VERIFIKASI *ALERT* BERDASARKAN KLASIFIKASI SERANGAN PADA DETEKSI INTRUSI KOLABORATIF

Riffin Sukmana Hakim, Idris Winarno,S.ST, M.kom, Wiratmoko Yuwono, ST
Program D IV Jurusan Teknik Informatika
Politeknik Elektronika Negeri Surabaya-Institut Teknologi Sepuluh Nopember
Kampus ITS Keputih Sukolilo Surabaya 60111
Tel: (+62)31-5910040 Fax: (+62)31-5910040
E-mail: Riffin.sukmana@gmail.com

Makalah Penelitian

ABSTRAK

Sistem deteksi intrusi telah lama digunakan untuk mendeteksi adanya serangan-serangan yang masuk ke suatu system jaringan maupun computer itu sendiri. System deteksi intrusi selama ini mendeteksi serangan yang masuk ke dalam system (baik jaringan maupun computer) berdasarkan aktifitas yang terjadi pada system. Setiap aktifitas yang dianggap berbahaya yang tercatat akan disimpan di dalam database system deteksi intrusi.

Dalam praktiknya, system deteksi intrusi selama ini kadang melakukan pendeteksian yang kurang akurat. Sistem deteksi intrusi mendeteksi aktifitas yang tidak berbahaya *(false positive)* atau tidak mendeteksi serangan yang berbahaya (*false negative*). Ketika jumlah alert false positive ini berjumlah sangat banyak, bahkan terjadi alert flood, maka ini akan mempersulit analisa kondisi serangan itu sendiri.

Dengan memanfaatkan kolaborasi dan korelasi sistem deteksi intrusi, kemudian melakukan korelasi alert, maka akan mereduksi kelemahan sistem deteksi intrusi (*false positive* dan *false negative*), dan menghasilkan alert yang lebih baik dan akurat.

Kata kunci : sistem deteksi intrusi, alert, false positive, korelasi alert, kolaborasi alert.

1. PENDAHULUAN

1.1 Latar Belakang

Semakin meningkatnya kejahatan yang terjadi dalam bidang teknologi informasi, maka kebutuhan keamanan jaringan menjadi suatu bagian yang penting sekali. Pada beberapa tahun ini telah dikembangkan tool-tool yang mendukung keamanan jaringan komputer. Salah satu tool yang dikembangkan untuk keamanan jaringan komputer adalah sistem deteksi intrusi.

Sistem deteksi intrusi digunakan untuk melindungi kemanan jaringan komputer dengan cara memonitoring dan mendeteksi serangan yang terjadi pada jaringan serta memberikan *alert* dan keterangan jenis serangan yang terjadi. Tetapi dalam implementasinya, sistem deteksi intrusi kadangkala memberi *alert* yang salah (*false positive*) atau menampilkan banyak sekali alert (*alert flood*). Sehingga, user sulit untuk menganalisa serangan dan melakukan tindakan yang tepat.

Untuk menyelesaikan permasalahan keterbatasan sistem deteksi intrusi tersebut, maka dapat dilakukan kolaborasi dan korelasi dengan deteksi intrusi yang lain. Kolaborasi dengan sistem deteksi intrusi yang lain dilakukan agar sistem deteksi intrusi mendeteksi serangan dengan lebih mendalam, mengamati dan mencegah resiko keamanan yang potensial. Sedangkan korelasi dilakukan untuk menghasilkan analisa serangan yang lebih mendalam dan mengurangi alert yang terlalu banyak (*alert flood*) yang disebabkan oleh kolaborasi dengan sistem deteksi intrusi yang lain.

Pada tugas akhir akan dirancang suatu sistem deteksi intrusi baru yang merupakan gabungan dari dua sistem deteksi intrusi, untuk menghasilkan analisa terhadap suatu serangan yang lebih mendalam dan lebih efisien

1.2 Rumusan Permasalahan

Berdasarkan uraian diatas, maka permasalahan yang timbul berkaitan dengan pengerjaan proyek akhir ini adalah :

1. Mengkolaborasikan dua sistem deteksi intrusi untuk menghasilkan alert baru yang merupakan gabungan alert sebelumnya.
2. Melakukan korelasi sistem deteksi intrusi untuk menghasilkan analisa terhadap serangan agar lebih akurat.

1.3 Tujuan

Tujuan dari proyek akhir ini adalah membangun kolaborasi sistem deteksi intrusi yang diharapkan mampu untuk :

1) Mampu menekan permasalahan-permasalahan yang selama ini terjadi dalam sistem deteksi intrusi.
2) Memberikan analisa terhadap serangan secara lebih akurat.

1.4 Batasan Permasalahan

Pada proyek akhir ini permasalahan difokuskan pada permasalahan seperti :

1. Alert yang digunakan adalah intrusi yang telah terdeteksi oleh sistem deteksi intrusi yang digunakan sebagai pendeteksi.
2. Jumlah sistem deteksi intrusi yang digunakan adalah dua sistem deteksi intrusi, salah satunya adalah sistem deteksi intrusi network-based, sedangkan yang lain menggunakan sistem deteksi intrusi host-based.
3. Kedua sistem deteksi intrusi menggunakan di dalam satu komputer.
4. Tahap korelasi multistep tidak digunakan.

2. TINJAUAN PUSTAKA

2.1 Sistem Deteksi Intrusi

2.1.1Deskripsi

Sistem deteksi intrusi mendapatkan informasi dari suatu sistem informasi untuk menjalankan diagnosa pada status keaamanan. Tujuannya adalah mengamati perusakan terhadap keamanan, usaha-usaha perusakan sistem, atau kelemahan terbuka yang berpotensi menyebabkan gangguan keamanan.

2.1.2 Jenis Sistem Deteksi Intrusi

Sistem deteksi intrusi, jika dari letak sensornya maka terbagi menjadi 3, yaitu Sistem Deteksi Intrusi *Host Based*, Sistem Deteksi Instrusi *Network Based* dan sistem deteksi intrusi *application based*:

1. Sistem Deteksi *Network Based*
   Sistem deteksi intrusi berdasar network adalah sistem intrusi deteksi yang mendeteksi intrusi melalui lalu lintas paket dalam jaringan. Sumber informasi yang digunakan oleh sistem deteksi intrusi adalah paket jaringan dan informasi SNMP.
2. Sistem Deteksi Host Based
   Sistem deteksi intrusi ini berdasarkan host, melakukan pemeriksaan data yang dihasilkan oleh sistem operasi. Sensor jenis ini melakukan deteksi log dari system call yang telah dieksekusi. Beberapa IDS menggunakan isi dari harddisk sebagai inputnya.
3. Sistem Deteksi Intrusi Application Based
   Sistem deteksi intrusi application based mengambil dari log suatu aplikasi. Jenis sensor ini digunakan untuk melindungi aplikasi jaringan.

