SISTEM PENDETEKSI SERANGAN PADA JARINGAN KOMPUTER MENGGUNAKAN SNORT BERBASIS SMS GATEWAY

(*STUDI KASUS di TAMAN PINTAR YOGYAKARTA*)

Untuk Memenuhi Sebagian Syarat Memperoleh Gelar Sarjana
Strata Satu

SKRIPSI

Disusun oleh:

KHAIRUL ANAM

NIM. 05650023

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2011

Universitas Islam Negeri Sunan Kalijaga **FM-UINSK-BM-05-07/R0**

PENGESAHAN SKRIPSI/TUGAS AKHIR

Nomor : UIN.02/D.ST/PP.01.1/1198/2011

Skripsi/Tugas Akhir dengan judul	:	Sistem Pendeteksi Serangan Pada Jaringan Komputer Menggunakan Snort Berbasis SMS Gateway (Studi Kasus di Taman Pintar Yogyakarta)
Yang dipersiapkan dan disusun oleh	:	
Nama	:	Khairul Anam
NIM	:	05650023
Telah dimunaqasyahkan pada	:	27 Juni 2011
Nilai Munaqasyah	:	A -

Dan dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga

TIM MUNAQASYAH :

Ketua Sidang

Landung Sudarmana, M.Kom
NIY.0527027001

Penguji I

Sumarsono, S.T, M.Kom
NIP.19710209 200501 1 003

Penguji II

Bambang Sugiantoro, M.T, CompTIA
NIP. 19751024 200912 1 002

Yogyakarta, 1 Juli 2011
UIN Sunan Kalijaga
Fakultas Sains dan Teknologi
Dekan

Prof. Drs. H. Akh. Minhaji, M.A, Ph.D
NIP. 19580919 198603 1 002

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi/Tugas Akhir
Lamp :

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Khairul Anam
NIM : 05650023
Judul Skripsi : ***Sistem Pendeteksi Serangan Pada Jaringan Komputer Menggunakan Snort Berbasis SMS Gateway (Studi Kasus di Taman Pintar Yogyakarta)***

sudah dapat diajukan kembali kepada Program Studi Teknik Informatika. Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Teknik Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqsyahkan. Atas perhatiannya kami ucapkan terima kasih.

Yogyakarta, 18 Juni 2011
Pembimbing I

Landung Landung Sudarmana, ST., M.Kom
NIP.

PERNYATAAN KEASLIAN SKRIPSI

Yang bertanda tangan dibawah ini

Nama : Khairul Anam

NIM : 05650023

Program Studi : Teknik Informatika

Fakultas : Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

Menyatakan bahwa skripsi dengan judul **"SITEM PENDETEKSI SERANGAN PADA JARINGAN KOMPUTER MENGGUNAKAN SNORT BERBASIS SMS GATEWAY (STUDI KASUS DI TAMAN PINTAR YOGYAKARTA)"** tidak pernah terdapat karya yang diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan berdasarkan penelusuran saya tidak terdapat karya atau dokumentasi yang pernah ditulis dan diterbitkan orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan didalam daftar pustaka

Yogyakarta, Juni 2011

Knairul Anam

NIM. 05650023

MOTTO

Tak ada kesuksesan yang didapat dengan mudah

Melakukan hal-hal yang tidak biasa
untuk mendapat sesuatu yang luar biasa

Investasikan waktu masa depan dengan mengerjakan hal-hal positif,
tidak ada kata terlambat untuk memulai hal baru.

Keberhasilan bukan ditentukan oleh besarnya otak seseorang, melainkan
oleh besarnya cara berfikir seseorang.

HALAMAN PERSEMBAHAN

Skripsi ini kupersembahkan sepenuhnya untuk :

Ibu dan Bapakku tercinta

Adikku tersayang

Teman-teman seperjuangan di kota pelajar

Almamater tercinta

Program Studi Teknik Informatika Fakultas Sains dan teknologi

UIN Sunan Kalijaga Yogyakarta

Pondok Pesantren Darul Ulum Banyuanyar Madura

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan pertolongan dan ilmu-Nya kepada penulis sehingga dapat terselesaikan penelitian ini. Penelitian yang berjudul "Sistem Pendeteksi Serangan Pada Jaringan Komputer Menggunakan Snort Berbasis SMS Gateway" yang mengambil contoh studi kasus di Taman Pintar Yogyakarta. Selanjutnya penulis mengucapkan terima kasih kepada :

1. Prof. Drs. H. Akh Minhaji, M.A, Ph.D, selaku Dekan Fakultas Sains & Teknologi UIN Sunan Kalijaga.
2. Bapak Agus Mulyanto, M.Kom, sebagai kepala program studi teknik informatika UIN Sunan Kalijaga.
3. Landung Sudarmana, ST., M.Kom, sebagai Dosen Pembimbing I yang telah banyak membantu dalam penyusunan skripsi ini.
4. Bapak M. Didik R. Wahyudi, ST, MT, sebagai Dosen Pembimbing II yang membantu penulis dalam penelitian dan dengan kesabarannya telah membimbing, memberikan koreksi, masukan kepada penulis selama penyusunan skripsi ini.
5. Para Dosen Program Studi Teknik Informatika yang telah memberi bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal sholeh yang berkesinambungan di dunia hingga akhirat.
6. Mas Riswinarno, Ridwan, Mas Ganang, Suhadi dan teman-teman Taman Pintar yang banyak membantu di memberikan bantuan fasilitas peralatan dan membantu di lapangan.

7. Kedua orang tuaku Buhasan dan Hosniyah yang telah rela mecucurkan keringat hanya untuk keberlangsungan masa studi ananda, serta adik-adikku tercinta Muhammad Iksan, Zainal Arifin, Abdurrahman dan keluarga besar yang selalu memberikan motivasi doa dan nasehat untuk senantiasa bersyukur atas semua nikmat yang diberikan Allah SWT.
8. Aji Kisworomukti, Ganjar Alfian, S.Kom, Sepran "indo-code" terima kasih banyak atas bantuan dan *sharing* ilmunya dalam penyelesaian dan persiapan skripsi ini.
9. Nurul Hidayati (terima kasih atas Motivasinya) M. Iqball Jalaluddin, Afriz, Novita Praci Putri, Ardhi 'kasdu', Nurul Bahiyah, serta teman-teman program Studi Teknik Informatika yang tidak bisa di tulis satu persatu khususnya angkatan 2005 yang telah banyak memberi dukungan.
10. Untuk Meli Amiati, terima kasih atas dukungan *support* dan motivasinya selama ini.
11. Teman-teman Kotrakan Forum Komunikasi Mahasiswa Santri Banyuanyar (FKMSB) Tanzil "Geng", Selamet " slem" dll, rekan-rekan laboratorium Teknik Informatika, Kelompok Studi Linux UIN Sunan Kalijaga Yogyakarta, Infinity IT-Club, Suka-Press, Suka-News, LPKM Introspektif, Komunitas Ubuntu Indonesia Sub Loco Jogjakarta, Komunitas Plurker Jogjakarta dan teman-teman yang lain yang tidak bisa penulis tulis satu persatu terima kasih telah membantu dan menyemangati agar skripsi ini selesai.

12. Terima kasih Pinky yang telah setia menemani mengahadirkan karya-karya dan membantu bertahan hidup di kota pelajar ini.

Penulis menyadari masih banyak kekurangan dan kelemahan dalam penelitian ini. Oleh karena itu demi perkembangan penelitian selanjutnya penulis sangat mengaharap kritik dan saran dari pembaca. Akhirnya semoga penelitian ini bermanfaat bagi pembaca.

Yogyakarta, Juni 2011

Penulis

DAFTAR ISI

BAB II TINJAUAN PUSTAKA DAN LANDASAN TEORI

BAB III METODE PENELITIAN

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ABSTRAK

Kejahatan dari pencurian informasi atau lebih dikenal dengan sebutan *cybercrime*, menggangu aktifitas jaringan, dan sampai pengrusakan sistem menjadi permasalahan yang juga dialami oleh TamPin. Seringkali jaringan di instansi pemerintah tersebut seringkali mengalami *down*, koneksi menjadi semakin lambat, dan dimunkinkan adanya penyusup yang mengakses informasi penting yang ada di instansi tersebut. Walau belum bisa dipastikan jaringan *trouble* dikarenakan kesalahan teknis, yang jelas di TamPin belum diterapkan manajemen keamanan sehingga keseluruhan sistem jaringan pada saat sekarang masih bias diakses oleh siapapun dan hal tersebut bias disalah gunakan oleh pihak yang tidak bertanggung jawab.

