Implementasi Skema SecLaaS-RW dalam Membuat Aplikasi Secure *Log*ging

Giovanni Victo Araya[1)], Setiyo Cahyono[2)]

1) Rekayasa Keamanan Siber, Politeknik Siber dan Sandi Negara, giovanni.victo@student.poltekssn.ac.id

2) Rekayasa Keamanan Siber, Politeknik Siber dan Sandi Negara, setiyo.cahyono@poltekssn.ac.id

Abstrak

Kegiatan pengumpulan log file yang disebut logging, dan analisis log file sangat diperlukan oleh setiap organisasi. Oleh karena itu setiap organisasi dianjurkan untuk membangun infrastruktur log meliputi software, hardware, jaringan, dan media untuk melakukan pembangkitan, penyimpanan, analisis, hingga penghapusan log. Namun, dalam membuat infrastruktur log tersebut tidak mudah. Terdapat tantangan dan persoalan yang perlu dihadapi dalam membangun dan mengoperasionalkan infrastruktur log. Tantangan tersebut semakin besar ketika suatu organisasi ingin menggunakan cloud untuk mendukung infrastruktur log. Hal tersebut dikarenakan log yang tersimpan pada cloud rawan terhadap ancaman modifikasi, intersepsi dan fabrikasi terhadap data log. Salah satu cara untuk melakukan proteksi pada saat penyimpanan log adalah dengan menggunakan aplikasi secure logging. Pada penelitian ini diimplementasikan skema SecLaaS-RW untuk menghasilkan aplikasi secure logging. Implementasi skema SecLaaS-RW dilakukan menggunakan Python 3 dengan memanfaatkan library python yang telah tersedia. Skema SecLaaS-RW diimplementasikan menjadi 2 aplikasi yaitu aplikasi client dan aplikasi server. Aplikasi yang dihasilkan pada penelitian ini untuk mendukung proses bisnis organisasi meliputi secure logging, manajemen log, dan manajemen pengguna. Aplikasi yang dibangun dapat dijalankan pada sistem operasi Windows, Ubuntu/ Linux, dan MacOS. Pada penelitian ini juga dijelaskan bagaimana cara mengimplementasikan skema SecLaaS-RW menjadi aplikasi secure logging.

Kata kunci: *log* (1), *software* (2), *secure logging* (3), SecLaaS-RW (4)

Abstract

Log file collection activities which called logging and log file analysis are very necessary by every organization. Therefore, every organization is advised to build a log infrastructure including software, hardware, network, and media to do activities such as generation, storage, analysis, and log deletion. However, creating a log infrastructure is not easy. There are challenges and issues that need to be addressed in building and operating a log infrastructure. The challenge is even greater when an organization wants to use the cloud to support its log infrastructure. This is because the logs stored in the cloud are vulnerable to data lag modification, interception, and fabrication. The solution to protect against log storage is to use a secure logging application. In this research, the SecLaaS-RW scheme is implemented to make a secure logging application. The implementation of the SecLaaS-RW scheme is done using Python 3 and by utilizing the available python libraries. The SecLaaS-RW scheme is implemented into 2 applications, namely a client application and a server application. The application produced in this study is able to support the organization's business processes including secure logging, log management, and user management. Applications that are built can run on Windows, Ubuntu/Linux, and MacOS operating systems. This research also explains how to implement the SecLaaS-RW scheme into a secure logging application

Keywords: *log (1), software (2), secure logging (3), SecLaaS-RW (4)*

1. PENDAHULUAN

Kegiatan pengumpulan *log file*, yang biasa disebut *logging*, dan analisis *log file* sangat berguna ketika dilakukan audit, forensik, investigasi internal organisasi, pembuatan *baseline* yaitu standar minimal batas aman organisasi, dan pengidentifikasian *trend* masalah yang terjadi pada suatu organisasi. Hal tersebut dikarenakan pada *log file* terdapat *log entries* yang berisi informasi spesifik kejadian yang terjadi pada suatu sistem. Dengan menggunakan metode analisis *log* terhadap *log* yang telah dikumpulkan, maka dapat digunakan untuk menemukan bukti adanya suatu tindakan kejahatan, perilaku penyerang, dan mengetahui bagaimana serangan dapat terjadi [1]. Selain itu, melakukan kegiatan analisis *log* secara rutin dapat menguntungkan suatu organisasi dalam mengidentifikasi insiden keamanan, pelanggaran kebijakan, aktivitas penipuan, dan masalah operasional yang terjadi pada organisasi tersebut [1]. Oleh karena itu, kegiatan pengumpulan dan analisis *log* menjadi suatu kegiatan penting dan pokok yang harus dilakukan oleh suatu organisasi.

Setiap organisasi dianjurkan untuk membangun infrastruktur *log* yang terdiri dari *hadware*, *software*, jaringan, dan media yang dapat digunakan untuk membangkitkan, mengirim, menyimpan, menganalisis, dan menghapus *log* [1]. Namun, dalam membuat infrastruktur *log* tersebut tidak mudah. Terdapat tantangan dan persoalan yang perlu dihadapi dalam membangun dan mengoperasionalkan infrastruktur *log*. Menurut V Raval, tantangan yang perlu dihadapi dalam membangun dan mengoperasionalkan infrastruktur *log* yaitu

keterbatasan sumber daya yang dimiliki untuk tempat penyimpanan, pemeliharaan dan perawatan infrastruktur, memastikan data yang ada pada *log* tesebut tetap utuh, memastikan tipe data yang telah menjadi standar tidak berubah, dan *log* tersebut tetap terproteksi [2]. Adapun, menurut K. Kent and M. Souppaya, tantangan yang dihadapi dalam membangun dan mengoperasikan infrastruktur *log* yaitu kerahasiaan *log*, integritas *log*, ketersediaan *log*, keragaman format atau *standard log* [1]. Oleh karena itu, diperlukan upaya untuk untuk menghadapi tantangan-tantangan tersebut.

Salah satu upaya yang dapat dilakukan untuk membangun infrastruktur *log* dan menghadapi tantangan-tantangan tersebut yaitu dengan menggunakan aplikasi *secure logging*. Dalam membuat aplikasi *secure logging* tidak dapat dilakukan sembarangan dan diperlukan suatu teknik, skema, atau mekanisme tertentu. Oleh karena itu muncul beberapa penelitian yang mengajukan skema, teknik atau mekanisme *secure logging*. Mekanisme dalam pengiriman suatu *log* melalui jaringan dilakukan dengan menggunakan *Syslog Protocol* yang pengirimannya dilakukan dari *log generato*r (sys*log client*) ke *log collector* (sys*log server*) [3].