2.13 Jenis *Alert* pada sistem deteksi intrusi

Alert pada sistem deteksi memiliki tingkatan tertentu untuk menunjukkan kevalidan suatu serangan yang ditujukan kepada suatu sistem jaringan komputer. Tingkatan-tingkatan tersebut adalah :

a) **True positif :** Suatu kondisi dimana sistem deteksi intrusi mendeteksi adanya serangan (mengeluarkan *alert*) dan serangan tersebut benar-benar terjadi.
b) **True negatif :** Suatu kondisi dimana sistem deteksi intrusi tidak mendeteksi adanya suatu serangan(tidak mengeluarkan *alert*), tetapi terjadi serangan terhadap sistem jaringan komputer.
c) **False positif :** Suatu kondisi dimana sistem deteksi intrusi mendeteksi suatu serangan, tetapi sebenarnya serangan tersebut bukan suatu tindakan yang mencurigakan atau merusak
d) **False negatif :** Suatu kondisi dimana sistem deteksi intrusi tidak mendeteksi adanya serangan, dan tidak ada serangan yang terjadi

2.1.4 Permasalahan Sistem Deteksi Intrusi :

2.1.4.1 False Positive

Sistem deteksi intrusi didesain untuk mencari aktifitas jaringan sebagai bukti penyalahgunaan fasilitas. Ketika algoritma sistem deteksi intrusi mendeteksi adanya aktifitas yang tidak berbahaya dan mencurigakan, hal ini disebut sebagai false positive. *False positive* terjadi bukan karena algoritma yang tidak tepat, tetapi *false positive* terjadi karena kurang lengkapnya algoritma yang digunakan oleh sistem deteksi intrusi.

Terjadinya false postive memberikan banyak pengaruh, cara penyebaran dan operasi dalam jaringan modern :

- Ketika sistem deteksi intrusi menggunakan konfigurasi yang mengijinkan untuk mendeteksi false positive, maka sistem deteksi intrusi akan mendeteksi banyak sekali *alert* dan logs yang akan memenuhi space dari disk, membuat database lambat dan mempersulit proses *alert* karena sudah terlalu banyak *alert* yang muncul.

- Karena terlalu banyak nya *false positive* yang muncul, maka administrator harus melakukan analisa pada setiap event yang tersimpan dalam *alert* dan logs.

- Munculnya *false positive* yang terlalu banyak, akan menyebabkan *real-time* yang dimiliki sistem deteksi intrusi untuk melakukan sistem respon secara otomatis, seperti konfigurasi firewall menjadi tidak digunakan. Jika sistem response tersebut digunakan tentu akan filtering yang sangat besar sekali.

2.1.4.2 Kurangnya Keterangan :

Sistem deteksi intrusi kadang memberikan keterangan yang kurang lengkap sehingga administrator tidak memiliki informasi yang cukup untuk melakukan tindakan. Kurangnya keterangan ini disebabkan sensor dari sistem deteksi intrusi hanya satu wilayah saja. Sebagai contoh, pada sistem deteksi intrusi berbasis host (*host based IDS*), jika sensor menangkap adanya *alert* yang muncuk, maka *alert* tersebut tidak menyertakan IP penyerang, sedangkan pada sistem deteksi intrusi berbasis jaringan (*network based IDS*), yang tercatat dalam *alert* adalah trafic paket yang terdapat pada jaringan, tanpa mencatat PID yang mencurigakan atau perubahan yang terjadi pada host.

2.1.4.3 Non-contextual *Alert*

Non-contextual/non-relevant positif *alert* adalah *alert* yang direkam oleh sensor, tetapi *alert* tersebut berasal dari serangan yang gagal mencapai tujuannya. Nonrelevant positif *alert* jika dilakukan berulang-ulang terkadang dimanfaatkan untuk melihat kebenaran dari parameter serangan atau dimanfaatkan sebagai serangan *buffer overflow*.

Tingkat kerawanan kecelakaan lalu lintas suatu jalan dilihat dari data angka kecelakaan dijalan tersebut.

2.2. Kolaborasi Sistem Deteksi Intrusi

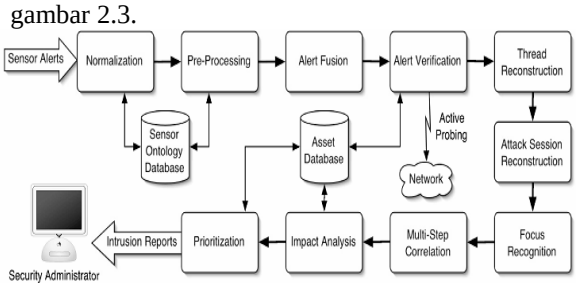
Sistem deteksi intrusi kolaborasi bertujuan untuk membuat deteksi intrusi menjadi lebih akurat, lebih efisien dan lebih dapat diatur dengan cara menggabungkan informasi dari beberapa sistem deteksi intrusi, jaringan dan mekanisme keamanan jaringan yang lain.

2.2 Korelasi Sistem Deteksi Intrusi

Korelasi *alert* adalah proses untuk menganalisa *alert* yang dihasilkan oleh satu atau beberapa sistem deteksi intrusi dan memberikan gambaran jelas dari kejadian atau usaha-usaha intrusi. Proses ini terdiri dari komponen-komponen yang merubah *alert* sensor deteksi intrusi menjadi laporan (*report*) intrusi. Karena *alert* dapat merujuk ke bermacam-macam serangan pada level *granurality* yang berbeda, proses korelasi tidak bisa mencakup semua *alert* dengan cara yang sama, sehingga harus dibuat komponen-komponen yang fokus pada aspek yang berbeda dari korelasi secara keseluruhan.

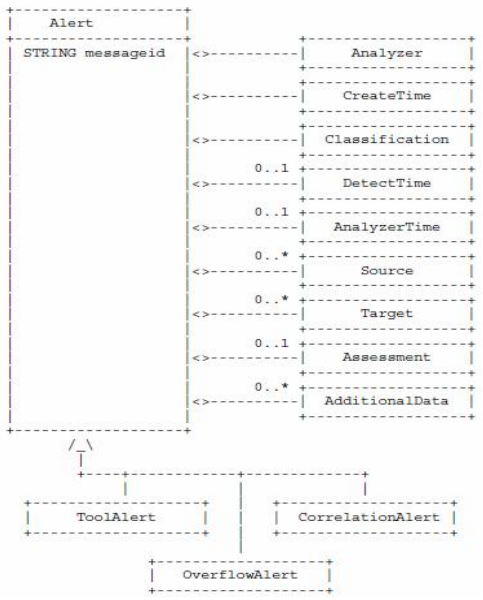
2.3.1 Proses Korelasi Sistem Deteksi Intrusi

Tahapan dari korelasi sistem deteksi intrusi dapat dilihat pada gambar 2.3. terdapat sepuluh tahap untuk menyelesaikan proses korelasi sistem deteksi intrusi. Berikut adalah penjelasan dari gambar 2.3.



Gambar 2.3. Gambaran umum proses korelasi

Pada awal fase pertama, komponen *normalization* menerjemahkan setiap *alert* yang diterima menjadi format standart yang dipahami semua komponen. Untuk penyamaan format atribut alert untuk normalisasi ini menggunakan format tertentu, salah satunya adalah IDMEF. Hal ini harus dilakukan karena *alert* dari sensor yang berbeda dikodekan dengan cara yang berbeda pula. Pada gambar 2.4 adalah data model dari kelas alert yang diberikan oleh IDMEF.



Gambar 2.4 kelas *alert* IDMEF

Kemudian, komponen *preprocessing* menambahkan *alert* yang telah dinormalisasi menjadi *alert* yang memiliki nilai.