Untuk itu pada penelitian ini penulis menawarkan sistem pengamanan jaringan menggunakan *Intrusion Detection System* (IDS) untuk dipasang di TamPin. IDS sendiri bertugas sebagai pengawas sistem yang akan melakukan identifikasi akses oleh siapa saja yang menggunakan sistem. Mesin *tools* IDS yang akan digunakan pada studi kasus ini menggunakan snort yang akan diintegrasikan dengan SMS Gateway agar bisa memonitoring jaringan secara *realtime*.

Sistem keamanan yang ada di TamPin mampu memonitoring jaringan secara realtime dan memberikan report yang disimpan dalam bentuk log file. File-file yang tersimpan tersebut bisa di audit menggunakan AcidBase yang berbentuk grafis melalui web interface. Sedangkan smsgateway akan mengirimkan sms ke admin ketika terjadi serangan yang terindentifikasi oleh mesin snort.

Kata kunci : *cyber crime*, IDS, Snort, AcidBase, SMS Gateway.

ABSTRAC

Crime of theft of information or better known as cybercrime, disrupting network activity, and destruction of the system into the problems experienced by TamPin. Often these networks in government agencies often face down, connections become slowly, and the suspected presence of intruders who access the important information in that agency. Although not yet certain network trouble due to technical errors, which clearly TamPin not been applied so that the overall security management system in the present in network can still be accessed by anyone and it can be misused by irresponsible parties.

Therefore in this study the author offers a network security system using the Intrusion Detection System (IDS) to be installed in TamPin. IDS it self served as a supervisor system that will identify access by anyone using the system. IDS machine tools to be used in this case study is using snort to be integrated with the SMS Gateway in order to monitor the network in real time.

The existing system security in TamPin able to monitor the network in real time and provide a report that is stored in a log file. The files are stored can be audited using the form AcidBase graphics via the web interface. While SMS Gateway will send sms to admin when an attack is identified by the Snort engine.

Keywords : Cyber Crime, IDS, Snort, AcidBase, SMS Gateway.

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Perkembangan teknologi dibidang pendidikan ataupun bisnis dewasa ini mengalami pertumbuhan sangat signifikan, seiring laju perkembangan Teknologi Informasi dan Komunikasi global, lembaga yang telah memutuskan untuk memasang perangkat Teknologi Informasi dan Komunikasi (TIK) harus benar mampu untuk mengimplementasikan secara tepat agar bisa meningkatkan laju organisasi agar lebih baik dan mempunyai daya saing tinggi.

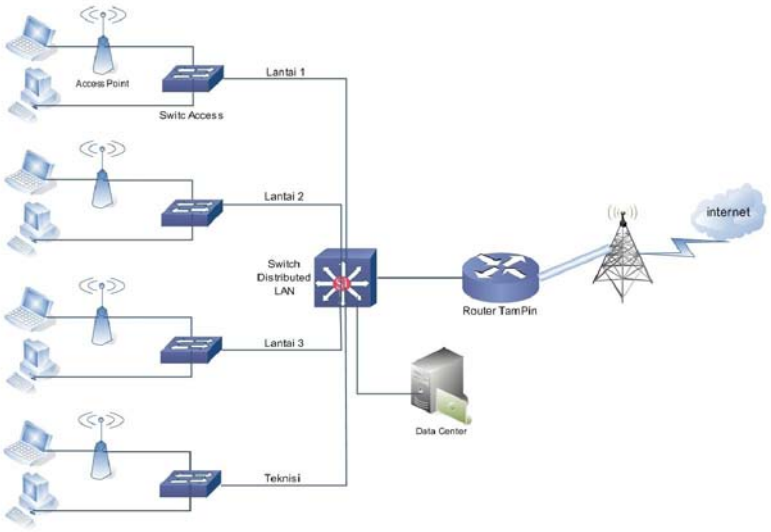
Adanya perangkat teknologi yang serba *modern* atau canggih akan tidak ada artinya tanpa diimbangi oleh pengaturan dan penggunaan secara tepat efektif dan efesien. Perangkat yang sederhana namun dikelola secara tepat bisa menstabilkan bahkan akan sangat membantu terhadap perkembangan perusahaan, hal tersebut disebabkan keterbatasan *resource* sehingga harus betul-betul memanfaatkan teknologi yang dimiliki. Dalam suatu teknologi jaringan diperlukan yang namanya manajemen jaringan yang fungsinya adalah untuk mengelola seluruh *resource* di jaringan agar bisa memberikan *good services* kepada penggunanya. Mengutip suatu definisi dari Mathews, D.C, bahwa proses suatu manajemen itu adalah "suatu proses yang ditujukan untuk merepresentasikan pengetahuan suatu organisasi kepada suatu langkah kongrit yang akan menghasilkan sesuatu yang diharapkan" (Kumar R, 2002). Oleh karena itu dibutuhkan startegi dan pengaturan yang tepat untuk mendapatkan

kehandalan jaringan dan bisa menjadi apa yang diinginkan oleh perusahaan. Yang tidak kalah penting, ketika desain dan proses manajemen jaringan selesai hendaknya diawasi oleh seorang penjaga gawang di jarinngan yang khusus bertugas untuk melakukan perawatan dan pengawasan terhadap aktifitas jaringan.

Seorang *administrator* jaringan bertanggung jawab penuh atas segala sesuatu ketersediaan dan kerahasiaan informasi. Tidak hanya itu, pemeliharaan perangkat keras maupun lunak, menganalisa masalah, memantau kerja jaringan agar bisa selalu tersedia bagi pengguna menjadi aktivitas keseharian dari seorang *administrator* jaringan (cyberfreeforum.com). Untuk itu tugas dari seorang *administrator* cukup berat, sehingga dibutuhkan sebuah sistem *scurity* yang bisa diandalkan untuk membantu kerja sang admin.

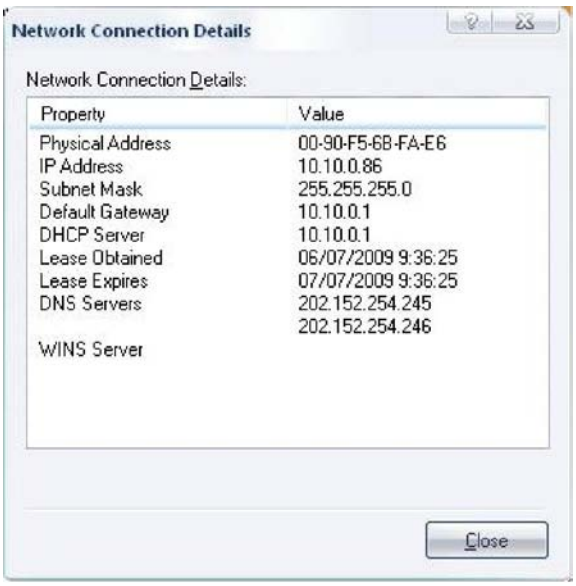
Taman Pintar (TamPin) merupakan instansi profit yang dibawahi oleh pemerintah kota Daerah Istimewa Yogyakarta (DIY) yang bergerak didalam dunia hiburan sekaligus pembelajaran. Perangkat-perangkat yang ada didalamnya berisi alat-alat sains mulai dari Sekolah dasar (SD) sampai Sekolah Menengah Atas (SMA). Sebagai instansi yang beregerak dibidang *profit oriented* maka Taman Pintar (TamPin) memperagai instansinya dengan sistem komputer yang sudah canggih. Rata-rata Komputer yang digunakan untuk melakukan proses bisnis disana menggunakan pentium IV 3.0 keatas. Untuk perangkat jaringan TamPin mempercayakan kehandalan jaringannya dengan menggunakan produk CISCO seperti Cisco Router 1800 series, swicht 2400 series 24 port dan access point CISCO.