Dalam upaya untuk membatasi kemampuan dari seorang *attacker* untuk membaca, merusak atau mengubah tanpa terdeteksi dapat digunakan teknik *hashchain* dan *Message Authentication Code* (MAC) yang diajukan oleh B. Schneier and J. Kelsey [4]. Mekanisme pengamanan pada *log* berkembang seiring dengan berkembangnya teknik *publik key cryptography*. Skema *Forward-Secure Sequential Aggregate* (FssAgg) yang diajukan oleh D. Ma and G. Tsudik dan skema LogCrypt yang diajukan oleh J Holt merupakan skema berbasis *asymetric cryptography* yang efisien dalam melakukan verifikasi terhadap *log* dan tidak perlu memerlukan pihak ketiga [5] [6]. Selanjutnya terdapat skema *Blind Aggregate Forward* (BAF) yang diajukan oleh A. A. Yavuz and P. Ning yang dianggap lebih baik dari pada skema berbasis *symmetric key* dan *asymetric key cryptography* [7]. Skema berbasis *symmetric key* memiliki kekurangan yaitu *log* tidak dapat diverifikasi secara publik dan membutuhkan *resource* lebih dalam melakukan manajemen key. Pada skema berbasis *asymmetric key* memiliki kekurangan yaitu membutuhkan resource lebih untuk melakukan komputasi terutama jika diimplementasikan pada sistem yang memiliki banyak tugas dan sumber daya terbatas. Oleh karena itu, pada skema BAF digunakan teknik *signature aggregation, key update,* dan *individual signature generation* untuk memperbaiki kekurangan tersebut karena lebih efisien dalam komputasi, *storage* dan skalabilitas.

Seiring dengan berkembangnya teknologi *cloud*, maka muncul beberapa ide untuk memanfaatkan *cloud* sebagai bagian dari infrastruktur *logging*. Hal tersebut dikarenakan *cloud* mampu memberikan keuntungan kepada penggunanya untuk dapat menghemat biaya penggunaan, perawatan, dan pemeliharaan perangkat. Oleh karena itu diajukan skema *secure logging* as *service* oleh Ray et al dan skema SecLaaS oleh Zawoad et al yang menggunakan *cloud* sebagai penyimpanan *log* [8] [9]. Skema tersebut dikembangkan oleh Khan et al menggunakan teknik *Reversible Watermarking* yang kemudian dikenal sebagai skema SecLaaS-RW [10]. Skema SecLaaS-RW ini memiliki kelebihan yaitu penggunaan *cloud* sehingga dapat menghemat biaya perawatan dan pemeliharaan perangkat. Skema tersebut juga baik jika dilihat dari aspek keamanan. Skema tersebut memenuhi persyaratan *secure logging* yaitu *Forward Integrity,* Integritas *log file,* Kebenaran *log file,* Integritas *log file, authorized user, dan* resistensi *log file.*

Untuk membuat aplikasi *secure logging* yang tepat dan dapat menjamin proteksi *log* maka dapat digunakan skema SecLaaS-RW dikarenakan kelebihan yang ditawarkan pada skema tersebut. Oleh karena itu, pada penelitian ini dilakukan implementasi skema SecLaaS-RW menjadi aplikasi *secure logging* yang dapat digunakan. Implementasi menggunakan phyton 3 karena phyton memiliki pustaka kriptografi seperti fungsi hash dan algoritme enkripsi yang diperlukan dan mudah digunakan.

2. LANDASAN TEORI

2.1. *Logging*

Logging adalah tindakan atau proses dalam mengumpulkan *log file* dari *log generator* seperti aplikasi, *firewall, router, switch,* IDPS, dan lain - lain. Contoh proses dari *logging* yaitu menyimpan *log entry* ke dalam *log file*, atau menyimpan data catatan audit dalam *file* biner atau *database* [10].

Dalam pengumpulan *log* atau proses *logging* harus dilakukan pengamanan dengan menggunakan teknik tertentu sehingga dapat mencegah atau mendeteksi apabila adanya serangan yang dapat merusak atau memodifikasi. Kemampuan atau tindakan yang dilakukan untuk melakukan pengamanan *log entry* atau *log* dengan menggunakan teknik tertentu disebut sebagai *secure logging*. Teknik dari *secure logging* harus efisien dan dapat memenuhi aspek keamanan *integrity*, *availability*, dan *confidentiality*. Terdapat beberapa syarat keamanan dari *logging* sebagai berikut [10]:

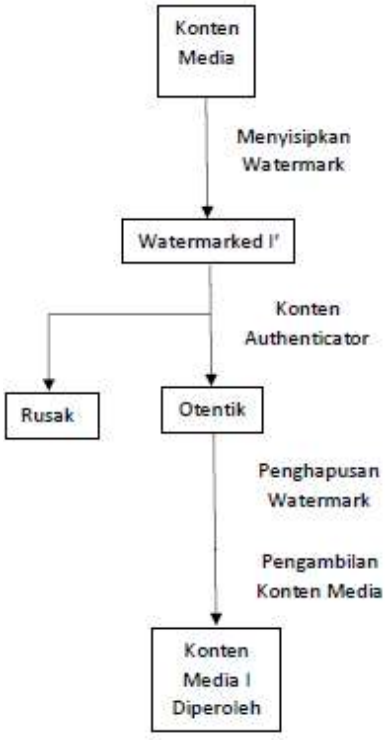
a. *Forward Integrity*: Ini adalah properti yang paling diinginkan dari setiap sistem untuk diamankan saat ini. Istilah ini pertama kali digunakan untuk audit *log* dan itu berarti bahwa jika terjadi serangan atau penyusupan, *log* yang dikumpulkan sebelumnya harus tetap utuh.
b. Integritas *log file:* Tidak ada tambahan atau manipulasi informasi yang ada pada *log file* asli. Agar memenuhi properti ini, perlu untuk memastikan bahwa *log entry* yang disimpan tidak

dapat diubah, dihapus atau ditambahkan oleh penyerang.

c. Kebenaran: *Log file* harus disimpan dalam format yang benar untuk jangka waktu yang lama dan harus memungkinkan memverifikasi integritas dan kebenarannya. Selama transmisi atau pemrosesan urutan rekaman harus tetap ada tidak diubah seperti yang ditransfer oleh pengguna.
d. Resistensi kerusakan: *Log file* harus tahan adanya kerusakan sehingga tidak ada yang dapat memodifikasi *log file*.
e. *Authorized user*: Hanya *user* yang memiliki hak saja yang dapat mengakses *log*.

2.2. *Reversible Watermarking*

Reversible Watermarking merupakan teknik *watermarking* yang memiliki cara kerja *watermark* yang telah disisipkan pada multimedia dapat diekstrak atau dihapus kemudian diperoleh multimedia konten yang asli [11]. Sama seperti seluruh teknik *fragile watermark*, teknik ini digunakan untuk autentikasi konten. Perbedaan teknik *reversible watermarking* dengan lainnya yaitu ketika *watermark* telah terdeteksi otentik maka *watermark* dapat dihapus kemudian diperoleh konten aslinya.



