersebut berada dalam satu Sistem Operasi host seperti yang tampak pada Gambar 3.



Gambar 3 Rancangan Ujicoba Komunikasi

Nilai delay yang digunakan dalam skenario ujicoba akan mengacu pada penelitian yang dilakukan oleh [10], tentang kebutuhan qos dari aplikasi jaringan pada internet khususnya aplikasi berbasis *enchanced web browsing*. Pada *Enchanced web browsing* nilai waktu respon kisaran 2-4 detik disarankan masih bisa diterima oleh sebagian besar pengguna. Untuk mendapatkan waktu respon pada kisaran 2 - 4 detik usaha terbaik yang harus dicapai untuk spesifikasi delay yang digunakan adalah kurang dari 400 ms.

Skenario komunikasi pada jaringan dengan menggunakan link *emulator Dummynet* [11] akan dibagi menjadi delapan skenario utama seperti pada Tabel 1. Tujuannya adalah mengetahui bagaimana covert channel jika diimplementasikan pada komunikasi jaringan dengan kondisi link berbeda tingkat delay, loss dan out of order.

Tabel 1 Skenario Ujicoba Covert Channel

No	Skenario	Tujuan
1	Komunikasi single link tanpa delay dan packet loss	Mengetahui bagaimana komunikasi covert channel bekerja tanpa dipengaruhi delay, loss dan reorder
2	Komunikasi single link dengan delay komunikasi 100ms	Mengetahui pengaruh delay terhadap komunikasi
3	Komunikasi single link dengan delay komunikasi 100ms dan packet loss 10%	Mengetahui pengaruh delay dan loss terhadap komunikasi
4	Komunikasi single link dengan delay komunikasi 100ms dan packet loss 20%	Mengetahui Pengaruh delay dengan loss yang ditingkatkan
5	Komunikasi single link dengan delay komunikasi 100ms dan packet loss 30%	Pengaruh delay dengan loss yang ditingkatkan
6	Komunikasi multi link dengan delay 100ms, 200ms, 400ms	Pengaruh multipath dengan delay berbeda
7	Komunikasi multi link dengan delay 100ms, 200ms, 400ms dan packet loss 10%	Pengaruh multipath dengan delay berbeda dengan loss
8	Komunikasi multi link dengan delay 100ms, 200ms, 400ms dan packet loss 20%	Pengaruh multipath dengan delay berbeda dengan loss yang ditingkatkan

Tolak ukur pengujian ada di kinerja pengiriman dan waktu yang digunakan. Kinerja pengiriman disini adalah bahwa prototipe bisa menyediakan c*overt channel* dan berhasil dalam mengirimkan potongan pesan *covert channel* secara utuh dari pengirim ke penerima. Sedangkan waktu disini adalah rentang waktu yang dibutuhkan untuk mengirim pesan *covert channel* dari pengirim ke penerima diukur dari salah satu sisi (pengirim atau penerima).

3. HASIL DAN PEMBAHASAN

Pada ujicoba ini pesan yang digunakan adalah pesan teks yang disimpan dalam file dengan ukuran file 132 byte, yang terdiri dari 132 karakter. Untuk bisa mengirimkan data dalam bentuk file (ascii file dengan data biner dalam bentuk kode ascii), maka perlu dilakukan konversi dari bentuk biner ke bentuk teks dengan menggunakan encoding base64 sehingga mendapatkan hasil sebuah teks dengan panjang 176 bytes. Kemudian teks tersebut akan diolah menjadi potongan-potongan data dalam bentuk kode ascii dan akan ditanamkan pada data *masking-key* selebar 4 byte, sehingga data yang akan dikirim ke server dibagi menjadi 59 paket *masking-key*. Dari kedelapan skenario yang diujicobakan ada yang gagal dan ada yang berhasil.

Hasil pemantauan terhadap kedelapan skenario yang diujicobakan didapatkan data pada Tabel 2 untuk komunikasi dari klien ke server menggunakan paket ping pong. Sedangkan untuk pemantauan terhadap komunikasi dari server ke klien dengan menggunakan paket teks didapatkan data pada Tabel 3.