Seperti yang diketahui bahwasannya produk CISCO dikenal dengan sistem manajemen dan keamanannya yang cukup tangguh, namun bagi para *maniac* dunia maya tidak ada yang tidak mungkin, tidak ada sistem yang benar-benar 100% aman karena setiap produk pasti mempunyai *bug* disaat pembuatan program, dan hal tersebut pada umumnya selalu diamati oleh produsen suatu produk dan apabila terjadi kesalahan maka segera di perbaiki. Oleh karenanya software yang digunakan harus sering di *update*, tujuannya adalah untuk memperbaiki *bug* atau kesalahan pada software tersebut. TamPin yang sudah memperagai organisasinya dengan produk yang cukup canggih namun hal tersebut masih sangat rawan untuk mendapat serangan dari luar. Apalagi TamPin sebagai layanan publik juga memberi layanan *free* host*spot* bagi para pengunjung. Dari beberapa pengamatan yang pernah dilakukan penulis, di TamPin hak akses ke jaringan dibuka secara bebas tanpa menggunakan autentikasi. Hal ini sangat rentan bagi keamanan jaringan. Topologi jaringan yang dimiliki masih sangat sederhana dan belum diberlakukan *policy* yang memandai untuk setiap area di seluruh lingkungan TamPin. IP jaringan di set *Dynamic* Host *Configuration Protocol* (DHCP) untuk seluruh kebutuhan jaringan. Untuk lebih jelasnya bisa dilihat dari topologi logik yang dimiliki TamPin :



Gambar 1.1 : Arsitektur Jaringan Taman Pintar

Dari topologi tersebut bisa dilihat bahwa jaringan taman pintar mendapat koneksi internet dari ISP yang kemudian dihubungkan ke CISCO router kemudian di set DHCP dengan IP 10.10.0.1 gateway 10.10.0.1 netmask 255.255.255.0. dari dari router kemudian terkoneksi dengan sebuah switch yang kemudian di *share* ke semua area.



Gambar 1.2 *:* Gambar Configurasi IP Addres Jaringan Taman Pintar

Melihat dari beberapa informasi diatas, jaringan TamPin masih jauh dari standart keamanan sehingga rentan terhadap serangan. Seorang penyusup jaringan atau lebih akrab dikenal sebagai *hacker* bisa dengan leluasa masuk jaringan hanya dengan beberapa teknik saja. Beberapa *access point* yang terpasang-pun hanya diberlakukan *password* standart wireless yang sering digunakan seperti *Wired Equivalent Privacy* (WEP) atau Wi – Fi Protected Access (WPA) yang merupakan autentikasi standart bawaan *access point*. Kedua jenis autentikasi tersebut mudah ditembus dengan teknik *ARP spoffing* yang kemudian bisa di *generate* untuk mendapatkan *password*. Dari beberapa contoh autentikasi tersebut dapat disimpulkan bahwa sistem keamanan jaringan TamPin masih sangat kurang, untuk itu penulis ingin mencoba membahas dalam penelitian skripsi ini untuk mengimplementasikan dan mengembangan sistem pendeteksi serangan pada jaringan yang sering dikenal *Intrusion Detection System* (IDS).

1.2 RUMUSAN MASALAH

Setelah melakukan penelusuran terhadap gambaran jaringan di lapangan, maka dari sekian banyak masalah ditemui mencoba di rumuskan dalam rumusan masalah sebagai berikut :

1. Bagaimana membangun sistem pengaman jaringan yang bisa mendeteksi serangan.
2. Bagimana mengembangkan sistem pengamanan jaringan yang responsif.
3. Bagaimana *administrator* bisa memantau jaringan secara *real time*.
4. Bagaimana membuat *rule-rule* untuk pengkategorian serangan.
5. Bagaimana mengirimkan pemberian tahuan adanya serangan melalui sms.

1.3 BATASAN MASALAH

Dari sekian banyak permasalahan yang telah dirumuskan, maka agar penelitian ini lebih fokus penulis membatasi permasalahan yang akan dibahas kepada :

1. Membangun sistem keamanan jaringan berbasis IDS menggunakan SNORT.
2. Mengembangkan Sistem Pendeteksi Serangan IDS dengan menggunakan SMS Gateway.

1.4 TUJUAN PENELITIAN

Penelitian tugas akhir ini bertujuan untuk membangun sistem keamanan jaringan yang handal dilingkungan Instansi Taman Pintar berbasis IDS dengan menggunakan SNORT. Penulis memilih Snort karena selain *open source* dan gratis, juga bisa ditambahkan aturan-aturan yang bisa disesuaikan dengan kebutuhan. Sistem IDS snort kali ini sedikit berbeda dengan yang sudah ada, karena penulis berencana menambahkan SMS Gateway sebagai sistem *alert* ketika ada serangan. Dengan *alert* tersebut seorang *administrator* akan menerima pesan sms dari *server* bahwa telah terjadi serangan. Sehingga nantinya dengan peringatan tersebut admin bisa mengambil tindakan selanjutnya. Ini akan sangat membantu admin untuk bisa memantau jaringan secara real time tanpa harus *standby* didepan komputer. Selain itu tujuan mahasiswa yang sedang melakukan penelitian ini dapat melakukan konfigurasi *server* dengan menggunakan ubuntu, apache, Mysql, SNORT, SMSGateway yang akan dijadikan pemantau jaringan.

1.5 MANFAAT PENELITIAN

Pemanfaatan Snort kini banyak digunakan oleh instansi atau perusahaan baik itu skala besar maupun kecil untuk membantu mengamankan sistem jaringan di lingkungan instansi masing-masing. Untuk itu dengan adanya penelitian tugas akhir ini semoga bisa membantu mengamankan jaringan TamPin dan membantu admin dalam memantau keamanan secara *real time*. Selain itu hasil dari pembahasan penelitian tugas akhir ini nantinya semoga bisa menjadi rujukan atau referensi oleh siapa saja yang nantinya ingin membuat sistem pengamanan dalam pengelolaan sebuah jaringan.

Secara lebih detail Penelitian tugas akhir ini diharapkan dapat bermanfaat bagi TamPin antara lain :

1. Mesin IDS yang dibangun bisa membantu mengamankan jaringan TamPin.
2. Mempermudah pekerjaan admin dalam menjaga keamanann jaringan.
3. Mencegah penyusup yang hendak menerobos masuk sistem.
4. Administrator jaringan TamPin bisa lebih mudah dalam mengaudit jaringan.
5. Mesin snort yang integrasi dengan sms gateway bisa lebih responsif dalam mengawasi jaringan sehingga ketika terjadi usaha-usaha mencurigakan bisa langsung diberitahukan lewat sms.
6. Pemberitahuan lewat sms bisa membantu admin dalam memantau jaringan secara *real time.*

1.6 KEASLIAN PENELITIAN

Sistem pengamanan jaringan yang berbasis IDS menggunakan SNORT dan SMSGateway sebelumnya belum pernah dilakukan di instansi TamPin. Namun pemanfaatan SNORT dalam pengamanan jaringan sudah pernah dilakukan ditempat lain dan studi kasus yang berbeda-beda. Berdasarkan hasil penelusuran, kajian tentang IDS pernah dilakukan oleh mahasiswa AKPRIND Yogyakarta, UBINUS Jakarta, dan Universitas Tarumanegara.

BAB V

PENUTUP

5.1 KESIMPULAN

Dalam penelitian tugas akhir ini yang mengambil judul "Sistem Pendeteksi Serangan Pada Jaringan Komputer Menggunakan Snort Berbasis Sms Gateway (*Studi Kasus di Taman Pintar Yogyakarta*)" bahwa dapat ditarik kesimpulan:

1. Setelah melakukan penelitian maka mesin pendeteksi serangan menggunakan snort berhasil dibuat.
2. Mesin snort sudah bisa bekerja sesuai fungsinya yaitu bisa mendeteksi jenis-jenis paket yang membahayakan dan menyimpannya kedalam database.
3. Dari data snort yang disimpan di dalam database, acidbase yang dipergunakan untuk menganilis data sudah bisa dipergunakan sesuai fungsinya.
4. Sms gateway gammu yang diintegrasikan dengan mesin snort, sudah berhasil mengirimkan alert kepada administrator sebagai pemberitahuan.
5. Mesin snort, acidbase, dan sms gateway gammu menjadi satu kesatuan sebagai mesin pendeteksi serangan yang lebih responsif dan memberikan report secara *real time*.

6. Penulis telah dapat mengimplementasikan mesin snort sebagai pendeteksi serangan berbasis sms gateway sebagai alat pengaman jaringan di instansi taman pintar yogyakarta.