Gambar 1. Proses *Reversible Watermarking*

Gambar 1 merupakan ilustrasi yang menggambarkan bagaimana *reversible watermarking* bekerja. Konten Media I disisipkan dengan *watermark* sehingga menghasilkan Konten Media yang telah di-*watermark* I'. Selama pengiriman terdapat beberapa serangan yang mungkin dapat dilakukan pada konten media yang telah di-*watermark* tersebut. Setelah itu, konten *authenticator* mengecek apakah terdapat perubahan atau tidak. Jika terdapat perubahan maka *watermark* yang disisipkan pada media tersebut rusak. Jika tidak terdapat perubahan maka *watermark* tersebut dapat dihapus dan konten media original dapat diperoleh.

2.3. Skema *Secure Logging as a Service using Reversible Watermarking* (SecLaaS-RW)

Skema SecLaaS-RW adalah skema *secure log*ging yang diajukan oleh Khan et al pada tahun 2017 [10]. Skema ini merupakan pengembangan dari skema Zawoad *et al* [9]. Pada skema SecLaaS-RW dikembangkan teknik *reversible watermarking* untuk membuat perlindungan pada *log*. Dengan menggunakan teknik tersebut, maka peforma dari *secure logging* dapat meningkat**.**

Pada skema SecLaaS-RW*, logging server* memegang fungsi utama dari skema tersebut*. Logging server* bertanggung jawab dalam melakukan enkripsi, dekripsi, pembuatan *watermark* dan pengekstrakan *watermark*. Arsitektur dari skema ini terdiri dari empat pihak yaitu *logging device*, *log collector*, *logging client, logging cloud.*

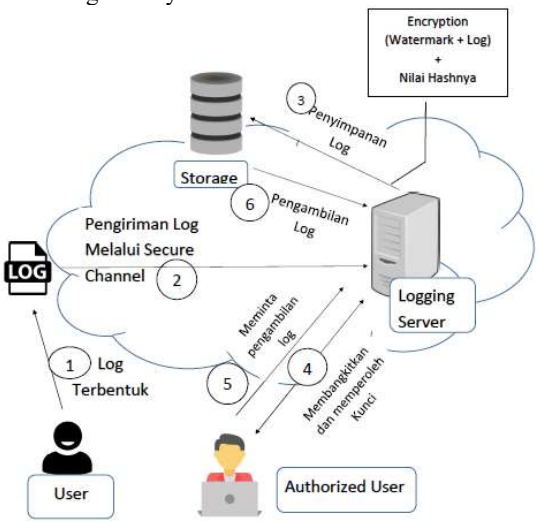
a. *Logging Device*: Merupakan perangkat yang bertanggung jawab dalam membangiktkan *event* dan *log messages*.
b. *Log Collector*: Merupakan pihak yang menerima pesan *log messages* dan bertanggung jawab dalam membuat *log file*. *Log collector* terletak pada *logging client* dan *logging server*. *Log collector* pada *client* menerima *log* yang dibuat oleh *log generator*. *Log generator* dapat terletak pada *server*, NIDS, antivirus, *router*, *switch*, dan lain - lain.
c. *Logging Client*: Merupakan pihak yang menerima *log messages* dan meneruskannya ke *log collector* lain.
d. *Logging Server/Cloud*: Merupakan pihak yang digunakan untuk menyimpan dan merawat layanan data *log* dalam jangka waktu yang lama.

Pada Gambar 2 dijelaskan skema dan alur kerja skema SecLaaS-RW. Pada skema tersebut terdapat dua pihak yang terlibat pada *secure logging*. *User* merupakan seseorang yang telah didaftarkan oleh administrator pada *logging server*. *Authorized user* merupakan *user* yang diberikan wewenang atau hak oleh *user* untuk mengakses *log*. Berikut dijelaskan alur dari skema SecLaaS-RW.

a. Pertama, *user* melakukan pembangkitan *log* dengan menjalankan *log generator*. *Log* tersebut terkumpul pada *logging client.*
b. *Log* yang telah terkumpul pada *logging client* dikirimkan ke *logging server* dalam bentuk *batch* yang berada di *cloud* melalui *secure channel.* Hanya pengguna yang terdaftar saja yang dapat memperoleh layanan.
c. Setelah itu, *log* yang dikirim disisipkan

watermark. *Log* yang telah di*watermark* kemudian dienkripsi. Selanjutnya, hasil enkripsi disimpan pada *storage. Log* yang telah diamankan tersebut kemudian dapat diberikan kepada *authorized user*.

d. Selanjutnya, *authorized user* membangkitkan dan memperoleh kunci.
e. Setelah membangkitkan kunci maka *authorized user* dapat melakukan permintaan untuk menghapus *log* atau mengambil *log* yang telah diberikan *user*.
f. Selanjutnya *log*ging *server* menunggu *request* dari *authorized user*. Apabila *request* dari *authorized user* berisi perintah pengambilan *log* maka *logging server* mengambil *log*, mendekripsi *log* tersebut dengan menggunakan kunci yang telah dimiliki oleh *authorized user,* mengekstraksi *watermark* dan memperoleh *log* yang diminta. Jika permintaan yang dilakukan yaitu menghapus, maka *logging server* akan menghapus *log* tersebut.
g. *Authorized user* memperoleh *log* dan dapat mengaksesnya.



Gambar 2. Skema SecLaaS-RW

3. METODE PENELITIAN

Metodologi penelitian yang dilakukan pada penelitian ini dilakukan berdasarkan metodologi *Parallel Waterfall Development*. Metodologi ini terdiri empat tahap yaitu *planning, analysis, design,* dan *implementation*. Selanjutnya, untuk memperoleh data dari penelitian ini maka digunakan metode kualitatif. Data tersebut diperoleh dari hasil pengujian *functional testing* dari implementasi skema SecLaaS-RW menjadi aplikasi.

3.1. *Planning* (Perencanaan)

Tahap Perencanaan merupakan proses yang fundamental dalam memahami mengapa aplikasi *secure logging* ini perlu dan bagaimana aplikasi *secure logging* ini dibuat. Pada tahap ini dilakukan identifikasi masalah, identifikasi peluang, studi kelayakan, dan rencana kerja.

3.2. *Analysis* (Analisis)

Tahap analisis merupakan proses untuk menjawab pertanyaan kebutuhan apa saja yang diperlukan system, baik secara fungsional maupun nonfungsional. Oleh karena itu pada tahapan analisis penelitian ini akan dianalisis secara fungsional dan nonfungsional yang diperlukan oleh aplikasi.

3.3. *Design* (Perancangan)

Tahap perancangan merupakan tahap untuk menjawab pertanyaan bagaimana gambaran aplikasi, cara kerja dan proses yang akan dikerjakan oleh aplikasi. Pada tahap perancangan penelitian ini dilakukan dengan tiga langkah yaitu perancangan arsitektur aplikasi, perancangan I, dan perancangan II.