Selanjutnya, *komponen fusion* menggabungkan *alert* yang sama yang muncul pada sistem deteksi intrusi yang berbeda. Tugas dari komponen *verification* adalah mengambil sebuah *alert* yang terjadi kemudian memastikan keberhasilan serangan yang cocok dengan dengan serangan yang baru saja tercatat dalam *alert*. Jika terdapat *alert* yang cocok dengan serangan yang gagal, maka *alert* tersebut akan ditandai dan pengaruhnya dengan korelasi akan dihapuskan.

Pada proses *verification* inilah dilakukan dengan memanfaatkan klasifikasi serangan. Penetuan keberhasilan suatu serangan dilakukan dengan cara memberikan klasifikasi tingkat serangan yang terjadi. Suatu serangan diberikan tingkatan masing-masing. Ketika serangan mencapai sistem operasi, service, atau koneksi, seringkali menunjukkan kebutuhan awal untuk mencapai serangan yang berhasil. Sedangkan pada serangan yang mencapai user, proses, dan file sistem, seringkali menunjukkan sebagai salah satu dampak dari serangan yang berhasil.

Tugas dari komponen *thread reconstruction* adalah menggabungkan deretan *alert* yang berkaitan dengan serangan yang dilancarkan oleh seorang penyerang terhadap suatu target. Kemudian, *attack reconstruction* menghubungkan *alert* yang terdapat pada sistem deteksi intrusi berdasar jaringan (*network based*) dengan *alert* pada berbasis host (*host based*) yang berelasi dengan serangan yang sama.

Komponen *focus recognition* bertugas mengidentifikasi host (source ataupun target) dari jumlah serangan yang banyak. Ini digunakan untuk mendeteksi serangan *denial of service* (DOS) atau *port scanning*. Komponen *multistep correlation* bertugas mengidentifikasi pola serangan seperti serangan *island-hopping*. Pola serangan ini terdiri atas serangkaian serangan individual, yang mana terjadi pada titik-titik yang berbeda jaringan.

Komponen-komponen terakhir dari proses korelasi adalah *impact analysis* dan *prioritization*. Komponen *Impact analysis* menentukan pengaruh dari serangan yang terdeteksi pada operasi dari jaringan yang dimonitori dan menjadi target dari aktifitas yang mencurigakan. Dengan menggunakan analisa ini, komponen *prioritization* memberikan prioritas yang tepat untuk setiap *alert*. Keterangan tentang prioritas ini menjadi penting untuk menghapus informasi yang tidak sesuai. Untuk metode prioritas yang digunakan untuk sistem, maka disesuaikan dengan 5 *severity level* yang dikeluarkan oleh CERT dan evaluasi dari sistem deteksi intrusi Snort. berikut adalah tabel dari evaluasi dari *attack severity numerical* :

Level serangan	**Kelas serangan**
1	Ambil alih super user
2	Ambil alih user
3	Denial of service
4	Perusakan integritas informasi
	Perusakan informasi atau sistem resource
5	Eksekusi malicious code
	Perusakan security policy

Table 2.1 tabel attack severity numerical

2.1 Klasifikasi Serangan

Klasifikasi serangan dilakukan untuk menunjukkan tingkat serangan yang dilakukan oleh hacker. Klasifikasi serangan dilakukan dengan cara membagi setiap serangan sesuai dengan resource jaringan komputer yang diserang. Contoh : Sistem operasi, Service, File sistem, user dan proses.

3. PERANCANGAN DAN IMPLEMENTASI

3.1 Gambaran Umum

3.1.1 Data Alert Sistem Deteksi Intrusi Sebagai Data Awal

Pada bab sebelumnya telah disebutkan bahwa sistem deteksi intrusi akan mendeteksi aktivitas yang mencurigakan yang terdapat pada jaringan untuk sistem deteksi intrusi *network-based* dan mengecek file sistem yang mengalami perubahan untuk sistem deteksi intrusi *host-based*. Alert dari

sistem deteksi tersebut akan disimpan di dalam database.

Data-data alert yang merupakan hasil dari sistem deteksi intrusi *host-based* dan *network-based* digunakan sebagai data awal proses korelasi sistem deteksi intrusi. Data-data yang tersedia pada database dilakukan query untuk mengambil beberapa data awal yang diperlukan.

3.1.2 Format Notifikasi Alert.

Format alert untuk menampilkan hasil korelasi menggunakan format yang telah distandarisasi pada tahap normalisasi. Atribut-atributnya telah disamakan dengan atribut yang terdapat pada IDMEF untuk mempermudah pembacaan hasil korelasi.

Hasil korelasi kedua system deteksi intrusi disimpan dalam file berekstensi *.alert dan/atau dikirimkan ke email user untuk mempermudah akses notifikasi alert.

3.2 Desain Sistem

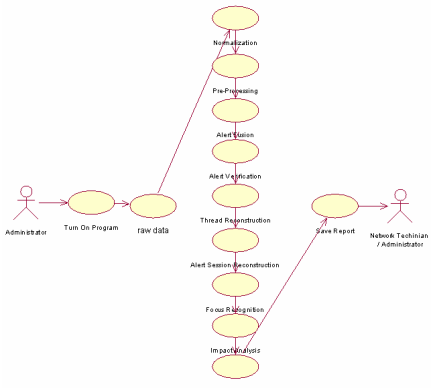
Desain system ini meliputi desain data alert dari system deteksi intrusi, baik itu system deteksi intrusi *host-based* dan system deteksi intrusi *network-based*, yang dimasukkan ke dalam system korelator, hingga data hasil korelasi system deteksi deteksi intrusi dapat dibaca oleh user.

3.2.1 Desain Input

Input dari system ini adalah database dari system deteksi intrusi, system deteksi intrusi *host-based* dan system deteksi intrusi *network-based*. Data-data tersebut digunakan sesuai dengan format yang dibutuhkan system korelator alert.

3.2.2 Alur Program

Berikut adalah usecase dari program yang dilewati oleh alert dan komponen-komponen korelasi alert .



Gambar 3.2 usecase program

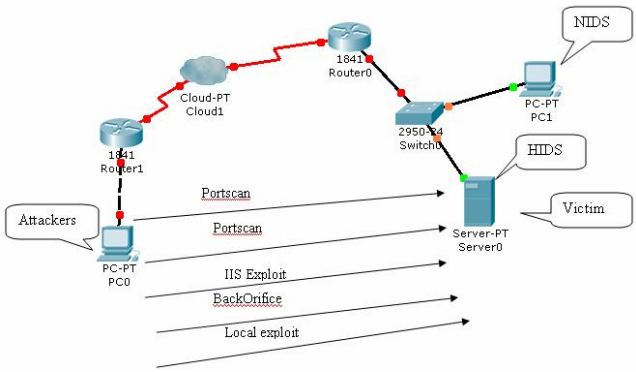
Setelah aplikasi dijalankan, maka aplikasi secara otomatis akan melakukan pengambilan data dari database sistem deteksi intrusi, baik itu sistem deteksi intrusi *host-based* dan *network-based*. Kemudian data tersebut dikirimkan ke korelator alert untuk dilakukan pemrosesan baik itu penambahan atribut atau pereduksian alert karena melewati tahap korelasi dengan beberapa persyaratan korelasi. Setelah melewati delapan tahap komponen korelasi, maka hasil dari proses tersebut akan disimpan ke dalam file dan dikirimkan ke email.