5.2 SARAN

Sistem pendeteksi serangan menggunakan snort ini tidak terlepas dari kekurangan dan kelemahan, terutama mesin snort sendiri harus mengupdate rule-rule yang mana setiap saat usaha orang untuk menyusup dan merusak sistem akan terus berkembang. Untuk itu agar sistem pengamanan ini dapat bekerja lebih optimal, peneliti menyarankan beberapa hal, antara lain:

1. Diharapkan pengembangan mesin pendeteksi serangan ini dengan menambahkan rule-rule baru atau metode baru yang bisa diterapkan pada mesin snort.
2. Pada penelitian selanjutnya ada beberapa hal yang perlu diperbaiki yaitu mengenai integrasi data dengan database yang lain seperti oracle, postgree dan database lainya.
3. Pengembangan audit menggunakan web interface acidbase bisa dibuat lebih sederhana tanpa mengurangi fungsi-fungsi didalam acidbase, semisal membuat acidbase versi mobile sehingga audit database snort bisa dilakukan dengan handphone.
4. Pada sistem sms gammu pada penelitian berikutnya bisa ditambahkan menajemen user dan manajemen kontak dan bisa dibuat user interface-nya.

DAFTAR PUSTAKA

Ambang Utomo, Prasetya. 2006. **"**Membangun Aplikasi SMS dengan Paket *Open Source* **".** Penerbit Andi.

Ariyus, Dony. 2007 *"Intrusion Detection System"*. Penerbit Andi Yogyakarta.

Cafaro, Massimo. 2011 *"Grids, Clouds, and Virtualization"*. Springer London Dordrecht Heidelberg New York.

Cihar, Michal. 2011. "Gammu Manual" http://wammu.eu/gammu/

Endorf, Carl. 2004 *"Intrusion Detection & Prevention"* McGraw-Hill.

Cooper, Peter. 2010. *"How to Install Ruby 1.9.2 and Rails 3.0 on Ubuntu 10.10"* http://www.rubyinside.com/how-to-install-ruby-1-9-2-and-rails-3-0-on-ubuntu-10-10-4148.html

Fadli Rosyad, Arief. 2010. *"Sistem Pendeteksi Penyusup Pada Jaringan Komputer Menggunakan IDS (Intrusion Detection System)*". Jurusan Teknik Komputer, Fakultas Teknik dan Ilmu UNIKOM Bandung.

Grinsing, Ki. *"Memahami Firewall DMZ*".www.sysneta.com/memahami-firewall-dmz

Howlett, Tony. 2005 *"Open Source Security Tools"* Prentice Hall Professional Technical Reference Upper Saddle River, New Jersey 07458.

Komputer Wahana. 2006. **"***Seri Panduan Lengkap Menguasai Pemograman Web dengan PHP 5***".** Penerbit Andi.

Maulana, Irpan. 2006. "*Pengujian pengembangan Aplikasi Bank" FSILKOM UI* http://www.digilib.ui.ac.id/file?file=digital/122533-SP-117-Pengembangan%20Aplikasi-Metodologi

NAI, Kurniawan. 2010. *"Sistem Deteksi dan Penanganan Intrusi Menggunakan Snort dan Base*". Institut Sains & Teknologi AKPRIND, Yogyakarta.

Northcutt, Stephen. 2004. *"Snort 2.1 Intrusion Detection, Second Edition"* Syngress Publishing, Inc

Ortega, Alberto. 2010 *"PenTBox 1.4"* http://www.pentbox.net/

Pardosi, Mico. 1997. *"Kamus Komputer Standart"*. Penerbit Indah Surabaya.

Provos, Niels. Thorsten Holz. 2007. *"Virtual Honeypots: From Botnet Tracking to Intrusion Detection"* Addison Wesley Professional.

Ray Paradeep, Kumar. 2002. *"Cooperative Management of Enterprice Networks***".** Kluwer Academic.

Tahir, Ahmad. 2010. "*Cyber Crime (Akar Masalah, Solusi, dan Penanggulangannya***)"** Penerbit Suka Press.

Threestayanti, Liana. 2008. *"Network Administrator (profesi yang menjanjikan)*".http://d3tkjuntad.cyberfreeforum.com/t260-network-*administrator*-profesi-yang-menjanjikan.

Toxen, box. 2000. *"Real World Linux® Security: Intrusion Prevention, Detection, and Recovery"* Prentice Hall PTR

Ur Rahman, Rafeeq. 2003. *"Intrusion detection System With SNORT*". Prentice Hall Upper Saddle River New Jersey 07458.