3.4. *Implementation* (Implementasi)

Implementasi merupakan tahapan untuk menjawab bagaimana hasil dari pembuatan aplikasi dan apakah sudah sesuai dengan yang direncanakan atau tidak. Pada tahap implementasi penelitian ini terdiri dari dua tahap yaitu implementasi I dan implementasi II.

Setelah aplikasi selesai dibuat maka dilakukan uji coba. Uji coba yang dilakukan yaitu dengan menggunakan *functional testing*. Pendekatan yang dilakukan pada *functional testing* yaitu dengan menggunakan teknik *black box*.

4. HASIL DAN PEMBAHASAN

4.1. Analisis Kebutuhan Aplikasi (*Analysis*)

Pada penelitian ini, proses analisis dilakukan dengan menggunakan teknik analisis dokumen [10]. Kebutuhan yang akan dibangun dalam penelitian ini diolah dari skema SecLaaS-RW. Analisis kebutuhan aplikasi ditinjau dari kebutuhan fungsional dan nonfungsional.

Sebelum menentukan kebutuhan fungsional dan nonfungsional dari masing masing aplikasi *client* dan *server*, perlu dilakukan identifikasi dari setiap pihak yang terlibat pada skema SecLaaS-RW dan peran yang dapat dilakukan pada skema tersebut. Pada Tabel 1 merupakan daftar pihak yang terlibat beserta peran yang dapat dilakukan.

Tabel 1. Pihak dan Peran

No	Nama Pihak	Peran yang dilakukan
1	*User*	Membangkitkan, menyimpan dan mengirimkan pada *logging server/ cloud.*
		Mengambil *log* yang telah tersimpan pada *cloud.*
		Memberikan hak/ wewenang mengakses *log* kepada *authorized user*.
		Melihat daftar *log* yang telah tersimpan pada *logging server/ cloud*.

		Menghapus *log* yang telah tersimpan pada *logging server/ cloud*
2	*Authorized User*	Melihat daftar *log* yang telah tersimpan pada *logging server/ cloud*
		Mengambil *log* yang telah tersimpan pada *cloud*.
		Menghapus *log* yang telah tersimpan pada *logging server/ cloud*
3	Administrator	Mengelola pengguna *logging server*. Mendaftarkan akun baru dan menghapus akun.

a. Analisis Kebutuhan Fungsional Aplikasi *Client*

Kebutuhan fungsional aplikasi *client* secara spesifik dijabarkan pada Tabel 2. Kebutuhan tersebut dianalisis berdasarkan penelitian dari Khan, Yaqoob, Sarwar, Tahir, and Ahmed [10].

Tabel 2. Tabel Kebutuhan Fungsional *Client*

No	Kebutuhan Fungsional *Client*
1	Aplikasi harus dapat berkomunikasi dengan *server*.
2	Aplikasi harus dapat digunakan untuk meminta layanan *secure logging* dari *server* berupa penyimpanan *log* dan pengambilan *log*.
3	Aplikasi menyediakan fungsi halaman *login* untuk pengguna dan halaman menu layanan yang dapat diakses sesuai dengan masing masing peran.
4	Aplikasi harus dapat digunakan untuk meminta layanan manajemen pengguna dari *server* berupa penambahan akun baru dan penghapusan akun.
5	Aplikasi harus dapat digunakan untuk meminta layanan manajemen *log* dari *server* berupa pemberian akses *log* untuk pengguna lain, informasi *list log* yang tersimpan pada *server*, dan penghapusan *log*.

b. Analisis Kebutuhan Nonfungsional Aplikasi *Client*

Kebutuhan nonfungsional aplikasi *client* dijabarkan pada Tabel 3. Kebutuhan tersebut dianalisis berdasarkan penelitian dari Khan, Yaqoob, Sarwar, Tahir, and Ahmed [10].

Tabel 3. Kebutuhan Nonfungsional *Client*

No	Kebutuhan Nonfungsional
1	Aplikasi harus dapat dijalankan pada perangkat komputer.
2	Aplikasi dibangun dengan menggunakan Python3
3	Aplikasi dibangun menggunakan arsitektur *client-server*

c. Analisis Kebutuhan Fungsional Aplikasi Server

Kebutuhan fungsional aplikasi server secara spesifik dijabarkan pada Tabel 4. Kebutuhan tersebut dianalisis berdasarkan penelitian dari Khan, Yaqoob, Sarwar, Tahir, and Ahmed [10].

Tabel 4. Kebutuhan Fungsional Server

No	Kebutuhan Fungsional *Server*
1	Aplikasi harus dapat berkomunikasi dengan *client*.
2	Aplikasi memiliki fungsi *secure logging* berupa penyimpanan *log* dan pengambilan *log* secara aman
3	Aplikasi menyediakan fungsi untuk dapat mengidentifikasi *client* yang ingin *login*.
4	Aplikasi memiliki fungsi manajemen pengguna berupa pembuatan akun dan penghapusan akun
5	Aplikasi memiliki fungsi manajemen *log* berupa pemberian akses *log* dari satu *user* ke *user* lain, penghapusan *log*, dan pemberian informasi berupa *list log* yang ada pada *server*

d. Analisis Kebutuhan Nonfungsional Aplikasi *Server*

Kebutuhan nonfungsional *server* merupakan berbagai karakteristik dan kegunaan yang harus ada pada aplikasi *server*. Pada Tabel 5 dijabarkan kebutuhan nonfungsional aplikasi *server* berdasarkan penelitian Khan, Yaqoob, Sarwar, Tahir, and Ahmed [10].

Tabel 5. Kebutuhan Nonfungsional *Server*

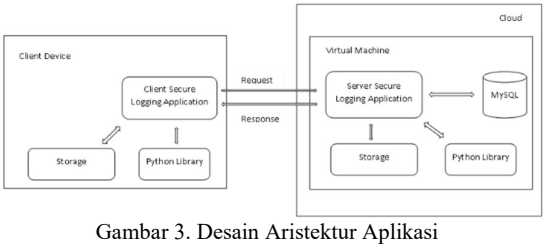
No	Kebutuhan Nonfungsional
1	Aplikasi harus dapat membedakan *user* "admin", *user* biasa atau *user* "*authorized user*" dan membatasi layanan antara *user* "admin", *user* biasa dan *user* "*authorized user*".
2	Aplikasi harus dapat dijalankan pada perangkat komputer.
3	Aplikasi dibangun dengan menggunakan Python3
4	Aplikasi dibangun dengan menggunakan teknologi virtualisasi atau *cloud*
5	Aplikasi harus dapat mendeteksi adannya perubahan pada *log* untuk memastikan integritas dan keotentikan *log*.
6	Aplikasi harus memiliki arsitektur *client-server*
7	Aplikasi mampu berkomunikasi secara multi-*client*

4.2. Perancangan Aplikasi (*Design*)

a. Perancangan Arsitektur

Perancangan arsitektur digunakan untuk menggambarkan bagaimana bentuk aplikasi yang akan dibuat. Gambar 3 merupakan rancangan arsitektur dari aplikasi yaitu arsitektur *client – server*. Dengan menggunakan arsitektur tersebut, maka aplikasi yang dibuat terletak pada *client* dan juga *server*.