3.2.2.1 Komponen Korelasi Alert

Korelasi alert adalah proses multikomponen yang menerima input berupa aliran alert dari beberapa sistem deteksi intrusi. Pada setiap komponen proses, alert digabungkan menjadi laporan intrusi yang levelnya lebih tinggi, atau ditandai sebagai non-relevant jika alert tersebut menunjukkan serangan yang sukses. Akhirnya, alert diprioritaskan sesuai dengan kebijakan keamanan dan menjadi laporan alert.

3.2.2.2 Skenario Serangan

Untuk melakukan evaluasi terhadap sistem korelasi alert, maka akan dibuat skenario serangan berikut :



Gambar 3.3 Skenario Attack

Dengan menggunakan skenario serangan diatas, setiap tahap dari korelasi alert dapat diketahui hasil dari perubahan pada setiap tahap.

3.2.2.3 Atribut yang Ditambahkan

Atribut yang digunakan pada aleran alert ini merujuk ke IDMEF, sehingga format tersebut lebih mudah untuk dipahami. Berikut adalah daftar atribut (isi alert) yang ditambahkan pada alert :

Attribut alert	Penjelasan
Alertid	ID unik yang menunjukkan alert
Analyzertime	Waktu ketika IDS mengirimkan alert
Attackernodes	Nodes yang merupakan sumber serangan
Consequence	Sistem yang menjadi sasarang serangan
Createtime	Waktu ketika IDS mengenerate waktu
Detecttime	Waktu ketika IDS mendeteksi serangan
End_time	Waktu ketika serangan terhenti
Name	Nama serangan
Priority	Nilai yang menunjukkan tingkat serangan
Receivedtime	Waktu yang diperlukan alert untuk diterima korelator
Reference	Referensi ke serangan lain
Sensornode	Node tempat menggenerate alert
Start_time	Waktu ketiak serangan dimulai
Tipe	Tipe serangan yang terjadi
Verified	Jika serangan berhasil dilancarkan
Victimnodes	Node korban yang diserang
Victimservice	Port dan protokol service yang diserang

Table 3.1 Atribut yang Ditambahkan

3.2.2.4 Meta Alert

Isi dari meta-alert adalah sama dengan alert lain, tetapi asal dari meta-alert merupakan turunan dari beberapa alert yang digabungkan. Keputusan apakah suatu alert digabungkan atau tidak, tergantung pada komponen korelasi masing-masing dan pada nilai atribut alert yang relevan. Tujuan dari pembuatan meta-alert ini adalah mengurangi informasi serangan dan menampilkan alert instant yang merangkum semua informasi yang relevan untuk dapat analisa.

3.2.2.5 Tahap-Tahap Korelasi.

Tahap-tahap korelasi alert yang digunakan pada project ini telah digambarkan pada *usecase program*, yaitu normalisasi, pre-processing, alert fusion, alert verification, thread reconstruction, attack session reconstruction, focus recognition, impact analysis, dan prioritazion.
Berikut adalah penjelasan tahap-tahap tersebut :

a) **Normalization :** alert yang dihasilkan oleh beberapa vendor dan grup riset yang berbeda, maka akan memiliki format yang berbeda. Dengan mengggunakan normalisasi ini, maka format dari semua alert akan disamakan standart penulisannya agar dapat dikenali dan diproses. Berikut adalah pseudocode dari tahap *normalization*

```
1. global normalization_db,
     alertname_db
2. normalize(raw_alert) {
3. alert ← new alert
4. alert.alertid ← get_unique_id()
5. alert.name ← get alertname from
     alertname_db using
6. (raw_alert.name,
     raw_alert.sensortype)
7. as key
8. mappings ← get all m:mapping from
     normalization_db
9. where m.sensor = raw_alert.sensor
10.for each m:mapping in mappings:
11.alert_attr ← m.alert_attr
12.raw_attr ← m.raw_attr
13.alert.alert_attr←raw_alert.raw_attr
14.pass alert to next correlation
     component
15.}
```

b)Pre-Processing : Alert yang ternormalisasi adalah alert telah distandarkan namanya dan dapat dibaca oleh sistem korelasi. Tetapi tahap pre-processing ini diperlukan untuk menambahkan data-data yang diperlukan karena pada sensor tertentu terkadang menghilangkan atau tidak mencantumkan data penting yang akan digunakan untuk proses korelasi. Pada project ini, data yang ditambahkan adalah waktu alert. Untuk menambahkan waktu alert, ada beberapa macam waktu yang dapat digunakan, yaitu `createTime` (waktu ketika alert digenerate oleh sistem deteksi intrusi) , `detectTime` (waktu ketika event mengeluarkan alert yang dideteksi oleh sistem deteksi intrusi) dan `AnalyzerTime` (waktu pada IDS ketika alert dikirimkan ke sistem korelasi). Berdasarkan prioritas presisinya, maka `DetectTime` menjadi prioritas utama, kemudian diikuti `createTime` dan yang terakhir adalah `analyzerTime`. Sehingga ketika di dalam telah tersedia field untuk `detectTime`, maka lebih baiknya kita mengambil data dari

detectTime daripada yang lain. Berikut adalah psedocode dari preprocessing

```
1.  global attack_type_db
2.  preprocess(alert) {
3.  if alert.start_time is null:
4.  if alert.detecttime is not null:
5.  alert.start_time←alert.detecttime
6.  else if alert.createtime is not
     null:
7.  alert.start_time←alert.createtime
8.  else if alert.analyzertime is not
     null:
9.  alert.start_time←alert.analyzertime
10. else:
11. alert.start_time←alert.receivedtime
12. if alert.end_time is null:
13.     alert.end_time←alert.start_time
14. if alert is produced by a host-
     based IDS:
15.
     alert.victimnodes←alert.sensornod
     e
16.
     alert.attackernodes←alert.sensorn
     ode
17. alert.type←get    alertype    from
     attack_type_db using
18. alert.name as key
19. pass alert to next component
20. }
```

c) Alert Fusion : tujuan dari alert fusion adalah menggabungkan alert yang muncul secara sendiri-sendiri dan memiliki nama yang sama pada deteksi sistem deteksi intrusi. Ketika ada alert yang baru masuk ke dalam proses korelasi, maka alert tersebut dimasukkan ke dalam antrian alert berdasarkan urutan waktu. Kemudian alert tersebut dibandingkan dengan alert yang terdapat dalam antrian dimulai dari urutan waktu yang paling awal. Jika semua atribut tersebut memiliki kecocokan dengan alert yang terdapat pada antrian, maka alert tersebut ”*match*” dan bisa digabungkan. Jika alert tersebut tidak ada yang cocok dengan semua alert dari semua antrian alert yang ada,maka alert tersebut dimasukkan ke dalam antrian alert tersebut. Berikut adalah pseudocode dari alert fusion :

```
1.  global attack_type_db
2.   preprocess(alert) {
3.   if alert.start_time is null:
4.   if alert.detecttime is not null:
5.   alert.start_time←alert.detecttime
6.   else if alert.createtime is not null:
7.   alert.start_time←alert.createtime
8.   else if alert.analyzertime is not null:
9.   alert.start_time←alert.analyzertime
10.  else:
11.  alert.start_time←alert.receivedtime
12.  if alert.end_time is null:
13.      alert.end_time←alert.start_time
14.  if alert is produced by a host-based
     IDS:
15.      alert.victimnodes←alert.sensornode
16.
     alert.attackernodes←alert.sensornode
17.  alert.type←get      alertype      from
     attack_type_db using
18.  alert.name as key
19.  pass alert to next component
20. }
```