Zam Kerinci, Efvy. 2010. *"Hacking is Very Easy"*. Penerbit connexi yogyakarta

LAMPIRAN A
KODE SUMBER (SOURCE CODE) MESIN SNORT

Nama File : snort.conf

```
#--------------------------------
------------------

#            http://www.snort.org
Snort 2.8.5.2 Ruleset

#               Contact:   snort-
sigs@lists.sourceforge.net

#--------------------------------
------------------
# $Id$
#
###################################
##################
# This file contains a sample
snort configuration.
# You can take the following
steps to create your own custom
configuration:
#
#  1) Set the variables for your
network
#   2) Configure dynamic loaded
libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#   5) Add any runtime config
directives
#  6) Customize your rule set
#
###################################
##################
#  Step  #1:  Set  the  network
variables:
# You must change the following
variables to reflect your local
network. The
# variable is currently setup for
an RFC 1918 address space.
#
# You can specify it explicitly
as:
#
# var HOME_NET 10.1.1.0/24
# if Snort is built with IPv6
support enabled (--enable-ipv6),
use:
#
# ipvar HOME_NET 10.1.1.0/24
#   or   use   global   variable
$<interfacename>_ADDRESS    which
will be always
# initialized to IP address and
netmask of the network interface
which you run
# snort at.  Under Windows, this
must be specified as
#     $(<interfacename>_ADDRESS),
such as:
#     $(\Device\Packet_{12345678-
90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP
addresses for HOME_NET
# by separating the IPs with
commas like this:
#
#          var          HOME_NET
[10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY
SPACES IN YOUR LIST!
#
# or you can specify the variable
to be any IP address
# like this:

var HOME_NET 192.168.1.0/24

# Set up the external network
addresses as well.  A good start
may be "any"

var EXTERNAL_NET !$HOME_NET

#var EXTERNAL_NET !$HOME_NET
# Configure your server lists.
This allows snort to only look
for attacks to
# systems that have a service up.
Why look for HTTP attacks if you
are not
# running a web server?  This
allows quick filtering based on
IP addresses
#   These configurations MUST
follow the same configuration
scheme as defined
# above for $HOME_NET.
# List of DNS servers on your
network
```

```
var DNS_SERVERS $HOME_NET


# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# List of telnet servers on your network
var FTP_SERVERS $HOME_NET

# List of snmp servers on your network
var SNMP_SERVERS $HOME_NET

# Configure your service ports. This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on. For
# example, if you run a web server on port 8180, set your HTTP_PORTS variable
# like this:
#
# portvar HTTP_PORTS 8180
#
# Ports you run web servers on

portvar HTTP_PORTS 80

# NOTE:  If you wish to define multiple HTTP ports, use the portvar
# syntax to represent lists of ports and port ranges. Examples:
## portvar HTTP_PORTS [80,8080]
##       portvar       HTTP_PORTS [80,8000:8080]
# And only include the rule that uses $HTTP_PORTS once.
#
#  The pre-2.8.0 approach of redefining the variable to a different port and
# including the rules file twice is      obsolete.            See README.variables for more
# details.
#  Ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

#  Ports you might see oracle attacks on
portvar ORACLE_PORTS 1521

# Ports for FTP servers
portvar FTP_PORTS 21

# other variables
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
var                    AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var              PREPROC_RULE_PATH /etc/snort/preproc_rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
#  truncation  or  options  of unusual length or infrequently used tcp options
# Stop generic decode events:
# config disable_decode_alerts
# Stop Alerts on experimental TCP options
#                            config disable_tcpopt_experimental_alerts
# Stop Alerts on obsolete TCP options
#                            config disable_tcpopt_obsolete_alerts
# Stop Alerts on T/TCP alerts
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
#  behavior for your network, disable the next option.
```

```
#                           config
disable_tcpopt_ttcp_alerts
#  Stop  Alerts  on  all  other
TCPOption type events:
# config disable_tcpopt_alerts
#  Stop  Alerts  on  invalid  ip
options
# config disable_ipopt_alerts
# Alert if value in length field
(IP, TCP, UDP) is greater than
the
# actual length of the captured
portion of the packet that the
length
# is supposed to represent:
#                           config
enable_decode_oversized_alerts
# Same as above, but drop packet
if in Inline mode -
#  enable_decode_oversized_alerts
must be enabled for this to work:
#                           config
enable_decode_oversized_drops
# Configure the detection engine

# ==============================
# Use a different pattern matcher
in case you have a machine with
very limited
# resources:
#
# config detection: search-method
lowmem

# Configure Inline Resets
# ======================
# If running an iptables firewall
with snort in InlineMode() we can
now
# perform resets via a physical
device. We grab the indev from
iptables
# and use this for the interface
on  which  to  send  resets.  This
config
# option takes an argument for
the src mac address you want to
use in the
# reset packet.   This way the
bridge can remain stealthy. If
the src mac
# option is not set we use the
mac address of the indev device.
If we
# don't set this option we will
default to sending resets via raw
socket,
# which needs an ipaddress to be
assigned to the int.
#
#      config      layer2resets:
00:06:76:DD:5F:E3

##################################
##################

# Step #2: Configure dynamic
loaded libraries
# If snort was configured to use
dynamically loaded libraries,
# those libraries can be loaded
here.
#
#   Each   of   the   following
configuration options can be done
via
# the command line as well.
#
# Load all dynamic preprocessors
from the install path
# (same as command line option --
dynamic-preprocessor-lib-dir)
#

dynamicpreprocessor     directory
/usr/lib/snort_dynamicpreprocesso
r/
#
#   Load   a   specific   dynamic
preprocessor   library   from   the
install path
# (same as command line option --
dynamic-preprocessor-lib)
#
#    dynamicpreprocessor    file
/usr/lib/snort_dynamicpreprocesso
r/libdynamicexample.so
# Load a dynamic engine from the
install path
# (same as command line option --
dynamic-engine-lib)

dynamicengine
/usr/lib/snort_dynamicengine/libs
f_engine.so

#
#   Load   all   dynamic   rules
libraries from the install path
# (same as command line option --
dynamic-detection-lib-dir)
#
#   dynamicdetection   directory
/usr/lib/snort_dynamicrules/
#
# Load a specific dynamic rule
library from the install path
# (same as command line option --
dynamic-detection-lib)
#
#      dynamicdetection      file
/usr/lib/snort_dynamicrule/libdyn
amicexamplerule.so
#

##################################
```

```
##################
#     Step     #3:     Configure
preprocessors
#   General   configuration   for
preprocessors is of
# the form
#                    preprocessor
<name_of_processor>:
<configuration_options>

#    frag3:    Target-based    IP
defragmentation
# ------------------------------
-------
#
#  Frag3  is  a  brand  new  IP
defragmentation preprocessor that
is capable of
#    performing    "target-based"
processing   of   IP   fragments.
Check out the
# README.frag3 file in the doc
directory for more background and
configuration
# information.
#
# Frag3 configuration is a two
step     process,     a     global
initialization phase
# followed by the definition of a
set of defragmentation engines.
#
#  Global  configuration  defines
the number of fragmented packets
that Snort can
# track at the same time and
gives you options regarding the
memory cap for the
#   subsystem   or,   optionally,
allows you to preallocate all the
memory for the
# entire frag3 system.
#
# frag3_global options:
#    max_frags: Maximum number of
frag trackers that may be active
at once.
#                Default value is
8192.
#     memcap: Maximum amount of
memory that frag3 may access at
any given time.
#           Default value is 4MB.
#       prealloc_frags:  Maximum
number of individual fragments
that may be processed
#                         at once.
This is instead of the memcap
system, uses static
#                   allocation to
increase performance.  No default
value.  Each
#                    preallocated
fragment typically eats ~1550
bytes.  However,
#                         the exact
amount is determined by the
snaplen, and this can
#                   go as high as
64K so beware!
#
#   Target-based   behavior   is
attached to an engine as a
"policy" for handling
# overlaps and retransmissions as
enumerated in the Paxson paper.
There are
# currently five policy types
available:  "BSD",  "BSD-right",
"First", "Linux"
# and "Last".  Engines can be
bound to standard Snort CIDR
blocks or
# IP lists.
#
# frag3_engine options:
#     timeout: Amount of time a
fragmented packet may be active
before expiring.
#             Default value is 60
seconds.
#     ttl_limit: Limit of delta
allowable for TTLs of packets in
the fragments.
#                    Based on the
initial received fragment TTL.
#    min_ttl: Minimum acceptable
TTL for a fragment, frags with
TTLs below this
#                   value will be
discarded.  Default value is 0.
#     detect_anomalies: Activates
frag3's     anomaly     detection
mechanisms.
#    policy: Target-based policy
to   assign   to   this   engine.
Default is BSD.
#    bind_to: IP address set to
bind this engine to.  Default is
all hosts.
#


# Frag3 configuration example:
#preprocessor       frag3_global:
max_frags  65536,  prealloc_frags
65536
#preprocessor       frag3_engine:
policy linux \
#
bind_to
[10.1.1.12/32,10.1.1.13/32] \
#
detect_anomalies
#preprocessor       frag3_engine:
policy first \
```

```
#
bind_to 10.2.1.0/24 \
#
detect_anomalies
#preprocessor       frag3_engine:
policy last \
#
bind_to 10.3.1.0/24
#preprocessor       frag3_engine:
policy bsd

preprocessor        frag3_global:
max_frags 65536
preprocessor frag3_engine: policy
first            detect_anomalies
overlap_limit 10

# stream5: Target Based stateful
inspection/stream reassembly for
Snort
# ------------------------------
---------------------------------
-----
#   Stream5   is   a   target-based
stream  engine  for  Snort.    It
handles both
# TCP and UDP connection tracking
as well as TCP reassembly.
#
# See README.stream5 for details
on the configuration options.
#
# Example config

preprocessor      stream5_global:
max_tcp 8192, track_tcp yes, \

track_udp no
preprocessor stream5_tcp: policy
first

# Not recommended in production
systems
#    preprocessor    stream5_tcp:
policy                    first,
use_static_footprint_sizes
#    preprocessor    stream5_udp:
ignore_any_rules

# Performance Statistics
# ----------------------
#   Documentation   for   this   is
provided  in  the  Snort  Manual.
You should read it.
# It is included in the release
distribution                   as
doc/snort_manual.pdf
#
# preprocessor perfmonitor: time
300  file  /var/snort/snort.stats
pktcnt 10000
#   http_inspect:   normalize   and
detect HTTP traffic and protocol
anomalies
#
# lots of options available here.
See doc/README.http_inspect.
# unicode.map should be wherever
your snort.conf lives, or given
# a full path to where snort can
find it.

preprocessor http_inspect: global
\
    iis_unicode_map   unicode.map
1252
preprocessor http_inspect_server:
server default \
    profile all ports { 80 8080
8180 } oversize_dir_length 500
#
#       Example    unique    server
configuration
#
#preprocessor
http_inspect_server:       server
1.1.1.1 \
#    ports { 80 3128 8080 } \
#    server_flow_depth 0 \
#    ascii no \
#    double_decode yes \
#    non_rfc_char { 0x00 } \
#    chunk_length 500000 \
#    non_strict \
#    oversize_dir_length 300 \
#    no_alerts

#   rpc_decode:   normalize   RPC
traffic
# ------------------------------
--
# RPC may be sent in alternate
encodings besides the usual 4-
byte encoding
# that is used by default. This
plugin  takes  the  port  numbers
that RPC

# services are running on as
arguments - it is assumed that
the given ports
# are actually running this type
of service. If not, change the
ports or turn
# it off.
#  The RPC decode preprocessor
uses generator ID 106
# arguments: space separated list
# alert_fragments - alert on any
rpc fragmented TCP data
#   no_alert_multiple_requests   -
don't alert when >1 rpc query is
in a packet
#    no_alert_large_fragments    -
don't alert when the fragmented
#
```

```
sizes exceed the current packet size
#  no_alert_incomplete  -  don't alert when a single segment
#                          exceeds the current packet size

preprocessor   rpc_decode:   111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.
#
# arguments:
#   syntax:
#      preprocessor bo: noalert { client  |  server  |  general  | snort_attack } \
#                       drop    { client  |  server  |  general  | snort_attack }
#   example:
#      preprocessor bo: noalert { general   server   }   drop   { snort_attack }
#
#
# The Back Orifice detector uses Generator ID 105 and uses the
# following SIDS for that GID:
#  SID     Event description
# -----   ------------------
#   1        Back Orifice traffic detected
#   2         Back Orifice Client Traffic Detected
#   3         Back Orifice Server Traffic Detected
#    4         Back Orifice Snort Buffer Attack

preprocessor bo

#  ftp_telnet:  FTP  &  Telnet normalizer, protocol enforcement and buff overflow
# ---------------------------------------------------------------------------
#  This preprocessor normalizes telnet negotiation strings from telnet and
# ftp traffic.   It looks for traffic that breaks the normal data stream
# of the protocol, replacing it with a normalized representation of that
# traffic so that the "content" pattern matching keyword can work without
# requiring modifications.
#
#  It  also  performs  protocol correctness checks for the FTP command channel,
# and identifies open FTP data transfers.
#
# FTPTelnet has numerous options available, please read
#   README.ftptelnet   for   help configuring the options for the global
# telnet, ftp server, and ftp client sections for the protocol.

#####
# Per Step #2, set the following to     load     the     ftptelnet preprocessor
# dynamicpreprocessor file <full path                          to libsf_ftptelnet_preproc.so>
# or use commandline option
#      --dynamic-preprocessor-lib <full           path           to libsf_ftptelnet_preproc.so>

preprocessor ftp_telnet: global \
   encrypted_traffic yes \
   inspection_type stateful
preprocessor ftp_telnet_protocol: telnet \
   normalize \
   ayt_attack_thresh 200



# This is consistent with the FTP rules as of 18 Sept 2004.
# CWD can have param length of 200
# MODE has an additional mode of Z (compressed)
#  Check for string formats in USER & PASS commands
# Check nDTM commands that set modification time on the file.

preprocessor ftp_telnet_protocol: ftp server default \
   def_max_param_len 100 \
   alt_max_param_len 200 { CWD } \
   cmd_validity MODE < char ASBCZ > \
   cmd_validity  MDTM  <  [  date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
   chk_str_fmt  {  USER  PASS  RNFR RNTO SITE MKD } \
   telnet_cmds yes \
   data_chan
```

```
preprocessor ftp_telnet_protocol:
ftp client default \
   max_resp_len 256 \
   bounce yes \
   telnet_cmds yes

# smtp: SMTP normalizer, protocol
enforcement and buffer overflow
# ------------------------------
---------------------------------
-----------
# This preprocessor normalizes
SMTP commands by removing
extraneous spaces.
# It looks for overly long
command lines, response lines,
and data header lines.
# It can alert on invalid
commands, or specific valid
commands. It can optionally
# ignore mail data, and can
ignore TLS encrypted data.
#
# SMTP has numerous options
available, please read
README.SMTP for help
# configuring options.

#####

# Per Step #2, set the following
to load the smtp preprocessor
# dynamicpreprocessor file <full
path to libsf_smtp_preproc.so>
# or use commandline option
#      --dynamic-preprocessor-lib
<full path to
libsf_smtp_preproc.so>

preprocessor smtp: \
  ports { 25 587 691 } \
  inspection_type stateful \
  normalize cmds \
  normalize_cmds { EXPN VRFY RCPT
} \
  alt_max_command_line_len 260 {
MAIL } \
  alt_max_command_line_len 300 {
RCPT } \
  alt_max_command_line_len 500 {
HELP HELO ETRN } \
  alt_max_command_line_len 255 {
EXPN VRFY }

# sfPortscan
# ----------
# Portscan detection module.
Detects various types of
portscans and
# portsweeps. For more
information on detection
philosophy, alert types,
# and detailed portscan
information, please refer to the
README.sfportscan.
#
# -configuration options-
#     proto { tcp udp icmp ip all
}
#          The arguments to the
proto option are the types of
protocol scans that
#       the user wants to detect.
Arguments should be separated by
spaces and
#       not commas.
#          scan_type { portscan
portsweep decoy_portscan
distributed_portscan all }
#          The arguments to the
scan_type option are the scan
types that the
#          user wants to detect.
Arguments should be separated by
spaces and not
#       commas.
#                sense_level {
low|medium|high }
#             There is only one
argument to this option and it is