Tabel 2 Hasil Ujicoba Paket Ping Pong

No	Link	Delay	Packet Loss	Total Paket	Waktu (s)	Status
1	1	0	0	118	116	Sukses
2	1	100	0	118	116	Sukses
3	1	100	10	136	116	Sukses
4	1	100	20	144	116	Sukses
5	1	100	30	0	0	Gagal
6	3	100, 200, 400	0	118	116	Sukses
7	3	100, 200, 400	10	129	116	Sukses
8	3	100, 200, 400	20	0	0	Gagal

Tabel 3 Hasil Ujicoba Paket Teks

No	Link	Delay	Packet Loss	Total Paket	Waktu (s)	Status
1	1	0	0	59	58	Sukses
2	1	100	0	59	58	Sukses
3	1	100	10	64	58	Sukses
4	1	100	20	38	21	Gagal
5	1	100	30	0	0	Gagal
6	3	100, 200, 400	0	59	58	Sukses
7	3	100, 200, 400	10	67	58	Sukses
8	3	100, 200, 400	20	80	59	Sukses

Hasil ujicoba menunjukkan bahwa, ujicoba dengan menggunakan delay yang berbeda (skenario pertama, kedua, dan keenam) pada link emulasi tidak berpengaruh signifikan terhadap proses pengiriman potongan paket. Hal ini bisa dilihat pada Tabel 2 dimana untuk mengirim pesan covert channel secara utuh membutuhkan jumlah paket ping pong sebanyak 118 dengan lama proses pengiriman potongan dari klien ke server sebesar 116 s. Tetapi ketika *packet lost* dilibatkan dalam proses emulasi link komunikasi, maka akan berpengaruh pada banyaknya paket yang dibutuhkan untuk mengirim potongan pesan, hal ini disebabkan oleh proses retransmisi paket ping pong selama mengirim pesan *covert channel.*

Packet lost dengan nilai probabilitas lebih besar (skenario kelima dan kedelapan) berpotensi menggagalkan upaya pengiriman potongan pesan melalui covert channel, seperti digambarkan pada Tabel 2. Hal ini disebabkan karena semakin besar probabilitas packet loss semakin besar kemungkinan paket ping pong gagal dikirim dan diterima. Penggunaan emulasi link lebih dari satu juga tidak berpengaruh secara signifikan, data pada skenario pertama, kedua, dan keenam memiliki data hasil pengamatan yang sama baik jumlah paket ping pong, waktu yang dibutuhkan, dan potongan pesan yang dikirimkan bisa diterima sesuai dengan urutan pesan yang dikirimkan.

Kegagalan komunikasi pada skenario yang melibatkan *packet loss* (skenario kelima dan kedelapan) dikarenakan sesi komunikasi *WebSocket* sangat bergantung pada proses saling bertukar control packet berupa paket ping pong. Jika klien tidak bisa membalas paket ping dari server untuk beberapa kali putaran (dalam kasus penelitian ini dua kali berturut-turut) maka koneksi akan di-drop oleh server, karena dianggap klien sudah tidak aktif lagi dalam sesi komunikasi.

Pada ujicoba menggunakan paket teks skenario yang gagal ada pada skenario keempat dan kelima sedangkan pada skenario kedelapan dengan paket lost 20% komunikasi masih bisa berjalan. Pada skenario keempat dengan paket lost 20% komunikasi masih bisa berjalan, akan tetapi pesan tidak bisa dikirim seluruhnya dikarenakan sesi komunikasi berhenti sebelum keseluruhan pesan dikirimkan.

Seperti pada hasil ujicoba sebelumnya, semakin besar paket lost yang diterapkan, maka semakin banyak pula paket yang ditransmisikan ulang, seperti yang tampak pada Tabel 3. Semakin banyak paket yang dibutuhkan untuk mengirimkan pesan *covert channel* disebabkan karena banyak paket yang ditransmisi ulang. Besarnya *packet loss* juga mempengaruhi sukses atau tidaknya pengiriman pesan melalui *covert channel*, semakin besar packet loss kemungkinan pengiriman dan penerimaan *control packet* ping pong gagal makin besar juga.

Dari kedelapan skenario yang diujicobakan baik pada paket ping pong maupun paket teks, skenario dikatakan berhasil jika aplikasi sukses mengirim pesan *covert channel* ke server dan sekaligus mengunduh pesan dari server (skenario satu, kedua, ketiga untuk single link, dan skenario enam dan tujuh untuk link sebanyak tiga link). Skenario ujicoba dikatakan gagal ketika aplikasi terkendala dalam mengirim pesan atau mengunduh pesan *covert channel.*

4. KESIMPULAN

Berdasarkan hasil penelitian mengenai *Covert Channel* pada Aliran Data WebSocket untuk Komunikasi *Messaging XMPP* bisa ditarik kesimpulan bahwa :

1. Komunikasi *covert channel* pada messaging XMPP telah dibangun dengan menggunakan aplikasi klien XMPP yang sudah dimodifikasi untuk bisa mengirimkan pesan tersembunyi, yaitu dengan memanfaatkan field *masking-key* pada header paket-paket protokol *WebSocket.*
2. Komunikasi *covert channel* mampu mengirimkan pesan sebesar 3 byte untuk tiap paket yang dikirim dari klien ke server atau sebaliknya.
3. Aplikasi klien *covert channel* dan server bisa diimplementasikan pada jaringan dengan kondisi *packet loss* tidak lebih dari 10% baik untuk link masuk atau keluar.

4. Pada link jaringan dengan kondisi delay berbeda namun tetap berada dibawah 400ms tidak berpengaruh secara signifikan terhadap waktu yang dibutuhkan untuk mengirim dan menerima pesan *covert channel*. Begitu juga ketika Aplikasi dijalankan pada kondisi jaringan menggunakan link lebih dari satu tidak mempengaruhi komunikasi *covert channel*.

5. SARAN

Proses pengiriman dan penerimaan pesan *covert channel* dalam penelitian ini bergantung pada bagaimana memanfaatkan *field masking-key* pada *header* protokol WebSocket. Pada proses server mengirim pesan *covert channel* ke klien juga penggunaan field *masking-key*. Menurut draft standar protokol Websocket paket yang dikenakan masking adalah paket yang berasal dari klien ke server. Meskipun implementasinya sedikit berbeda dengan draft standar protokol WebSocket hal ini tidak berpengaruh terhadap sesi komunikasi XMPP yang sedang berlangsung. Maka perlu dikembangkan solusi lain dengan melibatkan protokol komunikasi selain WebSocket untuk proses pengiriman pesan *covert channel* dari server ke klien.

DAFTAR PUSTAKA

[1] Okhravi, H., Bak, S., dan King, S., 2010, Design, implementation and evaluation of covert channel attacks, *2010 IEEE International Conference on Technologies for Homeland Security (HST)*, hal 481–487.

[2] Rowland, C.H., 1997, Covert channels in the TCP/IP protocol suite, *First Monday*, vol 2, no 5, ISSN 13960466, http://firstmonday.org/ojs/index.php/fm/article/view/528.

[3] Giffin, J., Greenstadt, R., Litwack, P., dan Tibbetts, R., 2002, Covert messaging through TCP timestamps, *in Workshop on Privacy Enhancing Technologies*, 194–208.

[4] Qu, H., Su, P., dan Feng, D., 2004, A typical noisy covert channel in the IP protocol, *38th Annual 2004 International Carnahan Conference on Security Technology*, 189–192.

[5] Zander, S., Armitage, G., dan Branch, P., 2006, Covert channels in the IP time to live field, Proceedings of Australian Telecommunication Networks and Applications Conference (ATNAC).

[6] Trabelsi, Z. dan Jawhar, I., 2010, Covert file transfer protocol based on the ip record route option, Information Assurance and Security, 5, 64–73, http://faculty.uaeu.ac.ae/ijawhar/publications/covert%20ftp%20journ.pdf.

[7] Ji, L., Fan, Y., dan Ma, C., 2010, Covert channel for local area network, 2010 IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS), 316–319.

[8] Jankowski, B., Mazurczyk, W., dan Szczypiorski, K., 2010, Information Hiding Using Improper Frame Padding, CoRR, abs/1005.1925, http://arxiv.org/abs/1005.1925.

[9] Duncan, R. dan Martina, J.E., 2010, Steganographic Message Broadcasting using Web Protocols, In Proc. of SBSeg 2010 - Simposio Brasilerio de Seguranca, Fortaleza, Brasil.

[10] Chen, Y., Farley, T., dan Ye, N., 2004, QoS Requirements of Network Applications on the Internet, Inf. Knowl. Syst. Manag., 4, 1, 55–76, ISSN 1389-1995, http: //dl.acm.org/citation.cfm?id=1234242.1234243.

[11] Rizzo, L., 2014, Portable packet processing modules for OS kernels, Network, IEEE, 28, 2, 6–11.