Gambar 3.7(a) Pseudocode fusi alert

d) **Alert Verification :** tujuan dari komponen ini adalah untuk menghapus (memberi tanda) alert yang tidak menunjukkan alert dari serangan yang berhasil. Metode verifikasi terdapat dua cara : aktif dan pasif. Untuk metode pasif, bergantung pada informasi priori yang dikumpulkan tentang host, topologi jaringan, dan service yang diinstall. Dengan menggunakan model dari topologi jaringan, maka akan dapat diketahui apakah target dari serangan tersebut ada atau tidak, dan service yang potensial untuk celah tersebut berjalan atau mati. Sedangkan dengan metode aktif, teknik ini harus melihat bukti bahwa suatu serangan tersebut harus sukses dilakukan, dengan cara mengecek informasi pada mesin korban. Berikut adalah pseudocode dari *alert verification* :

```
1.  global database_port
2.  verify(alert) {
3.  scripts ← get all s:script from
        nasl_scripts
4.  where s.name = alert.name
5.  if scripts is null:
6.  alert.verified ← unknown
7.  else:
8.  for each s:script in scripts:
a.  run s on alert.victimhosts
b.  if  script  reported  successful
        exploit:
c.  alert.verified ← true
d.  break
e.  else:
f.  alert.verified ← false
9.  pass  alert  to  next  correlation
        component
10. }
```

e) Attack Thread Reconstruction : tahap ini menggabungkan serangkaian alert yang merujuk yang dilakukan oleh seorang penyerang ke satu target. Tujuan komponen ini adalah meng-korelasikan alert yang dilakukan penyerang dengan menggunakan exploit yang berbeda-beda ke satu target atau menjalankan satu exploit dengan berkali-kali untuk mencoba variable yang tepat untuk menyerang. Attack Thread dibuat dengan cara menggabungkan alert dengan mencari kesamaan antara atribut sumber dan target serangan yang terjadi pada satu waktu tertentu. Berikut adalah pseudocode dari *attack thread reconstruction* :

```
1.  parameter window_size
2.  global alert_queue
3.  attack_thread(alert) {
4.  remove all a:alert from alert_queue
        where
5.  a.start_time < (alert.start_time -
        window_size)
6.  pass removed alerts to next correlation
        component
7.  al ← get a:alert with lowest
```

```
        start_time from
8.  alert_queue where (alert.victimhosts =
        a.victimhosts and
        alert.attackerhosts =
        a.attackerhosts)
9.  if al is not null:
10. replace al in alert_queue with
11. thread_merge(alert, al)
12. else:
13. add alert to alert_queue
14. }
15. thread_merge(alert1, alert2) {
16. r ← new alert
17. r.alertid ← get_unique_id()
18. r.start_time ← min(alert1.start_time,
19. alert2.start_time)
20. r.end_time ←
        max(alert1.end_time,alert2.end_tim
        e)
21. r.analyzer = alert1.analyzer U
        alert2.analyzer
22. r.reference ← alert1.alertid U
        alert2.alertid
23. if alert1.name = alert2.name:
24. r.name ← alert1.name
25. else:
26. r.name ← "Attack Thread"

27. for each attr:attribute except
        start_time, end_time, reference,
        analyzer, alertid:
28. if alert1.attr = alert2.attr:
    r.attr ← alert1.attr
29. else:
    r.attr ← null
30. return r
31. }
```

f) Attack Session Reconstruction : tujuan dari Attack session reconstruction adalah menghubungkan alert network based dengan alert host based.sensor network based biasanya memberikan sumber dan tujuan paket serta port yang memiliki bukti serangan, sedangkan host based sensor termasuk proses yang diserang dan juga user yang melakukannya karena itu informasi network based (IP dan port) dan informasi hostbased (proses dan identifier) tidak sejalan. Implementasinya adalah membuat database pemetaan antara port jaringan yang menerima koneksi dan nama serta user id proses yang terkoneksi dengan port tersebut. Berikut adalah pseudocode dari *attack session reconstruction:*

```
1.  parameter timeout
2.  global session_list, service_db
3.  attack_session(alert) {
4.  remove all e:element from session_list
      where
5.  e.timestamp < (alert.start_time -
      timeout)
6.  for each removed element e:
7.  if e.state = "Host":
8.  pass session_merge(e.alerts) to next
      correlation component
9.  else:
10. pass e.alerts to next correlation
      component
11. if alert is produced by a network-
      based IDS:
12. matching_session get s:session with
      lowest timestamp
13. from session_list where
14. alert.victimhosts = s.victimhosts and
15. alert.victimservice = s.victimservice
      and
16. s.state = "Network"
```

```
17. if matching_session is null:
18. matching_session new session
19. matching_session.victimhosts ←
20. alert.victimhosts
21. matching_session.victimservice ←
      alert.victimservice
22. matching_session.state ← "Network"
23. matching_session.timestamp ←
      alert.start_time
24. add alert to matching_session.alerts
25. else if alert is produced by a host-
      based IDS:
26. alert.victimservice get s:service from
      service_db where
27. s.host = alert.victimhosts and s.process
      = alert.victimprocess
28. matching_session get s:session with
      lowest timestamp
29. from session_list where
30. alert.victimhosts = s.victimhosts and
31. alert.victimservice = s.victimservice
32. if matching_session is null:
33. pass alert to next correlation component
34. else:
35. matching_session.timestamp ←
      alert.start_time
36. add alert to matching_session.alerts
37. matching_session.state "Host"
38. }
39. session_merge(alert_list) {
40. r ← new alert
41. r.alertid ← get_unique_id()
42. r.start_time ← min start_time of all
      alerts in alert_list
43. r.end_time ← max end_time of all
      alerts in alert_list
44. r.analyzer ← union of all analyzer in
      alert_list
45. r.reference ← union of all alertid in
      alert_list
46. if name is equal for all alerts in
      alert_list:
47. r.name ← alert_list[1].name
48. else:
49. r.name ← "Attack Session"
50. for each attr:attribute except
      start_time,
51. end_time, reference, analyzer, name,
52. alertid:
53. if attr is equal for all alerts in
      alert_list:
54. r.attr ← alert[1].attr
55. else:
56. r.attr ← null
57. return r
58. }
```

g) Attack Focus Recognition : tujuan dari komponen focus recognition adalah mengidentifikasi host yang baik sumber maupun target serangan, mengalami serangan berkali-kali. Hal ini untuk memastikan bahwa adanya serangan seseorang terhadap banyak target maupun banyaknya sumber yang menyerang satu target. Hal ini dilakukan untuk menghindari adanya DDOS dan scan skala besar. Alert yang berkaitan dengan DDOS dapat digabungkan menjadi *many2one* meta-alert, sedangkan alert yang merujuk ke satu scan dapat digabungkan ke single *one2many* meta-alert.