the level of
#        sensitivity in which to
detect portscans. The 'low'
sensitivity

#          detects scans by the
common method of looking for
response errors, such
#           as TCP RSTs or ICMP
unreachables. This level
requires the least
#          tuning. The 'medium'
sensitivity level detects
portscans and
#             filtered portscans
(portscans that receive no
response). This
#       sensitivity level usually
requires tuning out scan events
from NATed
#        IPs, DNS cache servers,
etc. The 'high' sensitivity
level has
#          lower thresholds for
portscan detection and a longer
time window than
#        the 'medium' sensitivity
level. Requires more tuning and
may be noisy
#        on very active networks.
However, this sensitivity levels
catches the
#       most scans.
#     memcap { positive integer }
#          The maximum number of
bytes to allocate for portscan
```

```
detection.  The
#         higher this number the
more nodes that can be tracked.
#     logfile { filename }
#       This option specifies the
file to log portscan and detailed
portscan
#        values to.  If there is
not a leading /, then snort logs
to the
#       configured log directory.
Refer  to  README.sfportscan  for
details on
#        the logged values in the
logfile.
#     watch_ip { Snort IP List }
#      ignore_scanners { Snort IP
List }
#       ignore_scanned { Snort IP
List }
#           These options take a
snort  IP  list  as  the  argument.
The 'watch_ip'
#           option specifies the
IP(s) to watch for portscan.  The
#        'ignore_scanners' option
specifies the IP(s) to ignore as
scanners.
#       Note that these hosts are
still  watched  as  scanned  hosts.
The
#        'ignore_scanners' option
is used to tune alerts from very
active
#       hosts such as NAT, nessus
hosts, etc.  The 'ignore_scanned'
option
#         specifies the IP(s) to
ignore  as  scanned  hosts.   Note
that these hosts
#          are still watched as
scanner      hosts.           The
'ignore_scanned' option is
#        used to tune alerts from
very active hosts such as syslog
servers, etc.
#     detect_ack_scans
#        This option will include
sessions  picked  up  in  midstream
by the stream
#              module,  which  is
necessary  to  detect  ACK  scans.
However, this can lead to
#        false alerts, especially
under  heavy  load  with  dropped
packets; which is why
#          the  option  is  off  by
default.
#

preprocessor sfportscan: proto  {
all } \
                         memcap {
10000000 } \

sense_level { low }

# arpspoof
#-------------------------------
--------
# Experimental ARP detection code
from  Jeff  Nathan,  detects  ARP
attacks,
#  unicast  ARP  requests,  and
specific ARP mapping monitoring.
To make use of
#  this  preprocessor  you  must
specify  the  IP  and  hardware
address of hosts on
#  the  same  layer  2  segment  as
you.   Specify  one  host  IP  MAC
combo per line.
# Also takes a "-unicast" option
to  turn  on  unicast  ARP  request
detection.
# Arpspoof uses Generator ID 112
and  uses  the  following  SIDS  for
that GID:

#  SID     Event description
# -----   -------------------
#   1       Unicast ARP request
#     2           Etherframe  ARP
mismatch (src)
#     3           Etherframe  ARP
mismatch (dst)
#    4        ARP cache overwrite
attack
#preprocessor arpspoof
#preprocessor
arpspoof_detect_host:
192.168.40.1 f0:0f:00:f0:0f:00

# ssh
# ------------------------------
#  The  SSH  preprocessor  detects
the      following      exploits:
Challenge-Response
# Authentication overflow, CRC 32
overflow,   Secure   CRT   version
string overflow,
#    and    protocol    version
mismatches.

#
#  Both  Challenge-Response  Auth
and  CRC  32  attacks  occur  after
the key exchange,
#  and  are  therefore  encrypted.
Both attacks involve sending a
large payload
#    (20kb+)    to    the    server
immediately       after       the
authentication challenge.
# To detect the attacks, the SSH
preprocessor counts the number of
```

```
bytes
# transmitted to the server.  If
those bytes exceed a pre-defined
limit,
#    set    by    the    option
"max_client_bytes", an alert is
generated. Since
#  the  Challenge-Response  Auth
overflow  only  affects  SSHv2,
while CRC 32 only
# affects SSHv1, the SSH version
string  exchange  is  used  to
distinguish
# the attacks.
#
# The Secure CRT and protocol
mismatch exploits are observable
before
# the key exchange.
#
#  SSH  has  numerous  options
available, please read README.ssh
for help
# configuring options.

#####
# Per Step #2, set the following
to load the ssh preprocessor
# dynamicpreprocessor file <full
path to libsf_ssh_preproc.so>
# or use commandline option
#      --dynamic-preprocessor-lib
<full           path           to
libsf_ssh_preproc.so>
#

preprocessor ssh: server_ports {
22 } \

max_client_bytes 19600 \

max_encrypted_packets 20 \

enable_respoverflow
enable_ssh1crc32 \

enable_srvoverflow
enable_protomismatch

# DCE/RPC
#-------------------------------
--------
#
# The dcerpc preprocessor detects
and  decodes  SMB  and  DCE/RPC
traffic.
# It is primarily interested in
DCE/RPC  data,  and  only  decodes
SMB
#  to  get  at  the  DCE/RPC  data
carried by the SMB layer.
#
#  Currently,  the  preprocessor
only  handles  reassembly  of
fragmentation
#  at  both  the  SMB  and  DCE/RPC
layer.  Snort rules can be evaded
by
#    using    both    types    of
fragmentation;      with      the
preprocessor enabled
# the rules are given a buffer
with a reassembled SMB or DCE/RPC
# packet to examine.
#
#  At  the  SMB  layer,  only
fragmentation using WriteAndX is
currently
#  reassembled.    Other  methods
will  be  handled  in  future
versions of
# the preprocessor.
#
# Autodetection of SMB is done by
looking  for  "\xFFSMB"  at  the
start of
#  the  SMB  data,  as  well  as
checking  the  NetBIOS  header
(which is always
# present for SMB) for the type
"SMB Session".
#
# Autodetection of DCE/RPC is not
as  reliable.    Currently,  two
bytes are
#  checked  in  the  packet.
Assuming  that  the  data  is  a
DCE/RPC header,
# one byte is checked for DCE/RPC
version (5) and another for the
type
#  "DCE/RPC  Request".    If  both
match, the preprocessor proceeds
with that
# assumption that it is looking
at DCE/RPC data.  If subsequent
checks
#  are  nonsensical,  it  ends
processing.
#
#  DCERPC  has  numerous  options
available,      please      read
README.dcerpc for help
# configuring options.

#####

# Per Step #2, set the following
to load the dcerpc preprocessor
# dynamicpreprocessor file <full
path to libsf_dcerpc_preproc.so>
# or use commandline option
#      --dynamic-preprocessor-lib
<full           path           to
libsf_dcerpc_preproc.so>
#
```

```
#preprocessor dcerpc: \
#    autodetect \
#    max_frag_size 3000 \
#    memcap 100000
# DCE/RPC 2
#--------------------------------
--------
#   See   doc/README.dcerpc2   for
explanations of what the
# preprocessor does and how to
configure it.
#

preprocessor dcerpc2
preprocessor      dcerpc2_server:
default

# DNS
#--------------------------------
--------
#     The     dns     preprocessor
(currently) decodes DNS Response
traffic
#     and     detects     a     few
vulnerabilities.
#
#   DNS   has   a   few   options
available, please read README.dns
for
# help configuring options.

#####

# Per Step #2, set the following
to load the dns preprocessor
# dynamicpreprocessor file <full
path to libsf_dns_preproc.so>
# or use commandline option
#      --dynamic-preprocessor-lib
<full           path           to
libsf_dns_preproc.so>

preprocessor dns: \
    ports { 53 } \
    enable_rdata_overflow

# SSL
#--------------------------------
--------
# Encrypted traffic should be
ignored   by   Snort   for   both
performance reasons
# and to reduce false positives.
The   SSL   Dynamic   Preprocessor
(SSLPP)
#   inspects   SSL   traffic   and
optionally determines if and when
to stop
# inspection of it.
#
# Typically, SSL is used over
port 443 as HTTPS.   By enabling
the SSLPP to
# inspect port 443, only the SSL
handshake of each connection will
be
# inspected.  Once the traffic is
determined to be encrypted, no
further
# inspection of the data on the
connection is made.
#
# If you don't necessarily trust
all of the SSL capable servers on
your
# network, you should remove the
"trustservers" option from the
configuration.
#
#       Important note: Stream5
should be explicitly told to
reassemble
#                       traffic on
the ports that you intend to
inspect SSL
#                        encrypted
traffic on.
#
#   To add reassembly on port 443
to Stream5, use 'port both 443'
in the
#   Stream5 configuration.

preprocessor                  ssl:
noinspect_encrypted, trustservers

##################################
##################################
##

#   Step   #4:   Configure   output
plugins
#
# Uncomment and configure the
output plugins you decide to use.
General
#   configuration   for   output
plugins is of the form:
#
#    output    <name_of_plugin>:
<configuration_options>
#
# alert_syslog: log alerts to
syslog
# --------------------------------
---
#   Use   one   or   more   syslog
facilities as arguments.   Win32
can also optionally
#    specify    a    particular
hostname/port.  Under Win32, the
default hostname is
#  '127.0.0.1', and the default
port is 514.
#
# [Unix flavours should use this
```

```
format...]
# output alert_syslog: LOG_AUTH
LOG_ALERT
#
# [Win32 can use any of these
formats...]
# output alert_syslog: LOG_AUTH
LOG_ALERT
#       output       alert_syslog:
host=hostname, LOG_AUTH LOG_ALERT
#       output       alert_syslog:
host=hostname:port,       LOG_AUTH
LOG_ALERT

# log_tcpdump: log packets in
binary tcpdump format
# ------------------------------
------------------
# The only argument is the output
file name.
#

output log_tcpdump: tcpdump.log

# database: log to a variety of
databases
# ------------------------------
--------
# See the README.database file
for more information about
configuring
# and using this plugin.
#
# output database: log, mysql,
user=root password=test dbname=db
host=localhost
#    output    database:    alert,
postgresql,            user=snort
dbname=snort

  output database: log, mysql,
user=root        password=rahasia
dbname=snort host=localhost

# output database: log, odbc,
user=snort dbname=snort
# output database: log, mssql,
dbname=snort           user=snort
password=test
# output database: log, oracle,
dbname=snort           user=snort
password=test
# <debian>
# Keep your paws off of these
(#DBSTART#) and (#DBEND#) tokens
#   or   you   *will*   break   the
configure     process     (snort-
pgsql/snort-mysql only)
# Anything you put between them
will be removed on (re)configure.
#
# (#DBSTART#)
# (#DBEND#)
#
# </debian>
#

# unified: Snort unified binary
format alerting and logging
# ------------------------------
------------------------------
#  The  unified  output  plugin
provides  two  new  formats  for
logging and generating
#   alerts   from   Snort,   the
"unified"  format.   The  unified
format is a straight
# binary format for logging data
out of Snort that is designed to
be fast and
# efficient.  Used with barnyard
(the  new  alert/log  processor),
most of the
#   overhead   for   logging   and
alerting to various slow storage
mechanisms such as
# databases or the network can
now be avoided.
#
#  Check  out  the  spo_unified.h
file for the data formats.
#
# Two arguments are supported.
#     filename - base filename to
write  to  (current  time_t  is
appended)
#     limit     - maximum size of
spool file in MB (default: 128)
#
# output alert_unified: filename
snort.alert, limit 128
# output log_unified: filename
snort.log, limit 128
# prelude: log to the Prelude
Hybrid IDS system
# ------------------------------
-------------
#
# profile = Name of the Prelude
profile  to  use  (default  is
snort).
#
#   Snort   priority   to   IDMEF
severity mappings:
# high < medium < low < info
#
# These are the default mapped
from classification.config:
# info   = 4
# low    = 3
# medium = 2
# high   = anything below medium
#
# output alert_prelude
#      output      alert_prelude:
profile=snort-profile-name
```

```
# You can optionally define new rule types and associate one or more output
# plugins specifically to that type.
#
# This example will create a type that will log to just tcpdump.
# ruletype suspicious
# {
#   type log
#           output    log_tcpdump: suspicious.log
# }
#
#  EXAMPLE  RULE  FOR  SUSPICIOUS RULETYPE:
# suspicious tcp $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC Server";)
#
# This example will create a rule type that will log to syslog and a mysql
# database:
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#    output database: log, mysql, user=snort           dbname=snort host=localhost
# }
#
#  EXAMPLE  RULE  FOR  REDALERT RULETYPE:
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)
#
#   Include   classification   &  priority settings
# Note for Windows users:  You are advised to make this an absolute path,
#            such            as: c:\snort\etc\classification.config

include classification.config

# Include reference systems
# Note for Windows users:  You are advised to make this an absolute path,
#            such            as: c:\snort\etc\reference.config

include reference.config

##################################
```

```
##################################
##

# Step #5: Configure snort with config statements
# See the snort manual for a full set of configuration references
#
# config flowbits_size: 64
#
# New global ignore_ports config option from Andy Mullican
#
# config ignore_ports: <tcp|udp> <list of ports separated by whitespace>
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53

##################################
##################################
##
# Step #6: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
#   The   snort   web   site   has documentation about how to write your own custom snort
# rules.
#================================
=========
# Include all relevant rulesets here
#
#  The following rulesets are disabled by default:
#
#        web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
#   Please   read   the   specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#================================
=========

include $RULE_PATH/local.rules
```

```
include           $RULE_PATH/bad-
traffic.rules
include $RULE_PATH/exploit.rules
include     $RULE_PATH/community-
exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include
$RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include     $RULE_PATH/community-
dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules

# Specific web server rules:
include $RULE_PATH/web-cgi.rules
include           $RULE_PATH/web-
coldfusion.rules
include $RULE_PATH/web-iis.rules
include           $RULE_PATH/web-
frontpage.rules
include $RULE_PATH/web-misc.rules
include           $RULE_PATH/web-
client.rules
include $RULE_PATH/web-php.rules
include  $RULE_PATH/community-sql-
injection.rules
include  $RULE_PATH/community-web-
client.rules
include  $RULE_PATH/community-web-
dos.rules
include  $RULE_PATH/community-web-
iis.rules
include  $RULE_PATH/community-web-
misc.rules
include  $RULE_PATH/community-web-
php.rules

# Rules for other services:
include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include        $RULE_PATH/attack-
responses.rules
include $RULE_PATH/oracle.rules
include     $RULE_PATH/community-
oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules
include     $RULE_PATH/community-
ftp.rules
include $RULE_PATH/smtp.rules
include     $RULE_PATH/community-
smtp.rules
include $RULE_PATH/imap.rules
include     $RULE_PATH/community-
imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/nntp.rules
include     $RULE_PATH/community-
nntp.rules
include     $RULE_PATH/community-
sip.rules
include         $RULE_PATH/other-
ids.rules

# Attack-in-progress rules:
include           $RULE_PATH/web-
attacks.rules
include $RULE_PATH/backdoor.rules
include     $RULE_PATH/community-
bot.rules
include     $RULE_PATH/community-
virus.rules

# This ruleset is almost useless
currently:
# include $RULE_PATH/virus.rules
#  Note: this rule is extremely
chatty, enable with care
#                           include
$RULE_PATH/shellcode.rules

# Policy related rules:
# include $RULE_PATH/policy.rules
#  include  $RULE_PATH/community-
policy.rules
# include $RULE_PATH/porn.rules
#  include  $RULE_PATH/community-
inappropriate.rules
# include $RULE_PATH/chat.rules
#                           include
$RULE_PATH/multimedia.rules
# include $RULE_PATH/p2p.rules
#  include  $RULE_PATH/community-
game.rules
#  include  $RULE_PATH/community-
misc.rules

# Extremely chatty rules:
# include $RULE_PATH/info.rules
#    include    $RULE_PATH/icmp-
info.rules
#  include  $RULE_PATH/community-
icmp.rules

# Experimental rules:
# NOTICE: this is currently empty

include
$RULE_PATH/experimental.rules

#                           include
$PREPROC_RULE_PATH/preprocessor.r
ules
#                           include
$PREPROC_RULE_PATH/decoder.rules
#  Include  any  thresholding  or
```

```
suppression commands. See threshold.conf in the
# <snort src>/etc directory for details. Commands don't necessarily need to be
# contained in this conf, but a separate conf makes it easier to maintain them.
# Note for Windows users: You are advised to make this an absolute path,
# such as: c:\snort\etc\threshold.conf
# Uncomment if needed.
# include threshold.conf
```