Client device dapat berupa perangkat *server* atau komputer. *Log* yang ada pada *client* dapat dihasilkan dari berbagai sumber *log generator*. *Log generator* merupakan sistem atau aplikasi yang digunakan untuk membuat *log* atau rekaman transaksi. *Log generator* dapat berupa web *server*, *mail server*, tcpdump, NIDS/NIPS, aktivitas perangkat, aktivitas *login*, dan lain-lain. *Log* yang dihasilkan oleh sumber–sumber tersebut merupakan *log* yang akan dikirimkan ke *server* dengan menggunakan aplikasi *secure logging client*. Aplikasi *secure logging client* dibuat menggunakan bahasa pemrograman Python3. Proses dan fungsi–fungsi yang dijalankan oleh aplikasi *client* menggunakan *library* atau modul yang telah tersedia pada Python3. Aplikasi *client* juga memerlukan *storage* atau direktori tempat lokasi *log* yang ada pada *client*. Aplikasi *secure logging* akan mengakses lokasi tersebut jika ingin mengirimkan *log* ke *cloud* atau mengambil *log* dari *cloud*.

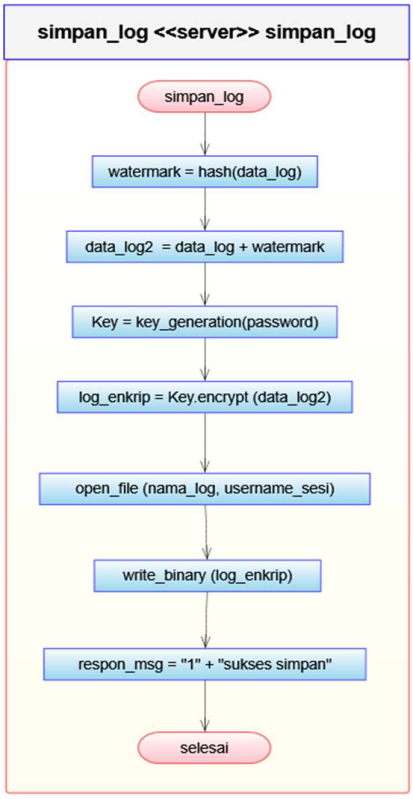


Gambar 3. Desain Aristektur Aplikasi

Logging Server merupakan *virtual machine* yang dijalankan di *cloud*. *Virtual machine* tersebut sebagai *environtment* aplikasi *secure logging server*. Aplikasi *secure logging* yang berada pada *server logging* ini bertidak sebagai *log collector*. Selain itu aplikasi *secure logging* yang berada pada *server* ini juga memiliki tugas untuk mengamankan *log* selama disimpan di *cloud*. Aplikasi secure *log*ging *server* dibuat menggunakan bahasa Python3. Proses dan fungsi–fungsi yang dijalankan oleh aplikasi *server* menggunakan *library* atau modul yang telah tersedia pada Python3. Aplikasi *server* juga memerlukan direktori (*storage*), yaitu tempat lokasi keseluruhan *log* yang ada pada *client*. Aplikasi *secure logging* akan mengakses lokasi tersebut jika ingin menyimpan *log* yang dikirimkan oleh *client* atau mengambil *log* dari *cloud* untuk dikirimkan pada *client*. Aplikasi *server* memerlukan aplikasi tambahan yaitu MYSQL yang digunakan untuk menyimpan keseluruhan data pengguna.

b. Perancangan *Reversible Watermarking*

Perancangan Pengiriman *Log* dan *Watermarking p*ada Gambar 4 merupakan rancangan yang menggambarkan proses ketika aplikasi pada *server* me*response request* untuk menyimpan *log*.

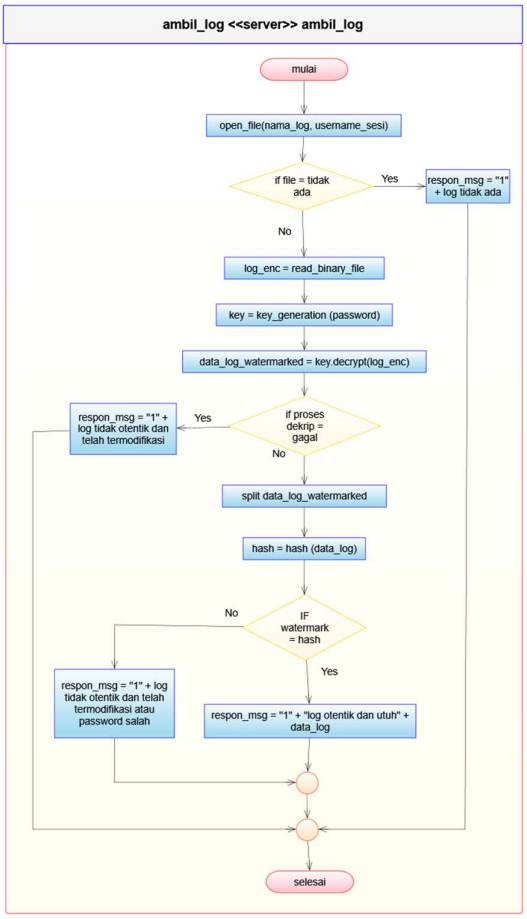


Gambar 4. Desain Proses Menyimpan *Log*

Pertama aplikasi membuat *watermark* menggunakan *hash* dari data *log*. Setelah itu, aplikasi menambahkan data *log* tersebut dengan *watermark* sehingga menghasilkan data *log* yang telah di-*watermark*. Selanjutnya, aplikasi membangkitkan kunci dengan menggunakan *password* seperti yang ada pada pesan *request*. Kunci tersebut digunakan untuk mengenkripsi data *log* yang telah di-*watermark*. Hasil enkripsi tersebut kemudian ditulis pada suatu *file* sesuai dengan nama *log* pada direktori sesi *username* pengguna aplikasi. Kemudian aplikasi membut pesan *response* untuk dikirimkan ke *client*.

Fungsi kirim *log* digunakan oleh aplikasi pada *server* untuk memberikan *log* yang diminta oleh *client*. Pada Gambar 5 merupakan rancangan yang menggambarkan fungsi ambil_*log*.

Pertama, aplikasi membuka *log file* sesuai dengan nama *log* di dalam direktori sesi *username*. Setelah itu, aplikasi membaca secara biner isi dari *log file* tersebut. Kemudian, aplikasi membangkitkan kunci dari *password* sesuai *request*. Kunci tersebut digunakan oleh aplikasi untuk mendekripsi data *log* terenkripsi sehingga diperoleh data *log* yang di*watermark*. Selanjutnya dilakukan pemisahan pada data *log* yang di-*watermark* sehingga diperoleh data *log* yang asli dan *watermark*. Jika *watermark* tidak dapat diambil maka *log* tersebut sudah mengalami modifikasi.



Gambar 5. Desain Proses Pengambilan *Log*

Hal tersebut juga dapat dilihat dengan

membandingkan antara *watermark* dan nilai *hash* dari data *log* yang berhasil diambil. Jika sama maka aplikasi akan membuat pesan *response* bernilai 1 dan bernotifikasi bahwa *log* otentik dan utuh bersamaan dengan data *log* itu. Apabila *watermark* tidak sama dengan nilai *hash* maka aplikasi akan membuat pesan *response* bernilai 1 dan memberikan notifikasi bahwa *log* tidak otentik dan telah dimodifikasi. Pesan *response* tersbut selanjutnya dikirim ke *client*.

4.3. Implementasi Aplikasi (*Implementation*)

Terdapat 12 fungsi yang diperlukan oleh aplikasi *client* dan dapat dilihat pada Tabel 6, sedangkan terdapat 18 fungsi yang diperlukan oleh aplikasi *server* yang dapat dilihat pada Tabel 7. Pembuatan fungsi-fungsi tersebut menggunakan bahasa Python3. Fungsi–fungsi tersebut diintegrasi menjadi kesatuan aplikasi pada tahap Implementasi II.

Tabel 6. Fungsi Aplikasi *Client*

No	Nama Program/ Fungsi	Kegunaan
1	Program Utama/ Main *Client*	Menjadi program utama untuk menjalankan fungsi – fungsi lain dan pembuatan jalur komunikasi.
2	*Setup_Connec*tion	Melakukan *setting* jalur komunikasi sehingga dapat terenkripsi dan aman.
3	*Log*in	Menyediakan *interface*/ halaman *login* kepada pengguna.
4	*Request*_Layanan	Membuat pesan *request* dan menentukan layanan apa yang dapat diperoleh pengguna.
5	*Request*_Layanan_ *User*_Biasa	Menyediakan menu layanan yang dapat diperoleh *user* biasa dan membuat pesan *request*.
6	*Request*_Layanan_ *Authorized_User*	Menyediakan menu layanan yang dapat diperoleh *authorized user* dan membuat pesan *request.*
7	*Request*_Layanan_ Admin	Menyediakan menu layanan yang dapat diperoleh admin dan membuat pesan *request*.
8	Kirim_*Request*	Mengirim pesan *request* kepada *server*.
9	Terima_respon	Menerima respon dari *server* dan menampilkan notifikasi.
10	*Hash*_data	Membangkitkan nilai *hash* berukuran 256 byte.
11	*Padding*Byte	Mem*padding* data menjadi kelipatan 16 byte.
12	Remove*Padding*Byte	Menghapus data *padding*.

Proses implementasi II dilakukan dengan memperhatikan rancangan arsitektur aplikasi. Sesuai dengan arsitektur aplikasi yang telah dirancang, maka proses implementasi II dilakukan dengan membuat dua buah aplikasi yaitu aplikasi *secure logging client* dan aplikasi *secure logging server*. Proses pembuatan masing masing aplikasi *secure logging client* dan aplikasi *secure logging server* dilakukan dengan mengintegrasikan fungsi–fungsi yang telah dibuat pada tahap implementasi I.

Implementasi II menghasilkan empat *file* aplikasi, yaitu *secure_logging_client*.py pada sisi *client*, *secure_logging_server.py* pada sisi *server*, *file* myapp.*log* pada sisi *server*, dan *file*_config.py pada sisi *server*. *File secure_logging_client*.py merupakan aplikasi yang digunakan oleh *client* untuk dapat berinteraksi dengan *server* termasuk meminta dan mendapatkan layanan dari *server*. *File* tersebut harus diletakan di perangkat komputer *client* dan dijalankan dengan menggunakan Python3. *File* tersebut dijalankan pada umumnya oleh saeorang administrator. *File file*_config.py merupakan *file* konfigurasi supaya aplikasi *server* dapat terhubung dengan MYSQL. *File* myapp.*log* merupakan *log file* yang berisi catatan aktivitas yang dilakukan aplikasi *server* termasuk mencatat ketika terdapat *error*.

Tabel 7. Fungsi Aplikasi Server

No	Nama Program/ Fungsi	Kegunaan
1	Program Utama/ Main *Server*	Menjadi program utama untuk menjalankan fungsi – fungsi lain dan pembuatan jalur komunikasi.
2	*Setup_Connec*tion	Melakukan *setting* jalur komunikasi sehingga dapat terenkripsi dan aman.
3	Identifikasi_*Client*	Menerima parameter *user*name dan password dari *client* kemudian melakukan identifikasi.
4	Terima_*Request*	Menerima pesan *request* dari *client*.
5	Respon_Layanan	Merespon *request* dari *client.*
6	Simpan_*log*	Menyimpan *log*.
7	Ambil_*log*	Mengambil *log*.
8	Hapus_*log*	Menghapus *log*.
9	Copy_*log*	Memberikan asks *log* dari suatu usr ke *user* lain.
10	Tambah_akun	Menambahkan akun baru.
11	Hapus_akun	Menghapus akun.
12	*Setup_database_ connec*tion	Setting koneksi *database*.
13	Database_*connect*	Mengkoneksikan / menghubungkan pada *database*.
14	Get_ip_address	Memperoleh *ip address* secara otomatis dari masukan *interface* jaringan.
15	Remove*Padding* Byte	Menghapus *padding*.
16	*Padding*Byte	Mem*padding* data menjadi kelipatan 16 byte.
17	*Hash*_data	Membangkitkan nilai *hash* berukuran 256 byte.
18	Buat_*log*	Membuat *log* untuk mencatat aktivitas pada aplikasi *server*.

Implementasi II menghasilkan empat *file* aplikasi, yaitu *secure_logging_client*.py pada sisi *client*, *secure_logging_server.py* pada sisi *server*, *file* myapp.*log* pada sisi *server*, dan *file*_config.py pada

sisi *server*. *File secure_logging_client*.py merupakan aplikasi yang digunakan oleh *client* untuk dapat berinteraksi dengan *server* termasuk meminta dan mendapatkan layanan dari *server*. *File* tersebut harus diletakan di perangkat komputer *client* dan dijalankan dengan menggunakan Python3. *File* tersebut dijalankan pada umumnya oleh saeorang administrator. *File file*_config.py merupakan *file* konfigurasi supaya aplikasi *server* dapat terhubung dengan MYSQL. *File* myapp.*log* merupakan *log file* yang berisi catatan aktivitas yang dilakukan aplikasi *server* termasuk mencatat ketika terdapat *error*.

Terdapat juga hal yang perlu diperhatikan pada proses implementasi II yaitu kunci rahasia dan kunci publik baik itu milik *client* atau *server* harus diletakan pada satu direktori dengan aplikasi *secure logging*. Ukuran kunci yang digunakan harus dibangkitkan sesuai dengan algoritme RSA 2048 bit supaya aplikasi tersebut dapat berjalan dengan baik.

Tampilan menu aplikasi secure *log*ging pada sisi client menampilkan sesuai dengan level *user* masing masing. Tampilan menu aplikasi dengan level administrator, *user* biasa, dan *authorized user* dapat dilihat pada Gambar 7, Gambar 8, dan Gambar 9. Administrator hanya dapat menggunakan aplikasi *secure logging* untuk membuat *user* baru atau menghapus *user* lama. *User* biasa dapat menggunakan aplikasi untuk menyimpan *log*, mengambil *log*, menghapus *log*, melihat daftar *log* yang tersimpan, dan memberikan akses *log* tersebut kepada *authorized user*. *Authorized user* dapat menggunakan aplikasi *secure logging* untuk melihat daftar *log* yang tersimpan pada *cloud*, menghapus *log*, dan mengambil *log*.

```
[>] ENTER YOUR NAME : aray

[>] ENTER YOUT PASSWORD: Aray1234
Anda Sukses login sebagai aray

Hello.... aray
Masukan pilihan layanan...
1. Daftarkan Akun baru
2. Hapus Akun
3. Exit
->
```

Gambar 6. Tampilan Menu Administrator

```
[>] ENTER YOUT PASSWORD: Dino1234
Anda Sukses login sebagai dino

Hello.... dino
Masukan pilihan layanan...
1. Simpan log
2. Ambil log
3. Tampilkan list log yang telah tersimpan
4. Hapus log
5. Memberi Akses Log ke User lain
6. Exit
```

Gambar 7. Tampilan Menu *User* Biasa

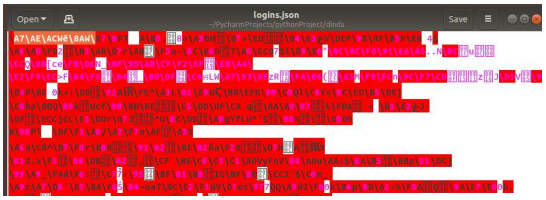
```
[>] ENTER YOUR NAME : dinda

[>] ENTER YOUT PASSWORD: Dinda1234
Anda Sukses login sebagai dinda

Hello.... dinda
Masukan pilihan layanan...
1. Ambil log
2. Tampilkan list log yang telah tersimpan
3. Hapus log
4. Exit
->
```

Gambar 8. Tampilan Menu *Authorized User*

Tampilan proses penyimpanan *log* dapat dilihat pada Gambar 9. Pada saat penyimpanan, *user* perlu memberikan input berupa nama *log file* yang ingin disimpan dan *password*. Ketika sudah disimpan pada *cloud*, *log* tersebut sudah tidak dapat dibaca lagi kecuali oleh *user* yang memiliki akses. Gambar 10 memperlihatkan tampilan *log* yang sudah tersimpan di *cloud* dalam kondisi sudah terenkripsi.

```
Hello.... dino
Masukan pilihan layanan...
1. Simpan log
2. Ambil log
3. Tampilkan list log yang telah tersimpan
4. Hapus log
5. Memberi Akses Log ke User lain
6. Exit
->1
masukan nama log yang ingin disimpan di central log?
->PetTrainSet2.csv
masukan password untuk perlindungan file anda?
->mantap

[!] file berhasil dikirim dan tersimpan
```

Gambar 9. Proses Penyimpanan *Log*



Gambar 10. Tampilan *log* yang telah tersimpan

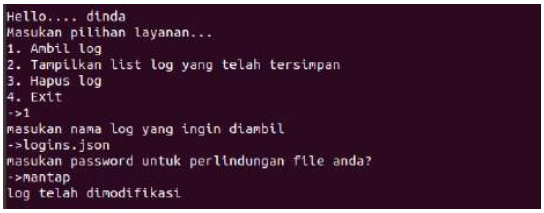
Tampilan proses pengambilan *log* dapat dilihat pada Gambar 11. Adapun Ketika *log* tersebut tidak terdapat modifikasi atau perusakan maka tampil pesan bahwa *log* dalam keadaan utuh dan otentik.

```
[!] SERVER SAID : OK ini log anda, log berhasil diambil dengan kondisi masih utuh dan otentik

Hello.... dinda
Masukan pilihan layanan...
1. Ambil log
2. Tampilkan list log yang telah tersimpan
3. Hapus log
4. Exit
->1
masukan nama log yang ingin diambil
->auth.log
masukan password untuk perlindungan file anda?
->mantap

[!] SERVER SAID : OK ini log anda, log berhasil diambil dengan kondisi masih utuh dan otentik
```

Gambar 11. Proses pengambilan *log* yang masih utuh

Apabila sebelum proses pengambilan *log* terjadi modifikasi data berupa penghapusan beberapa data yang dilakukan seperti contoh pada Gambar 12, maka saat pengambilan *log* akan tampil pesan bahwa *log* sudah mengalami perubahan dan tidak utuh. Pesan bahwa *log* telah mengalami perubahan data *log* dapat dilihat pada Gambar 13.

Gambar 12. Modifikasi data pada bagian yang di*block*



Gambar 13. Proses pengambilan *log* yang sudah tidak utuh

4.4. Pengujian Aplikasi (*Testing*)

Setelah proses implementasi II, maka selanjutnya dilakukan proses pengujian aplikasi. Pengujian dilakukan dengan menggunakan teknik *functional testing*. *Functional testing* merupakan pengujian yang dilakukan untuk memvalidasi dan menguji fungsionalitas dari aplikasi yang telah dibuat apakah sudah memenuhi kebutuhan fungsional aplikasi. Pendekatan yang dilakukan pada pengujian aplikasi pada penelitian ini yaitu menggunakan *black box testing*. Dengan menggunakan pendekatan *black box testing* maka proses pengujian fungsionalitas aplikasi dilakukan tanpa mengetahui struktur kode dan detail *source code* dari aplikasi. Fokus utama yang dilihat pada pengujian ini yaitu melihat *input* dan *output* dari aplikasi dengan didasari pada kebutuhan fungsional aplikasi.

Sebelum dilakukan pengujian aplikasi, maka perlu dilakukan persiapan *environtment* atau lingkungan pengujian. Lingkungan pengujian pada penelitian ini dijabarkan pada Tabel 8.

Sistem operasi yang digunakan pada *client* yaitu Windows 10, Ubuntu 18.04, dan MacOS High Sierra sedangkan pada *server* menggunakan Ubuntu 18.04. Bahasa pemrograman yang digunakan pada setiap aplikasi yaitu Python3 sehingga pada masing - masing *client* perlu diinstal terlebih dahulu Python3. Aplikasi *client* tidak memerlukan aplikasi tambahan, sedangkan pada aplikasi *server* memerlukan Mysql dan lingkungan *Cloud*. Pada penelitian ini lingkungan *cloud* disimulasikan dengan menggunakan VirtualBox. Hal tersebut dikarenakan konsep dari teknologi virtualisasi pada virtualbox dapat mewakili konsep *cloud*. File tambahan yang dibutuhkan masing masing aplikasi yaitu rahasia.txt yang berisi kunci rahasia dan publik.txt yang berisi kunci publik. Pasangan kunci tersebut dibangkitkan dengan menggunakan algoritme RSA 2048 bit.

Setelah proses persiapan lingkungan aplikasi dilakukan, maka dilanjutkan dengan proses *functional testing*. Pengujian tersebut menghasilkan bahwa aplikasi yang telah dibuat telah sesuai dengan yang direncakanan. Fungionalitas aplikasi telah memenuhi kebutuhan fungsional yaitu *secure logging*, manajemen *log*, dan manajemen pengguna. Dari 10 jenis *test cases*, 21 macam *output*, dan 21 macam *output* yang diharapkan, dihasilkan 21 macam *output* yang sesuai dengan *output* yang diharapkan.

Tabel 8. Lingkungan Sistem

No	Nama Lingkungan	*Client*	*Server*
1	Sistem Operasi	Windows/ Windows 10, 8inux/ Ubuntu 18.04, dan MacOS/ MacOS High Sierra.	Linux/ Ubuntu 18.04.
2	Bahasa Pemrograman	Python 3	Python 3
3	Modul/ *Sistem*	Pycryptodome, Lazyme, Strings, OS, Signal, Threading, Socket, dan *hash*lib.	Pycryptodome, Lazyme, Socket, Threading, Base64, *hash*lib, Fernet, Netifaces, OS, dan MYSQL.*Connecto* r.
4	Aplikasi Tambahan	-	Mysql, VirtualBox (*Cloud*).
5	*File* tambahan	Rahasiae.txt (berisi kunci rahasia *client*) dan publik.txt (berisi kunci publik *client*) yang dibangkitkan dengan menggunakan algoritme RSA 2048 bit.	Rahasiae.txt (berisi kunci rahasia *server*) dan publik.txt (berisi kunci publik *server*) yang dibangkitkan dengan menggunakan algoritme RSA 2048 bit.

Implementasi skema SecLaaS-RW menjadi aplikasi *secure logging* berhasil dibuat dengan menggunakan bahasa Python3 dan modul (*library*) yang telah ada pada Python3. Aplikasi berhasil dipasang pada lingkungan *cloud* dan dapat dijalankan. *Cloud* yang digunakan pada implementasi ini yaitu virtualbox.

Aplikasi dapat memenuhi beberapa properti *secure logging*. Aplikasi dapat memastikan keutuhan dan keotentikan *log* dengan memanfaatkan teknik *reversible watermarking* yang ada pada Skema SecLaaS-RW. Selain itu, kerahasiaan data yang ada pada *log* dapat terjaga mulai dari pengiriman hingga penyimpanan dengan menggunakan teknik enkripsi sesuai dengan skema SecLaaS-RW. Aplikasi yang dibuat dapat memastikan kebenaran format *log* yang disimpan karena dalam proses penyimpanan dan pengambilan *log*, format *log* tidak mengalami perubahan. Adapun format *log* tidak terbatas pada satu format saja melainkan pada beberapa format yang pada penelitian ini diguanakn format .*log*, .json, .pcap, dan .csv. Aplikasi ini juga dapat memastikan bahwa hanya *user* yang berhak mengakses saja yang dapat mengakses *log* mulai dari penyimpanan sampai pada pengambilan *log*.

Aplikasi yang telah dibuat dapat dijalankan di berbagai sistem operasi yaitu Windows, Linux, dan MacOS. Selain itu, aplikasi mendukung komunikasi multi-*client*, sehingga satu aplikasi *secure logging server* dapat melayani lebih dari satu *client* secara bersamaan.

5. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat diperoleh kesimpulan sebagai berikut:

1. Implementasi skema SecLaaS-RW menjadi aplikasi *secure logging* berhasil dilakukan sesuai dengan yang telah direncanakan.
2. Implementasi skema SecLaaS-RW berhasil dilakukan dengan menggunakan Python3 beserta modul yang telah tersedia pada Python3. Skema SecLaaS-RW diimplementasi menjadi dua aplikasi yaitu aplikasi *server* dan aplikasi *client*. Aplikasi yang telah dibuat memiliki fungsi *secure logging*, manajemen *log*, dan manajemen pengguna. Aplikasi yang telah dibuat memenuhi properties *secure logging* yaitu memastikan keutuhan dan keotentikan *log*, memastikan kebenaran format *log*, memastikan hanya pengguna yang terotorisasi saja yang dapat mengakses *log*, dan memastikan kerahasiaan data *log*.

REFERENSI

[1] K. Kent and M. Souppaya, "Guide to Computer Security *Log* Management," *Nist Special Publication*, 2006.

[2] V. Raval, "Challenges of Security *Log* Management," *Isaca Journal*, vol. 6, pp. 1–5, 2017.

[3] C. Lovick, "The BSD sys*log* Protocol," *Network Working Group*, p. 29, 2001.

[4] B. Schneier and J. Kelsey, "Secure Audit *Log*s to Support Computer Forensics," *ACM Transactions on Information and System Security*, vol. 2, no. 2, pp. 159–176, 1999, doi: 10.1145/317087.317089.

[5] D. Ma and G. Tsudik, "A New Approach to Secure *Log*ging".

[6] J. Holt, "*Log*crypt: forward security and public verification for secure audit *log*s," *Australasian workshops on Grid computing and e-research*, vol. 54, pp. 203–211, 2006.

[7] A. A. Yavuz and P. Ning, "BAF: An efficient publicly verifiable secure audit *log*ging scheme for distributed systems," *Proceedings - Annual Computer Security Applications Conference, ACSAC*, no. ii, pp. 219–228, 2009, doi: 10.1109/ACSAC.2009.28.

[8] I. Ray, K. Belyaev, M. Strizhov, D. Mulamba, and M. Rajaram, "Secure *log*ging as a service-delegating *log* management to the cloud," *IEEE Systems Journal*, vol. 7, no. 2, pp. 323–334, 2013, doi: 10.1109/JSYST.2012.2221958.

[9] S. Zawoad, A. K. Dutta, and R. Hasan, "Towards Building Forensics Enabled Cloud Through Secure *Log*ging-as-a-Service," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 148–162, 2016, doi: 10.1109/TDSC.2015.2482484.

[10] A. Khan, A. Yaqoob, K. Sarwar, M. Tahir, and M. Ahmed, "Secure *Log*ging as a Service Using *Reversible Watermarking*," *Procedia Computer Science*, vol. 110, pp. 336–343, 2017, doi: 10.1016/j.procs.2017.06.103.

[11] H. mani and S. Singh, "A Survey of Digital *Watermarking* Techniques and Performance Evaluation Metrics," *International Journal of Engineering Trends and Technology*, vol. 46, no. 2, pp. 128–132, 2017, doi: 10.14445/22315381/ijett-v46p220.