jika memungkinkan, alert yang digenerate oleh scenario *one2many* dan *many2one* diklasifikasikan lebih jauh sebagai serangan denial-of-Service. Lebih presisi lagi, serangan many2one dikelompokkan sebagai denial of service ketika jumlah serangan yang dilakukan mencapai batas yang telah ditentukan. Alert *one2many* dikelompokkan sebagai

scan horizontal ketika a port single discan pada semua mesin korban, atau sebagai multiscan horizontal ketika lebih dari satu port yang discan.

```
parameter: timeout, report_threshold
global scenarios

one2many(alert) {
remove all s:scenario from scenarios where
s.timestamp < (alert.start_time - timeout)
for each removed scenario s:
if number of distinct victimhosts in
s.alerts > report_threshold:
send onetomany_merge(s.alerts)
      to next correlation component
else:
      send s.alerts to next correlation component
scenario get s:scenario from scenarios
where s.hosts = alert.attackerhosts
if scenario is null:
scenario ← new scenario
scenario.hosts ← alert.attackerhosts
add scenario to scenarios
scenario.timestamp ← alert.start_time
add alert to scenario.alerts
}
one2many_merge(alert_list) {
r ← new alert
r.alertid ← get_unique_id()
r.start_time ← min start_time of all alerts in alert_list
r.end_time ← max end_time of all alerts in alert_list
r.analyzer ← union of all analyzer in alert_list
r.reference ← union of all alertid in alert_list
if all alerts in alert_list refer to same port
number:
r.name ← "Horizontal Scan"
else:
r.name ← "Horizontal Multiscan"
for each attr:attribute except start_time,
end_time, reference, analyzer, name,
alertid:
if attr is equal for all alerts in
alert_list: r.attr ← alert[1].attr
else:
r.attr ← null
return r
}
```

h)Impact Analysis : tujuan dari komponen ini adalah untuk menentukan dampak serangan pada operasi yang berjalan pada jaringan yang dilindungi. Proses impact analysis ini menggabungkan alert dari komponen korelasi sebelumnya dengan data yang terdapat pada asset database dan jumlah *"heartbeat monitors"*. Asset database menyimpan data tentang service-service jaringan yang diinstall, kebergantungan antar service, dan urgensi dari instalasi service ini.sebagai contoh mail server membutuhkan service dari DNS untuk berjalan dengan lebih baik. Tujuan dari heartbeat monitor adalah memastikan bahwa service tersebut berjalan dan tidak mati.

ketika alert mempengaruhi service jaringan tertentu, asset database akan mencari keterangan ke semua service yang berkaitan pada target tersebut. Heartbeat monitor kemudian mengecek apakah service yang berkaitan tersebut masih beroperasi atau tidak. Jika service tersebut diketahui sudah tidak berjalan,maka data ini dapat dimasukkan ke alert sebagai akibat dari serangan yang baru saja masuk..

```
parameter window_size
global asset_database, attack_table
impact_processalert(alert) {
remove all a:alert from attack_table where
a.start_time < (alert.start_times - window_size)
if alert.type is not "Reconnaissance":
add alert to attack_table
pass alert to next correlation component
}
impact_processheartbeat(heartbeat) {
if heartbeat.status = "Down":
dependencies ← get all a:asset from
asset_database where
(heartbeat.victimhost,
heartbeat.victimservice) is
dependent on a
for each dependecy:asset in dependencies:
attacks ← get all a:alert from attack_table
where dependency.host ∈ a.victimhosts and
dependency.service = a.victimservice
for each alert in attacks:
alert.consequence ← alert.consequence U
(heartbeat.victimhost,
heartbeat.victimservice)
}
```

3.11(a) Pseudocode *impact analysis*

i) Priorization :prioritas adalah tingkat urgensi suatu serangan untuk mengklasifikasi alert dan menghapus informasi yang tidak relevan atau tidak penting. Tujuan dari alert priorization adalah memberikan prioritas ke alert. Komponen ini harus disesuaikan dengan security policy dan security requirement pada tempat korelasi ini dijalankan. Berikut adalah pseudocode dari *alert prioritization* :

```
global asset_database, dos_attacks,
read_attacks, write_attacks
prioritize(alert) {
asset_list get all a:asset from
asset_database where:
a.host ∈ alert.victimhosts and
a.service = alert.victimservice
if alert.name ∈ dos_attacks:
alert.priority ← max availability_score of
   all assets in asset_list
else if alert.name ∈ read_attacks:
alert.priority ← max secrecy_score of all
assets in asset_list
else if alert.name ∈ write_attacks:
alert.priority ← max integrity_score of all
assets in asset_list
}
```

3.2.2.6 Desain Output

Desain output dari yang dihasilkan dari hasil korelasi tersebut adalah berupa plaintext. Kemudian plaintext tersebut dapat ditampilkan kedalam textbox user interface, disimpan dalam file dan/atau dikirimkan ke email

4. UJI COBA DAN ANALISA

Metode pengujian yang digunakan dengan cara membandingkan alert pada setiap tahap yang dilewati. Alert yang digunakan untuk menguji adalah alert yang membentuk scenario serangan terhadap system, baik itu serangan terhadap jaringan, aplikasi atau serangan local computer. Table 4.1 menggambarkan scenario serangan yang digunakan untuk proses pengujian korelasi system deteksi intrusi.

ID	Nama	Sensor	Time	Sumber	target	tag
1	IIS Exploit	N1	12.0/12.0	202.154.178.31	10.252.111.98/80	
2	Scanning	N2	10.1/14.8	202.154.178.6	10.252.111.98	
3	PortScan	N1	10.0/15.0	202.154.178.6	10.252.111.98	
4	Apache Exploit	N1	22.0/22.0	202.154.178.6	10.252.111.98/80	
5	Bad Request	A	22.1/22.1		localhost, Apache	
6	Local Exploit	H	24.6/24.6		linuxconf	
7	Local Exploit	H	24.7/24.7		linuxconf	
8	Backorifice	N1	25.8/25.9	202.154.178.6	10.252.111.50	
9	Backorifice	N1	27.8/27.9	202.154.178.6	10.252.111.69	

Tabel 4.1 Skenario serangan

Tabel 4.1 Skenario serangan

Selanjutnya, scenario serangan dari table 4.1, akan dilewatkan ke dalam tahap korelasi system deteksi intrusi .

4.1.1 Normalisasi :

Tahap normalisasi ini melakukan standarisasi penamaan dan format pada alert. Dapat dilihat pada table 4.2.

ID	Nama	Sensor	Time	Sumber	target	tag
1	IIS Exploit	N1	12.0/12.0	202.154.178.31	10.252.111.98 /80	
2	**PortScan**	N2	10.1/14.8	202.154.178.6	10.252.111.98	
3	**PortScan**	N1	10.0/15.0	202.154.178.6	10.252.111.98	
4	Apache Exploit	N1	22.0/22.0	202.154.178.6	10.252.111.97 /80	
5	Bad Request	A	22.1/22.1		localhost, Apache	
6	Local Exploit	H	24.6/24.6		linuxconf	
7	Local Exploit	H	24.6/24.6		linuxconf	

Tabel 4.2 Tahap Normalisasi

Pada table 4.1 yang berisi tentang scenario serangan, alert pada ID kedua bernama **scanning**, ketika melewati tahap normalisasi berubah menjadi **portscan**. Hal ini dilakukan untuk menyamakan format penamaan yang digunakan pada korelator system deteksi intrusi.

4.1.2 Preprocessing :

Pada tahap *preprocessing*, alert yang telah dinormalisasi akan dilakukan pengecekan pada atribut yang diperlukan tetapi dalam keadaan kosong, yaitu *time* dan *target*. Pada table 4.1 yang berisi scenario serangan, digambarkan bahwa terdapat kekosongan data pada sumber dan target serangan pada alert yang dihasilkan system deteksi intrusi *host-based*. Kemudian kekosongan tersebut diisikan sumber dan target serangan yang sama. Perhatikan pada table 4.3

ID	Nama	Sensor	Time	Sumber	target	tag
1	IIS Exploit	N1	12.0/ 12.0	202.154.178.31	10.252.111.98/ 80	
2	PortScan	N2	10.1/ 14.8	202.154.178.6	10.252.111.98	
3	PortScan	N1	10.0/ 15.0	202.154.178.6	10.252.111.98	
4	Apache Exploit	N1	22.0/ 22.0	202.154.178.6	10.252.111.98/ 80	
5	**Bad Request**	**A**	**22.1/ 22.1**	**10.252.111.98**	**10.252.111.98, localhost, Apache**	
6	**Local Exploit**	**H**	**24.6/ 24.6**	**10.252.111.98**	**10.252.111.98, linuxconf**	
7	**Local Exploit**	**H**	**24.7/ 24.7**	**10.252.111.98**	**10.252.111.98, linuxconf**	
8	Backorifi ce	N1	25.8/ 25.9	202.154.178.6	10.252.111.50	
9	Backorifi ce	N1	27.8/ 27.9	202.154.178.6	10.252.111.69	

Tabel 4.2 Tahap preprocessing

Hal ini dilakukan karena pada system deteksi intrusi *host-based* serangan dan target dilakukan pada computer yang sama.

4.1.3 *Alert Fusion*

Alert fusion adalah mengkorelasikan serangan yang mempunyai kesamaan pada *nama*, *sumber* dan *target* serangan. Dapat kita lihat dari table yang melewati tahap sebelumnya, yaitu table 4.2. dari scenario serangan yang ada, terdapat beberapa alert yang memiliki nama, sumber serangan dan target serangan yang sama. Maka alert tersebut dapat kita korelasikan dan menghapus data yang memiliki kesamaan dan membuat *meta-alert*. Proses korelasi alert dapat dilihat pada table 4.3.

ID	Nama	Sensor	Time	Sumber	Target	Tag
1	IIS Exploit	N1	12.0/ 12.0	202.154.178.31	10.252.111.98/ 80	
2	PortScan	N2	10.1/ 14.8	202.154.178.6	10.252.111.98	**Correlated**
3	PortScan	N1	10.0/ 15.0	202.154.178.6	10.252.111.98	**Correlated**
4	Apache Exploit	N1	22.0/ 22.0	202.154.178.6	10.252.111.98/ 80	
5	Bad Request	A	22.1/ 22.1	10.252.111.98	10.252.111.98, localhost, Apache	

6	Local Exploit	H	24.6/ 24.6	10.252.111.98	10.252.111.98, linuxconf	**Correlated**
7	Local Exploit	H	24.7/ 24.7	10.252.111.98	10.252.111.98, linuxconf	**Correlated**
8	Backorifice	N1	25.8/ 25.9	202.154.178.6	10.252.111.50	
9	Backorifice	N1	27.8/ 27.9	202.154.178.6	10.252.111.69	
8	**Meta-Alert**	**{N2, N1}**	**10.1/ 15.0**	**202.154.178.6**	**10.252.111.98**	**{2,3}**
9	**Meta-Alert**	**H**	**24.6/ 24.7**	**10.252.111.98**	**10.252.111.98, linuxconf**	**{6,7}**

Tabel 4.3 Tahap preprocessing

Pada table 4.3 diatas, alert pada ID 2 dan 3, serta alert dengan ID ke 6 dan 7, keempat alert tersebut memiliki kesamaan pada nama, sumber dan target serangan. Maka alert dengan ID 2 dan 3 dilakukan korelasi dan membentuk meta-alert seperti yang ditunjukkan pada alert 8. Pada alert dengan ID 6 dan 7 pun demikian, membuat meta-alert baru dengan ID 9. pembuatan meta-alert ini dengan menggabungkan kedua data alert yang terkorelasi, seperti waktu mulai dan berhentinya serangan, serta kedua alert ID.

4.1.4 Alert verification

Tahap *alert verification* diharapkan akan melakukan pemberitahuan suatu serangan tersebut relevan terhadap *host* yang diserang atau tidak. Perhatikan table 4.4 dibawah ini.

ID	Nama	Sensor	Time	Sumber	Target	Tag
1	**IIS Exploit**	**N1**	**12.0/ 12.0**	**202.154.178.31**	**10.252.111.98/ 80**	**irrelevant**
2	PortScan	N2	10.1/ 14.8	202.154.178.6	10.252.111.98	
3	PortScan	N1	10.0/ 15.0	202.154.178.6	10.252.111.98	
4	Apache Exploit	N1	22.0/ 22.0	202.154.178.6	10.252.111.98/ 80	
5	Bad Request	A	22.1/ 22.1	10.252.111.98	10.252.111.98, localhost, Apache	
6	Local Exploit	H	24.6/ 24.6	10.252.111.98	10.252.111.98, linuxconf	
7	Local Exploit	H	24.7/ 24.7	10.252.111.98	10.252.111.98, linuxconf	
8	Backorifice	N1	25.8/ 25.9	202.154.178.6	10.252.111.50	
9	Backorifice	N1	27.8/ 27.9	202.154.178.6	10.252.111.69	
8	Meta-Alert	{N2, N1}	10.1/ 15.0	202.154.178.6	10.252.111.98	{2,3}
9	Meta-Alert	H	24.6/ 24.7	10.252.111.98	10.252.111.98, linuxconf	{6,7}

Tabel 4.4 Tahap *Alert Verification*

4.1.5 Attack Thread Reconstruction

Pada tahap korelasi ini, *alert* yang memiliki kesamaan serangan pada atribut sumber serangan tetapi menjalankan serangan ke system dengan berbeda teknik akan dideteksi. Pada table 4.5 menunjukkan table serangan yang menunjukkan kesamaan sumber tetapi memiliki target system yang berbeda.

ID	Nama	Sensor	Time	Sumber	Target	Tag
1	IIS Exploit	N1	12.0/ 12.0	202.154.178.31	10.252.111.98/ 80	irrelevant
2	PortScan	N2	10.1/ 14.8	202.154.178.6	10.252.111.98	Fused
3	PortScan	N1	10.0/ 15.0	202.154.178.6	10.252.111.98	Fused
4	**Apache Exploit**	**N1**	**22.0/ 22.0**	**202.154.178.6**	**10.252.111.98/ 80**	**correlated**
5	Bad Request	A	22.1/ 22.1	10.252.111.98	10.252.111.98, localhost, Apache	
6	Local Exploit	H	24.6/ 24.6	10.252.111.98	10.252.111.98, linuxconf	Fused
7	Local Exploit	H	24.7/ 24.7	10.252.111.98	10.252.111.98, linuxconf	Fused
8	Backorifice	N1	25.8/ 25.9	202.154.178.6	10.252.111.50	
9	Backorifice	N1	27.8/ 27.9	202.154.178.6	10.252.111.69	
10	**Meta-Alert**	**{N2, N1}**	**10.1/ 15.0**	**202.154.178.6**	**10.252.111.98**	**{2,3}, correlated**
11	Meta-Alert	H	24.6/ 24.7	10.252.111.98	10.252.111.98, linuxconf	{6,7}
12	**Meta-Alert**	**{N1, {N2, N1}}**	**22.0/ 15.0**	**202.154.178.6**	**10.252.111.98/80**	**{4,8}**

Tabel 4.5 Tahap *Attack Thread Reconstruction*

Pada table 4.5 diatas, *alert* pada ID ke 4, yaitu serangan *apache exploit* memiliki kesamaan sumber dan target tetapi memiliki variasi serangan dengan meta-alert pada serangan dengan ID ke 8. maka kedua alert tersebut membuat satu meta-alert tersendiri.