Nama File : sqlinjection.rule

```
# Copyright 2005 Sourcefire, Inc. All Rights Reserved.
# These rules are licensed under the GNU General Public License.
# Please see the file LICENSE in this directory for more details.
# $Id: community-sql-injection.rules,v 1.10 2006/10/19 20:19:34 akirk Exp $

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"COMMUNITY SQL-INJECTION Microsoft BizTalk Server 2002 rawdocdata.asp"; flow:to_server,established; uricontent:"/rawdocdata.asp?"; nocase; pcre:"/rawdocdata.asp\x3F[^\r\n]*exec/Ui"; classtype:web-application-attack; reference:bugtraq,7470; reference:cve,2003-0118; reference:url,www.microsoft.com/technet/security/bulletin/MS03-016.mspx; sid:100000106; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"COMMUNITY SQL-INJECTION Microsoft BizTalk Server 2002 RawCustomSearchField.asp"; flow:to_server,established; uricontent:"/rawdocdata.asp?"; nocase; pcre:"/RawCustomSearchField.asp\x3F[^\r\n]*exec/Ui"; classtype:web-application-attack; reference:bugtraq,7470; reference:cve,2003-0118; reference:url,www.microsoft.com/technet/security/bulletin/MS03-016.mspx; sid:100000107; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"COMMUNITY SQL-INJECTION OpenBB board.php"; flow:to_server,established; uricontent:"/board.php"; pcre:"/board.php\x3F\w+\x3D[0-9]+\s/Ui"; classtype:web-application-attack; reference:bugtraq,7404; sid:100000108; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"COMMUNITY SQL-INJECTION OpenBB member.php"; flow:to_server,established; uricontent:"/member.php"; pcre:"/member.php\x3F\w+\x3D[0-9]+\s/Ui"; classtype:web-application-attack; reference:bugtraq,7404; sid:100000109; rev:1;)

#Rules submitted by rmkml

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"COMMUNITY SQL-INJECTION WIZZ ForumTopicDetails Sql Injection attempt"; flow:to_server,established; uricontent:"/ForumTopicDetails.php"; nocase; uricontent:"TopicID|3D|"; nocase; uricontent:"union"; nocase; uricontent:"select"; nocase; uricontent:"from"; nocase; uricontent:"ForumUser"; nocase; uricontent:"where"; nocase; reference:bugtraq,15410; reference:url,www.osvdb.org/displayvuln.php?osvdb_id=20846; classtype:web-application-attack; sid:100000192; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"COMMUNITY SQL-INJECTION WIZZ ForumAuthDetails Sql Injection attempt"; flow:to_server,established; uricontent:"/ForumAuthDetails.php"; nocase; uricontent:"AuthID|3D|"; nocase; uricontent:"union"; nocase; uricontent:"select"; nocase; uricontent:"from"; nocase; uricontent:"ForumUser"; nocase; uricontent:"where"; nocase;
```