4.16 *Attack Session Reconstruction*

Attack Session Reconstruction diharapkan akan melakukan korelasi terhadap serangan yang berjalan pada *host* dan jaringan dengan memanfaatkan data aliran alert yang terdapat pada system deteksi intrusi *host-based* dan juga system deteksi intrusi *network-based*. Korelasi yang dilakukan pada attack session reconstruction ini adalah sama, tetapi dideteksi oleh sensor yang berbeda. Table 4.6 menggambarkan korelasi *attack session reconstruction*.

ID	Nama	Sensor	Time	Sumber	Target	Tag
1	IIS Exploit	N1	12.0/ 12.0	202.154.178.31	10.252.111.98/80	Irrelevant
2	PortScan	N2	10.1/ 14.8	202.154.178.6	10.252.111.98	Fused
3	PortScan	N1	10.0/ 15.0	202.154.178.6	10.252.111.98	Fused
4	Apache Exploit	N1	22.0/ 22.0	202.154.178.6	10.252.111.98/80	Threaded
5	**Bad Request**	**A**	**22.1/ 22.1**	**10.252.111.98**	**10.252.111.98, localhost, Apache**	**Correlated**
6	Local Exploit	H	24.6/ 24.6	10.252.111.98	10.252.111.98, linuxconf	Fused
7	Local Exploit	H	24.7/ 24.7	10.252.111.98	10.252.111.98, linuxconf	Fused
8	Backorifice	N1	25.8/ 25.9	202.154.178.6	10.252.111.50	
9	Backorifice	N1	27.8/ 27.9	202.154.178.6	10.252.111.69	

10	Meta-Alert	{N2, N1}	10.1/ 15.0	202.154.178 .6	10.252.111.9 8	{2,3 }, Thre aded
11	Meta-Alert	H	24.6/ 24.7	10.252.111. 98	10.252.111.9 8, linuxconf	{6,7 }
12	**Meta-Alert**	**{N1, {N2, N1}}**	**22.0/ 15.0**	**202.154.178 .6**	**10.252.111.9 8/80**	**{4,8} , corr elate d**
13	**Meta-Alert**	**{A, {N2, N1}}**	**22.1**	**10.252.111. 98, 202.154.178 .6**	**10.252.111.9 8/ apache, 80**	**{5,1 1}**

Tabel 4.6 Tahap *Attack Session Reconstructions*

Pada table 4.7 diatas, *alert* dengan ID ke 8 dan 9, terdapat serangan backorifice yang dilakukan oleh single target, yaitu oleh 31.3.3.7, tetapi memiliki target host yang berbeda, yaitu 10.0.0.3 dan 10.0.0.5. kedua alert tersebut maka digabungkan menjadi satu alert untuk membentuk meta-alert yang baru.

4.8 *Impact Analysis*

Impact analysis adalah tahap korelasi yang menjalankan untuk melakukan pengecekan terhadap system target. System yang telah menjadi target serangan dicek kondisi portnya apakah masih menunjukkan aktivitas atau tidak. Jika tidak maka aplikasi tersebut dalam keadaan tidak aktif, tetapi jika port tersebut aktif, maka aplikasi tersebut masih dalam keadaan berjalan.

ID	Nama	Sensor	Time	Sumber	Target	Tag
1	Email Exploit	N1	12.0/ 12.0	202.154.178.31	10.252.111.98/ 25	
2	**Bind9**	**N1**	**12.0/ 12.0**		**10.252.111.98/ 53**	**correlated**

Tabel 4.7 Tahap *impact analysis*

Pada tahap *Alert prioritization*, semua alert yang telah diuji dilakukan pembobotan nilai, kemudian diurutkan sesuai dengan tingkat. Pada table 4.9 akan digambarkan proses prioritiasasi *alert* yang telah melewati tahap korelasi *alert*.

ID	Nama	Status	Time	Sumber	Target	Point
1	backorifice	Metadata	25.8/ 27.9	202.154.178.6	10.252.111.50	7
2	Local exploit	Metadata	24.6/ 24.7	10.252.111.98	10.252.111.98	5
3	PortScan	Metadata	10.1/15 .0	202.154.178.6	10.252.111.98	2
4	IIS Exploit	irelevant	12.0/12 .0	202.154.178.31	10.252.111.20	1

Tabel 4.8 Tahap *Alert Prioritization*

4. KESIMPULAN DAN SARAN

- **Kesimpulan**

Dari hasil percobaan dan analisa yang dilakukan, maka dapat dianalisa kesimpulan :

1. *alert* berasal dari system deteksi intrusi baik system deteksi intrusi *host-based* dan system deteksi intrusi *network-based* dapat dilakukan proses reduksi *alert* untuk mengurangi jumlah *alert false positive*.

2. Prosentase *alert positive* yang direduksi dalam proses korelasi untuk melakukan proses *verifikasi alert* sebanding dengan

$$Nreduksi\% = \frac{Nkorelasi}{Nalert\_asli} x100\%$$

- **Saran**

Proyek akhir ini masih banyak kekurangan dalam pengerjaan pembuatan system verifikasi pada system deteksi intrusi. Baik itu dalam tahap korelasinya ataupun fitur-fitur yang tersedia pada aplikasi verifikasi *alert* pada system deteksi intrusi. Berikut saran untuk pengembangan penelitian verifikasi *alert* pada system deteksi intrusi kolaboratif :

Pada tahap korelasi komponen, tahap korelasi multi-step belum dikembangkan dan diintegrasikan ke dalam system korelasi untuk verifikasi *alert*. Komponen ini bisa dimanfaatkan untuk mengetahui alur serangan sejak pertama kali dimunculkan hingga akhir tahap terakhir serangan.

5 . DAFTAR PUSTAKA :

[1] Idris Winarno, 2008, Pengembangan Kolaborasi Sistem Deteksi Intrusi Jaringan Tersebar berbasis publish-subscribe dengan metode Alert Correlation. Tesis Program Master, Jurusan Teknik Informatika, Institut Teknologi sepuluh Nopember.

[2] Min Xiao, Debao Xiao, 2007, Alert Verification Based on Attack Classification in Collaborative Intrusion Detection System. In 8th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD).

[3] Herve Debar, "An Introduction to Intrusion-Detection Systems" IBM Research, Zurich Research Labolatory.

[4] F. Valeur, G. Vigna, C. Kruegel, R.A. Kemmerer, 2004,"A Comprehensive Approach to Intrusion Detection Alert Correlation". IEEE Transactions on Dependable and Secure Computing.

[5] C. Kruegel, W. Robertson, and G. Vigna, 2004,"Using Alert Verification to Identify Successful Intrusion Attempts". Practice in Information Processing and Communication (PIK), Vol.27, No.4 , pp. 219-227.

[6] C. Kruegel, G. Vigna, F. Valeur, 2005 "Intrusion Detection And Correlation : Challenges and Solution", Springer Science + bussiness media, Inc.