FORM KUISIONER PENGUJIAN BLACK BOX TEST

Nama :

Pekerjaan :

Instansi :

Keterangan : Berilah tanda √ pada salah satu kolom setiap pertanyaan dibawah ini

No.	Pertanyaan	Penilaian	
		Ya	Tidak
1	Apakah Mesin IDS Snort sudah berjalan dengan baik ?		
2	Apakah Snort sudah mampu mendeteksi serangan ?		
3	Apakah snort sudah bisa menangkap serangan dan menyimpan log file dan alert ke dalam mysql ?		
4	Apakah Web interface AcidBase sudah bisa berjalan dengan baik ?		
5	Apakah jika login pada web acidbase username dan password sesuai dengan database snort?		
6	Apakah AcidBase sudah bisa menampilkan report database snort?		
7	Apakah gammu sms sudah berjalan dengan baik ?		
8	Apakah gammu sudah bisa mengirimkan alert snort lewat sms ?		

Catatan:

_____

_____

_____

Yogyakarta, Juni 2011

( )

CURRICULUM VITAE

Nama	: Khairul Anam
Tempat Tanggal Lahir	: Pamekasan, 8 Mei 1986
Nama Bapak / Pekerjaan	: Buasan / Wiraswasta
Nama Ibu / Pekerjaan	: Hosniyah / Ibu Rumah Tangga
Alamat Rumah	: Desa Poto'an Daya I, Kecamatan Palengaan, Kabupaten Pamekasan Madura.
Alamat di Yogya	: Papringan Gang Ori 2 No 8D Depok Sleman Yogyakarta 55281.
No HP	: 085643751665
Email	: aroel_ezain@yahoo.co.id, khairul.ku@gmail.com
Riwayat Pendidikan	
1993-1999	: SD Poto'an Daya I Palengaan Pamekasan Madura
1999-2003	: MTs. Darul Ulum I Banyuanyar Pamekasan
2003-2005	: MA Darul Ulum I Banyuanyar Pamekasan
2005-2011	: Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta