Snort-Setup for Statistics HOWTO

PoppiSandro

spoppi at gmx.de

도운이

서정룡

s_ryong at hotmail.com

송재숙

affinity at orgio.net

이 HOWTO 는 통계 도구인 ACID (Analysis Consol for Intrusion Databases) 와 SnortSnarf 와 함께 사용되는 Snort 버전 1.8.3 의 설정 방법을 기술한다. 또한 snort 에서 버려지는 패킷이 있는지와 같이 다소의 내부 통계 자료를 얻는 방법이 설명되어 있다.

추가적으로 Max Vision 의 규칙을 자동적으로 갱신하는 방법이 기술되어 있으며 유용한 약간의 스크립트와 swatch 데모 설정이 포함되어 있다.

고친 과정

고침 1.0 2002-01-01고친이 sp

- 최초 발표 버전 - Snort 버전 1.8.3 을 사용하였다 - www.snort.org 에 있는 RPM 을 사용하였다 - 저자의 snortd initscript 에 대한 링크를 추가하였다 -

고침 0.052001-11-14고친이 sp

- 문서이름을 Snort-Setup for Statistics HOWTO 로 변경하였다 - Greg Sarsons 에 고무되어 짧은 통계 스크립트를 추가하였다 - 몇몇 부분을 명확히 설

고침 0.042001-09-29고친이 sp

- Greg Sarsons 으로부터 제안된 "snort 내부 통계자료" 절을 추가하였다 - Greg Sarsons 이 제공한 짧은 통계 스크립트를 추가하였지만 더욱 일반적인

고침 0.032001-09-19고친이 sp

- swatch.conf 에 throttle 옵션을 추가하였다 - ACID 버전을 0.9.6b15 로 변경하였다 - ACID 절에 약간의 주석을 추가하였다 - MD5 체크섬을 추가하였지

고침 0.022001-09-16고친이 sp

Greg Sarsons 으로부터 제안된 약간의 설명

고침 0.012001-09-04고친이 sp

최초 버전

1. 소개

이 문서는 IDS (침입 탐지 시스템, Intrusion Detection System) 를 구현하려는 사람들에게 도움을 주기위해 Snort 와 몇몇 통계 도구를 사용하여 IDS 센서를 만들었을 때 작성되었다. 적어도 이 문서의 내용중 하나라도 도움이 될 수 있다면 이 문서를 작성한 가치가 있을 것이다.

Snort 는 여러 유닉스 플랫폼에서 사용할 수 있는 우수한 NIDS (Network IDS, 네트워크 침입 탐지 시스템) 이다. Snort 홈페이지는 http://www.snort.org/ 로 이 문서에서 기술된 버전은 문서 작성시점의 현재 버전인 1.8.3 이다.

이 문서에 기술할 통계 도구는 snort 에 대한 데이타베이스 분석 도구인 ACID 와 snort 로그에 대한 통계 도구인 SnortSnarf 로 각각 http://www.cert.org/kb/acid/ 와 http://www.silicondefense.com/software/snortsnaft/index.htm (http://www.silicondefense.com/software/snortsnarf/index.htm) 에서 다운로드받을 수 있다.

ACID 에 대해서는 추가적인 지원 패키지들이 필요한데 이들은 **apache** (http://www.apache.org/) 와 같은 웹 서버, PHP 에서 그래프 생성에 사용되는 PHPlot (http://www.phplot.com/) 와 PHP 와 함께 데이타베이스 연결에 사용되는 ADODB (http://php.weblogs.com/ADODB/) 이다.

또한 이 문서는 ACID 에 대해 어떤 추가적인 소프트웨어가 필요한지 그리고 snortd initscript 및 swatch (http://www.stanford.edu/~atkins/swatch) 에 대한 간략한 부분을 포함하여 저자가 사용한 몇몇 스크립트와 함께 펄로 작성된 로그 파일 감시 스크립트를 어떻게 설정하는지를 기술한다. 저자는 swatch RPM 을 작성했으며 이는 http://www.lug-burghausen.org/projects/Snort-Statistics/swatch-3.0.2-1.noarch.rpm 에서 찾을 수 있다.

한개 이상의 snort 센서를 보유하는데 관심이 있는 사람들은 http://www.activeworx.com/ 에서 IDSPM (IDS Policy Manager) 를 조사해 보길 바란다. 이는 새로운 규칙을 기존 규칙으로 병합시킴은 물론 상이한 정책을 갖는 여러 센서들을 유지하기 위한 애플리케이션이다. 단지 "난처한" 것은 W2K/XP 플랫폼에 작동하며 오픈 소스가 아니라는 것이다.

1.1. 저작권 정보

요약해서 말하면 저자는 가능한 많은 채널을 통해 이 정보가 유포되기를 바란다. 그러나 이 HOWTO 문서의 저작권이 유지되길 바라며 이 문서를 재배포하려는 모든 계획을 통보받고 싶다.

질문이 있다면 `<linux-howto at metalab.unc.edu>` 로 연락하기 바란다.

1.2. 부인

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3. 신규 버전

이 문서는 최초 버전이다.

이 HOWTO 문서의 메인 사이트는 http://www.lug-burghausen.org/projects/Snort-Statistics/ 이다.

미러 사이트들로는 Linux Documentation Project (http://www.linuxdoc.org/) 또는 Snort (http://www.snort.org/) 이 있다.

이 HOWTO 문서의 최신 버전은 늘 메인 사이트에서 다양한 포맷으로 얻을 수 있다:

- HTML (http://www.lug-burghausen.org/projects/Snort-Statistics/index.html).
- compressed postscript (A4) (http://www.lug-burghausen.org/projects/Snort-Statistics/Snort-Statistics-HOWTO.ps.gz).
- SGML source (http://www.lug-burghausen.org/projects/Snort-Statistics/Snort-Statistics-HOWTO.sgml).

1.4. 감사

다음의 사람을 포함하여 많은 사람들에게 감사드린다.

- Martin Roesch <`roesch at sourcefire.com`> Author of Snort
- Roman Danyliw <`roman at danyliw.com`> Author of ACID
- James Hoagland <`hoagland at SiliconDefense.com`> Author of SnortSnarf
- Stuart Staniford <`stuart at SiliconDefense.com`> Author of SnortSnarf
- Joe McAlerney <`joey at siliconDefense.com`> Author of SnortSnarf
- John Lim <`jlim at natsoft.com.my`> Author of ADODB
- Afan Ottenheimer <`afan at users.sourceforge.net`> Author of PHPlot
- Andreas ?tling <`andreaso at it.su.se`> Author of arachnids_upd
- Max Vision <`vision at whitehats.com`> "Distributor" of vision.rules and maintainer of http://www.whitehats.com/
- Greg Sarsons <`gsarsons at home.com`> for proof reading and suggestions
- All the peaople on the **snort-users** mailinglist, they helped me and of course they will help YOU >;)
- ...

If I missed someone it was not because of not honoring her or his work!

1.5. 피드백

이 문서에 대한 피드백은 언제나 환영한다. 여러분의 제안과 정보가 없었다면 이 문서는 존재하지 않을 것이다. 추가사항, 견해 및 비평을 다음 이메일 주소 : <`spoppi at gmx.de`> 로 보내주기 바란다.

2. 문서 구조

이 문서는 snort 버전 1.8.3, MySQL 데이타베이스와 그 지원 패키지 PHPlot 와 ADODB 와 함께 snort 의 실시간 통계 자료를 위한 웹 기반 프론트엔드인 ACID, snort 로그파일 분석을 위한 웹 프론트엔드를 갖는 통계 도구 SnortSnarf, Max Vision 의 http://www.whitehats.com/ 사이트로부터 늘 실제 규칙들을 얻기 위한 arachnids_upd 와 snort 가 정지했을 때 snort 에러를 보고하는지를 검사하는 데 사용하는 예제 swatch 설정의 설치 및 설정 방법에 대한 단계별 가이드이다.

3. 기술적인 개요

Snort 는 네트워크 침입 탐지 시스템 (Network Intrusion Detection System, NIDS) 으로 오픈 소스이며 마이크로 소트프뿐만 아니라 다양한 유닉스 플랫폼에서 사용할 수 있다.

NIDS 는 호스트 기반 IDS 가 단지 IDS 가 실행되고 있는 호스트만을 감시하는 것과는 달리 전체 네트워크 세그먼트를 감시한다.

NIDS 는 대부분 방화벽과 함께 사용되기 때문에 공격 자체에 취약하지 않아야 하는 것이 필수적이다. 따라서 snort 와 바인드되어 사용되는 모든 인터페이스들은 ip 주소없이 설치되어야 한다. 그러나, 이는 모든 설정에서 가능한 것이 아니기 때문에, 예를들어 snort 를 isdn 인터페이스 ippp0 에 바인드하려는 경우 snort 에 대해 독립적인 컴퓨터를 사용해 이를 다이얼업 연결에 대한 방화벽 및 라우터로 설치하는 것을 고려해야 한다.

이 주제에 대한 더욱 자세한 정보는 **Firewall-HOWTO** (http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html) 또는 저자의 **Firewalling+Masquerading+Diald+dynamic IP-HOWTO** (http://www.lug-burghausen.org/projects/firewall/firewall-masq-diald.html) 를 보라.

Snort 는 한개 이상의 네트워크 세그먼트를 감시하는데 사용될 수 있으며 이는 뒤에 논의될 것이다.

Snort 는 네트워크 관련 문제를 해결하기 위해 스니퍼 (sniffer, 탐지기) 로도 사용할 수 있지만 이는 이 문서의 주제가 아니다.

ACID (Analysis Console for Intrusion Databases) 는 AIR-CERT 프로젝트의 일부분으로 PHP 에서 그래프 생성을 위한 라이브러리인 PHPlot 와 PHP 와 MySQL 및 PostgreSQL 과 같은 여러 데이타베이스 시스템을 결합하기 위한 추상 라이브러리인 ADODB 를 이용한다. ACID 홈페이지의 내용이다:

"ACID 는 침입 탐지 시스템과 방화벽과 같은 보안 관련 소프트웨어에 의해 생성된 보안 사건 관련 데이타베이스 검색 및 처리를 위한 **PHP** 기반 분석 엔진이다**."**

Max Vision 의 IDS 규칙 (이는 다운로드할 수 있는 파일명으로 **vision.rules**) 이 snort 에 내장되어 있는 규칙들을 완벽하게 하기 위해 사용된다.

arachnids_upd 는 작지만 훌륭한 펄 스크립트로 **wget**를 사용해 현재의 **vision.rules** 을 다운로드하고 아스키 파일내의 규칙들을 선택적으로 삭제한다.

4. 설정

이 장은 snort 와 각종 도구들의 설치 및 실행을 위한 여러 태스크를 기술한다.

저자는 레드햇 리눅스 7.x 을 사용하고 있기 때문에 모든 주어진 경로이름 및 설정 옵션은 결국 레드햇에 특정적인 것이다. 그러나 다른 배포판에 이 문서의 내용을 적용할 때 큰 문제는 없을 것이다.

4.1. Snort 설정하기

현재의 타르볼을 http://www.snort.org/ 에서 얻어 스스로 컴파일하거나 또는 배포판내의 바이너리를 사용해 snort 를 설치할 수 있다.

버전 1.8.3 의 경우 RPM 기반 리눅스 배포판, FreeBSD, 솔라리스 및 윈도우 플랫폼에 대해 www.snort.org (http://www.snort.org/) 에서 미리 컴파일된 바이너리를 얻을 수 있다.

저자는 RPM 을 더이상 유지보수하고 있지 않지만 (버전이 바뀔때 마다 작업을 해야 하기 때문에) http://www.lug-burghausen.org/projects/Snort-Statistics/snortd.multi 에서 **snort.multi** initscript 를 제공할 것이다.

MySQL 을 지원하는 (PostgreSQL 은 지원하지 않는) 저자의 1.8.1 RPM 은 http://www.lug-burghausen.org/projects/Snort-Statistics/snort-1.8.1-4.i386.rpm 에서 얻을 수 있다. PostgreSQL 을 지원하는 버전을 만들기 위해서는 소스 RPM (http://www.lug-burghausen.org/projects/Snort-Statistics/snort-1.8.1-4.src.rpm) 을 다운로드한 후 spec 파일을 편집해 재구축해라. RPM 작성에 익숙하지 않다면 **RPM-HOWTO** (http://www.linuxdoc.org/HOWTO/RPM-HOWTO.html) 또는 RPM 에 대한 많은 좋은 자료와 함께 RPM 에 대해 다운로드할 수 있는 책 **Maximun RPM** 이 있는 http://www.rpm.org/ 을 보아야 한다.

4.1.1. /etc/snort/snort.conf

RPM 을 설치한 후 요구에 맞게 **/etc/snort/snort.conf** 를 편집해야 한다. Martin Roesch 는 snort 타르볼과 RPM 에 포함되어 있는 Snort 사용자 메뉴얼을 PDF 버전으로 작성하였는데 이 문서에서는 설정에 필요한 옵션들만을 다룰 것이기 때문에 사용할 수 있는 다른 옵션들에 대해서는 메뉴얼을 보아야 한다.

또한 타르볼/RPM 에 있는 예제 **/etc/snort/snort.conf** 파일에 세부적인 설명이 있기 때문에 시작하는데 있어 좋은 출발점이 될 것이다.

4.1.1.1. Snort 변수

우선 네트워크 위상을 반영하기 위해 HOME_NET, EXTERNAL_NET 과 DNS_SERVERS 와 같은 변수들을 정의해야 한다. 정확한 주소를 사용했는지 확인해라 그렇지 않은 경우 불가사의한 또는 더욱 나쁘게는 어떠한 경보도 얻을 수 없을 것이다.

복잡한 환경에서 snort 를 사용할 때, 가령 한개의 센서로 다수의 인터페이스를 감시할 때 HOME_NET 과 EXTERNAL_NET 은 정의하기 어렵거나 또는 매우 긴 리스트로 정의될 수 있는데 이 경우 두 변수를 **any** 로 정의할 수 있다. 커다란 내부 네트워크에 아주 많은 네트워크 범위를 넣지 않기 위해서는 어떤 종류의 미리 필터링된 것을 없애야 한다. 그리고 각 패킷에 대해 매우 많은 주소를 통해 snort 를 실행시킬 때 성능에 대한 영향을 최소화시켜야 한다.

몇몇의 다루기 힘든 잘못된 포트스캔 메시지를 없애기 위해서는 변수 DNS_SERVERS 를 snort 의 포트스캔 모듈을 유발하는 네트워크 관리 스테이션과 같은 다른 노드들뿐만이 아니라 모든 dns 서버들의 ip 주소를 보유하도록 정의해야 한다. 이는 현재 진행중인 작업이다.

또한 각자 규칙에서 언급될 수 있는 자신만의 변수들을 정의할 수 있다. 이는 유용한데 예를 들어 자신의 환경에 적합하게 **pass rules**를 사용할 수 있다.

모든 다른 변수들을 적절한 값 또는 **/etc/snort/snort.conf** 에 정의되어 있는 $HOME_NET 으로 정의해라.

```
var HOME_NET any
```

```
      var EXTERNAL_NET any
      # DNS_SERVERS 는 포트스캔시 무시되어지는 DNS 또는 네트워크 관리 스테이션과 같
은 noisy 컴퓨터들의 주소들이다.
      var DNS_SERVERS [1.1.1.1/32,2.2.2.2/32]
      var SMTP_SERVERS $HOME_NET
      ...
```

4.1.1.2. Snort 전처리기

다음은 사용되는 전처리기들을 설정해야 한다. 더욱 많은 전처리기를 사용할 수록 더욱 많은 경보를 유발시킬 수 있지만 성능이 저하된다. 따라서 전처리기를 선택하는데 주의해라.

어떤 전처리기들은 반대되고 있기때문에 또한 Marty 의 **Snort** 사용자 메뉴얼을 보아야 하는데, 이들에 대해서는 새롭게 도입된 것들을 사용해야 한다.

전처리기 **minfrag** 와 **stream** 은 **stream4** 로 전처리기 **defrag** 는 **frag2** 으로 대체되었다.

frag2 는 snort v1.8 에 도입된 새로운 IP defragmentation (단편화된 조각들을 연속적인 조각으로 만들어줌) 처리기로 **defrag/minfrag** 보다 더욱 메모리 효율적이다.

Snort 사용자 메뉴얼로부터: **Stream4** 모듈은 **snort** 에 **TCP** 스트림 재조합 및 상태에 대한 분석 **(stateful analysis)** 능력을 제공한다**. Snort** 는 견고한 스트림 재조합 능력에 의해 "상태가 없는 **(stateless)"** 공격들을 무시한다**. Stream4** 모듈은 또한 사용자들에게 **256** 개 이상의 동시적인 **TCP** 스트림을 추적할 수 있도록 한다**. Stream4** 는 **64,000** 개 이상의 **TCP** 접속을 다루기 위해 확장될 수 있어야 한다**.**

Stream4 모듈은 **stream4** 와 **stream4_reassemble** 2 개의 전처리기로 이루어져 있는데 둘 모두 사용되어야 한다.

두 전처리기에 대해서는 여러가지 옵션이 있지만 **stream4** 에 대해서는 포트스캔 이벤트시 경보를 주기 위한 **detect_scans** 과 공격적인 RST 패킷, SYN 패킷에 데이타가 있고 윈도우 시퀀스 넘버의 범위를 넘어서는 것과 같이 스트림 이벤트가 일어날 때 알리기 위해 **detect_state_problems** 옵션만을 사용할 것이다.

stream4_reassemble 에 대해서는 재조합이 단지 약간의 미리 정의된 포트대신 모든 포트를 감시하도록 하는 **ports all** 옵션을 사용한다. 솔직히 이는 일종의 편집증으로 snort 센서의 cpu 이용상황에 영향을 끼친다. 그러나 저자는 평균적으로 낮은 이용상황을 갖는 펜티엄 III 800 MHz 컴퓨터가 세개의 100Mbit/s full duplex line 에서 감시하는 경우 어떠한 나쁜 결과도 얻지 않았기 때문에 이 옵션이 더욱 좋은 해결 방법이라고 생각한다.

두개의 다른 전처리기들은 **portscan** 와 **portscan-ignorehosts** 로 각각 포트스캔 탐지 담당 및 포트스캔 탐지가 무시되는 호스트들이다.

0.0.0.0/0 형태를 사용하여 **portscan**이 모든 네트워크를 찾기 위해서 접근되어지는 포트 넘버의 수를 설정하고 초당 탐지 기간을 정의하라. 추가적으로 포트스캔 로그 파일에 대한 절대 경로를 제공해야 한다.

portscan-ignorehosts 를 이용해 네임 서버 및 네트워크 관리 스테이션과 같이 너무 많이 말하며 포트스캔 탐지를 유발하는 호스트로부터의 어떤 수상한 경보를 제거한다 (위의 변수 **DNS_SERVERS** 를 보라).

Marty 의 사용자 메뉴얼에 언급되어있지 않지만 여기서 사용할 어떤 전처리기들이 있다. **unidecode** 는 **http_decode** 를 대체하는데 http 및 UNICODE 공격을 정규화한다 (표준적인 상황으로 해석한다). **rpc_decode** 주어진 포트에서의 rpc 트래픽을 정규화하며 **bo** 는 백 오러피스 공격을 검사하며 **telnet_decode** 는 tenlnet 협상 스트링을 정규화한다.

SPADE 와 같은 다른 전처리기들은 여기서 다루지 않지만 추후 버전에서는 다루어질 것이다.

결국 여기서 언급한 것들은 **/etc/snort/snort.conf** 의 전처리기 부분이다.

```
preprocessor frag2
preprocessor stream4: detect_scans detect_state_problems
preprocessor stream4_reassemble: ports all
preprocessor unidecode: 80 8080
preprocessor rpc_decode: 111
preprocessor bo: -nobrute
preprocessor telnet_decode
preprocessor portscan: 0.0.0.0/0 6 3 /var/log/snort/portscan.log
preprocessor portscan-ignorehosts: $DNS_SERVERS
```

4.1.1.3. Snort 출력 모듈

다음은 출력 모듈들의 설정으로 이 중에서 syslog 모듈인 **alert_syslog** 를 사용해 syslog 에 경보를 보낼 것이며 **database** 모듈을 사용해 MySQL 데이타베이스에 추가적으로 기록할 것이다.

alert_syslog 모듈은 기록되어져야 하는 것들에 대해 어떤 옵션을 필요로한다. 저자와 같이 로그파일을 분석하기 위해 SnortSnarf 을 사용한다면 **LOG_PID** 옵션을 추가해야한다. 그렇지 않다면 SnortSnarf 를 사용할 때 문제가 있다.

앞에서 말했듯이 ACID 를 사용할 것인데 따라서 데이타베이스에 기록하기 위해 snort 를 설정할 필요가 있다. 특별한 이유없이 MySQL 을 선택했다 (postgreSQL 보다 MySQL 을 더 많이 들었을 뿐이다).

database 출력 모듈은 다음 변수들을 필요로 한다:

log | alert

alert 기능에 대한 로그. **log** 기능도 가능하다. 포트스캔 경보를 데이타베이스에 저장하고 싶다면 **alert** 를 사용해야 한다.

mysql|postgrsql|odbc|oracle|mssql

데이타베이스 타입.

user=<username>

데이타베이스에 사용될 사용자이름을 정의한다.

password=<password>

주어진 사용자에 대해 필요한 패스워드.

dbname=<databasename>

기록에 사용되는 데이타베이스 이름.

host=<hostname>

데이타베이스가 실행중인 호스트를 정의한다. 데이타베이스가 snort 센서에서 실행되고 있다면 localhost 를 사용해라.

sensor_name=<sensor name>

하나 이상의 센서가 하나의 데이타베이스에 기록하고 있다면 여러 센서를 구별하기위해 고유의 이름을 부여한다.

/etc/snort/snort.conf 의 출력 모듈 부분을 보자.

```
       output alert_syslog: LOG_AUTH LOG_ALERT LOG_PID
       output database: alert, mysql, user=snort password=mypassword db-
name=snort host=localhost sensor_name=mysensor
```

하나 이상의 snort 센서를 사용하며 하나의 데이타베이스에 기록하고 싶다면 별도의 머신에 중앙 데이타베이스를 사용하기 바란다. 이런 경우 공격이 발견될 때 경보 데이타를 하나의 컨솔과 연관시킬 수 있으며 더욱 좋은 개요를 얻을 수 있다.

4.1.1.4. Snort 규칙 집합

규칙들은 snort 의 필수적인 부분으로 다양한 항목으로 나누어져 있는데 이들은 ***.rules** 로 끝나며 **/etc/snort/** 에서 찾을 수 있다. 1.8 이상의 버전에서는 분류 형태를 반영하기 위해 포맷이 변경되었다. 또한 classtype 의 우선권 설정을 정의할 수 있다.

원래의 snort tarball 을 사용하고 있다면 모든 규칙 파일들과 **classification.config** 파일을 복사하길 바란다.

분류 타입의 설정은 **/etc/snort/classification.config** 파일에서 하는데 이 파일은 적재된 snort 규칙에 대해 미리 조정되어 있기 때문에 손댈 필요는 없다. 그러나 Max Vision 의 **vision.rules** 를 사용하려 한다면 classtype 이 다르기 때문에 약간의 라인들을 추가해야 할 것이다. 그저 모든 **config classification:** 라인을 **vision.conf** 에서 **/etc/snort/classification.config** 로 복사해서 붙여넣어라. 이전 파일이 snort 1.8 에서 도입된 새로운 포맷에 맞지 않는 경우 snort 1.8 에 대한 **vision.rules** 를 잊지말고 얻어라 (http://www.whitehats.com/에서 **vision18.rules** 및 **vision18.conf**)

다음은 **vision.rules**을 사용한 **/etc/snort/classification.config** 파일이다:

```
       #
       # config classification:간략한 이름, 간략한 설명, 우선권
       #
       #config classification: not-suspicious,Not Suspicious Traffic,0
       config classification: unknown,Unknown Traffic,1
       config classification: bad-unknown,Potentially Bad Traffic, 2
       config classification: attempted-recon,Attempted Information Leak,3
       config classification: successful-recon-limited,Information Leak,4
       config classification: successful-recon-
largescale,Large Scale Information Leak,5
       config classification: attempted-dos,Attempted Denial of Service,6
       config classification: successful-dos,Denial of Service,7
       config classification: attempted-
user,Attempted User Privilege Gain,8
       config classification: unsuccessful-
user,Unsuccessful User Privilege Gain,7
       config classification: successful-
user,Successful User Privilege Gain,9
       config classification: attempted-
admin,Attempted Administrator Privilege Gain,10
       config classification: successful-
admin,Successful Administrator Privilege Gain,11

       # vision18.conf 에서 추가된 부분
       # classification for use with a management interface
       # low risk
       config classification: not-
suspicious,policy traffic that is not suspicious,0
       config classification: suspicious,suspicious miscella-
neous traffic,1
       config classification: info-
failed,failed information gathering attempt,2
       config classification: relay-failed,failed relay attempt,3
       config classification: data-failed,failed data integrity attempt,4
       config classification: system-
failed,failed system integrity attempt,5
       config classification: client-
failed,failed client integrity attempt,6
       # middle risk
       config classification: denialofservice,denial of service,7
       config classification: info-attempt,information gathering attempt,8
       config classification: relay-attempt,relay attempt,9
       config classification: data-attempt,data integrity attempt,10
       config classification: system-attempt,system integrity attempt,11
       config classification: client-attempt,client integrity attempt,12
       config classification: data-or-info-
attempt,data integrity or information gathering attempt,13
```

```
       config classification: system-or-info-
attempt,system integrity or information gathering attempt,14
       config classification: relay-or-info-
attempt,relay of information gathering attempt,15
       # high risk
       config classification: info-
success,successful information gathering attempt,16
       config classification: relay-success,successful relay attempt,17
       config classification: data-
success,successful data integrity attempt,18
       config classification: system-
success,successful system integrity attempt,19
       config classification: client-
success,successful client integrity attempt,20
```

분류 및 규칙 파일들은 **/etc/snort/snort.conf** 파일에 포함되어 있는데 여기서 사용된 어떤 규칙 파일들은 표준 배포에 포함되어 않기 때문에 CVS 에서 복사되었다. 예를 들면 **virus.rules**.

전에 말했듯이 **vision.rules** 파일은 뒤에 논의되는 **arachnids_upd** 도구를 통해 가져올 것이다.

Arachnids_upd 는 **vision18.rules** 에서 **vision.rules** 로 이름을 변경시키지만 물론 규칙은 1.8 이상 버전에 대해 준비된 것들이다.

vision.rules 에서의 INTERNAL 및 EXTERNAL 에 대한 변수 정의가 snort 규칙과 동일하지 않기 때문에 이러한 이름들을 변경시키기 위해 스크립트를 사용한다. 아래의 **arachnids_upd** 절을 봐라.

```
       # Include classification & priority settings
       include /etc/snort/classification.config

       include /etc/snort/exploit.rules
       include /etc/snort/scan.rules
       include /etc/snort/finger.rules
       include /etc/snort/ftp.rules
       include /etc/snort/telnet.rules
       include /etc/snort/smtp.rules
       include /etc/snort/rpc.rules
       include /etc/snort/rservices.rules
       include /etc/snort/backdoor.rules
       include /etc/snort/dos.rules
       include /etc/snort/ddos.rules
       include /etc/snort/dns.rules
       include /etc/snort/netbios.rules
       include /etc/snort/web-cgi.rules
       include /etc/snort/web-coldfusion.rules
       include /etc/snort/web-frontpage.rules
       include /etc/snort/web-iis.rules
       include /etc/snort/web-misc.rules
```

```
include /etc/snort/sql.rules
include /etc/snort/x11.rules
include /etc/snort/icmp.rules
include /etc/snort/shellcode.rules
include /etc/snort/misc.rules
include /etc/snort/policy.rules
include /etc/snort/info.rules
#include /etc/snort/icmp-info.rules
include /etc/snort/virus.rules
include /etc/snort/local.rules

# vision.rules will be catched by arachnids_upd
include /etc/snort/vision.rules
```

/etc/snort/snort.conf 설정을 마친 후 **/etc/rc.d/init.d/snortd start** 명령을 통해 snort 를 실행시키고 **/var/log/messages** 로그 파일내의 모든 에러들을 수정해야 한다 (데이타베이스는 아직 설정되어 있지 않기 때문에 데이타베이스와 관련된 모든 메세지는 무시해라). 모든 것이 잘 진행되었다면 다른 부분의 설정으로 가자.

4.1.2. /etc/rc.d/init.d/snortd

/etc/rc.d/init.d/snortd 에서는 적어도 인터페이스 부분을 편집해야 하는데, **INTERFACE="eth0"** 를 각자 사용하는 인터페이스로 수정해라. 이는 다른 이더넷 (**ethx**) 및 **pppx** 또는 **ipppx** 인터페이스일 수 있다. 예를 들어 ISDN 을 사용하고 있다면 인터페이스 정의는 다음과 같다.

```
INTERFACE="ippp0"
```

Snort 센서가 하나의 인터페이스에서만 감시하고 있다면 snortd initscript 를 사용하는 것으로 충분하다. 그러나 하나 이상의 인터페이스를 갖고 있다면 이에 대해 저자가 확장한 스크립트를 보기 바란다. 단지 하나의 인터페이스를 갖고 있지만 저자와 같이 swatch 를 사용하고 싶다면 swatch 부분을 snortd 스크립트로 복사해라 (RPM 문서의 contrib 절을 보라).

다음 저자가 확장한 snortd initscript 는 snort 가 하나 이상의 인터페이스를 감시함을 알 수 있을 것이다. 인터페이스 이름에 **any** 를 사용할 수도 있다고 말할 수도 있다. 왜냐하면 기본적인 **libpcap** 이 이를 가능하게 하기 때문이다. 그러나 snort 센서가 설치되어 있는 로컬 네트워크 감시에 관심이 없기 때문에 이는 저자가 사용하려고 하는 의도가 아니다. 이 로컬 네트워크는 방화벽 설치와 같은 부가적인 보안 설정이 되어 있는 별도의 네트워크 세크먼트로 보안적인 환경에 있어야 한다. 따라서 snort 네트워크 자체를 목표로한 공격의 탐지를 원하는 경우를 제외하고는 스니핑은 의미가 없다. 이 세크먼트에 하나 이상의 센서를 사용한다면 세크먼트를 보호하기 위해 모든 센서가 아닌 단지 하나의 센서만 설정할 필요가 있다.

저자는 **/etc/rc.d/init.d/functions** 내의 레드햇사의 **daemon** 함수에서 파생된 새로운 함수 **daemonMult** 를 추가하였다. 이는 프로그램을 한번 이상 시동시킬 수 있다. 저자는 새로운 옵션 **--mult** 를 도입하기 위해 레드햇사에 **daemon** 함수에 대한 패치를 보냈다. 이 추가가 이루어진다면 **daemonMult** 함수는 쓸모없게 될 것이며 호출은 **daemonMult** 에서 **daemon --mult** 로 변경될 것이다.

또한 저자는 리부팅시 에러 메세지를 없애기 위해 하부시스템 이름을 snort 에서 snortd 로 변경하였다 (레드햇에서 killall 스크립트는 정확한 이름에 의존한다).

저자의 스크립트를 이용하면 감시할 다수의 인터페이스를 정의할 수 있는데 밑의 스크립트에서와 같이 **INTERFACE** 에 공백으로 분리된 목록을 사용해라.

듣고 있는 인터페이스가 이미 작동중인지 IP 주소가 정의되어 있는지를 보기 위해 약간의 정상성 검사도 또한 포함되어 있다. 정의된 IP 주소가 있다면 레드햇 리눅스의 경우 **/etc/sysconfig/network-scripts/ifcfg-(interface-name)** 에 있는 해당 config 가 사용될 것이다. 그렇지 않다면 인터페이스는 promiscuous 모드에서 IP 없이 설정될 것이다.

이는 이더넷 인터페이스를 제외한 어떠한 인터페이스에서도 아직 테스트되지 않았다. 저자는 곧 ISDN 인터페이스에 대해 검토할 것이며 어떠한 차이가 있는 지를 보고할 것이다.

하나의 snort 프로세스가 각 인터페이스에서 시작되며 또한 **swatch**가 규칙 갱신을 위해 snort 를 재시작할 때 에러를 검사하기 위해 실행될 것이다 (밑의 **swatch** 절을 보라).

snort 를 셧다운할 때 모든 IP 없는 인터페이스는 셧다운될 것이다. 그러나 IP 설정이 되어 있는 인터페이스는 셧다운되지 않을 것이다. 이는 snort'ed 인터페이스가 snort 센서에 필수적인 경우 접근불가를 야기할 수 있기 때문이다.

아마도 더욱 좋은 해결방법은 다음과 같은 엔트리에 대해 인터페이스의 config 파일을 검사하는 것으로

```
ONBOOT=yes
```

yes 가 없는 경우에만 인터페이스가 셧다운될 것이다. 그러나 이것은 아직 구현되어 있지 않다.

다음은 확장된 snort initscript 이다:

```
#!/bin/sh
#
# snortd         Start/Stop the snort IDS daemon.
#
# chkconfig: 2345 40 60
# 설명:  Snort 는 현재 1100 개 이상의 호스트 및 네트워크 취약성, 포트스캔, 백도어 등
을 탐지하는
#               경량 네트워크 침입 탐지 시스템이다.
#
# June 10, 2000 -- Dave Wreski Dave Wreski <dave at linuxsecurity.com>
#   - 최초 버전
# July 08, 2000 Dave Wreski <<dave at guardiandigital.com>
```

```
#   - snort user/group 을 추가하였다
#   - 1.6.2 버전 지원
# April 11, 2001 Sandro Poppi <spoppi at gmx.de>
#   - 다이얼 업 라인 또는 한개 이상의 스니퍼 인터페이스 사용을 위해 다중 인터페이스 옵
션을 추가하였다
#     저자는 "-i any" 를 사용하는 libpcap 옵션이 좋은 선택이라고 생각하지 않는다.
#     왜냐하면 snort 가 한개이상의 ip 가 없는 인터페이스들을 감시하도록 설정될 수 있
기 때문이다.
#      그러나 모니터 인터페이스는 보호되지 않은 상태로 존재한다.
#   - 리부팅할때의 에러 메시지를 없애기 위해 하부시스템 이름을 snort 에
서 snortd 로 변경하였다
#      (레드햇의 killall 스크립트는 정확한 이름에 의존한다)
#   - snort 의 다중 인스턴스를 실행시킬 수 있도록 /etc/rc.d/init.d/functions 내
의 daemon 함수에서 파생된
#      daemonMult 함수를 추가하였다
#      (결국 이는 레드햇의 daemon 함수내로 통합될 수 있는데 저자에게 연락해라)
# January 01, 2002 Sandro Poppi <spoppi at gmx.de>
#   - swatch 가 설치되어 있는지의 검사를 추가하였다
#   - 이더넷이외의 인테페이스들에 대한 검사를 추가하였다. 이들은 ifconfig 로 작동된다
고 생각하기 때문이다.
#
# Source function library.
. /etc/rc.d/init.d/functions

# 프로그램을 한번 이상 시작하기 위한 함수
# /etc/rc.d/init.d/functions 내의 데몬 함수를 재작성한 것이다
daemonMult() {
        # 구문을 테스트한다.
        gotbase=
        user=
        nicelevel=0
        while [ "$1" != "${1##-}" -o "$1" != "${1##+}" ]; do
          case $1 in
            ”)    echo ’$0: Usage: daemon [+/-nicelevel] {program}’
                   return 1;;
            --check)
                   shift
                   base=$1
                   gotbase="yes"
                   shift
                   ;;
            --user)
                   shift
                   daemon_user=$1
                   shift
                   ;;
            -*|+*) nicelevel=$1
                   shift
                   ;;
```

```
             *)    nicelevel=0
                   ;;
          esac
        done

        # basename 을 저장한다.
        [ -z $gotbase ] && base=`basename $1`

        # 반드시 어느 곳에도 코어 덤프하지 않도록 해라; 이것이 데몬과 관련된
        # 문제를 저지하는 동시에 또한 약간의 보안 문제를 없앤다.
        ulimit -S -c 0 >/dev/null 2>&1

        # Echo daemon
        [ "$BOOTUP" = "verbose" ] && echo -n " $base"

        # 데몬을 구동시킨다.
        if [ -z "$daemon_user" ]; then
           nice -n $nicelevel initlog $INITLOG_ARGS -
c "$*" && success "$base startup" || failure "$base startup"
        else
           nice -n $nicelevel initlog $INITLOG_ARGS -c "su $daemon_user -
c \"$*\"" && success "$base startup" || failure "$base startup"
        fi
}

# 네트워크 인터페이스(들)를 지정해라
INTERFACE="eth1 eth2"

# See how we were called.
case "$1" in
start)
        if [ -x /usr/bin/swatch ] ; then
          echo -n "Starting swatch: "
          # swatch 를 사용하기 위해 저자가 써넣었다
          # snort 구동시의 에러에 대한 지시를 얻기 위해 snort 전에 이를 실행시킨다
          # snort 옵션 -s 를 사용한다면 /var/log/secure 를 사용해라
          # snort.conf 에서 alert_syslog: 출력 옵션을 사용한다
면 /var/log/messages 를 사용해라
          /usr/bin/swatch --daemon --tail /var/log/messages --config-
file /etc/swatch/swatchrc &
          touch /var/lock/subsys/swatch
          echo "done."
          echo
        fi

        # 다중 인터페이스 옵션을 추가하였다
        for i in `echo "$INTERFACE"` ; do
          echo -n "Starting snort on interface $i: "
```

```
     # 스크립트 구동시 snort 에 대해 ip 가 없는 스니퍼 인터페이스를 구현하
기 위해 삽이하였다
     # 인터페이스가 아직 적재되지 않았거나 또는 구동되지 않았다면
     if [ `/sbin/ifconfig $i 2>&1 | /bin/grep -
c "Device not found"` = "0" \
       -o `/sbin/ifconfig $i 2>&1 | /bin/grep -
c "UP"` = "0" ] ; then

      # 이더넷외의 다른 인터페이스를 검사한다
      if [ `echo $i | /bin/grep -c "^eth"` = "1" ] ; then
       # 주어진 인터페이스에 대한 config 가 있는지를 검사한다
       # 보통 이는 스니퍼 인터페이스에 대해서는 보안상의 이유로 생략되어야 한
다
       if [ -s "/etc/sysconfig/network-scripts/ifcfg-$i" ]; then
        # config 를 사용한다
        /sbin/ifup $i
       else
        # ip 가 없는 스니퍼 인터페이스
        /sbin/ifconfig $i up promisc
       fi
      fi
     fi
     # 위의 재작성된 데몬 함수를 실행시킨다
     daemonMult /usr/sbin/snort -u snort -g snort -d -D \
        -i $i -I -l /var/log/snort -c /etc/snort/snort.conf
     echo
    done

    touch /var/lock/subsys/snortd

    ;;
 stop)
    echo -n "Stopping snort: "
    killproc snort
    rm -f /var/lock/subsys/snortd

    # 저자가 삽입하였다
    if [ -x /usr/bin/swatch ] ; then
     echo
     echo -n "Stopping swatch: "
     kill `ps x|grep "/usr/bin/swatch"|grep -
v grep|awk '{ print $1 }'`
     rm -f /var/lock/subsys/swatch
    fi

    # 단지 ip 주소가 없다면 인터페이스를 셧다운시킨다
    # 이더넷 인터페이스라면 인터페이스를 셧다운시키길 원치 않기 때문에 다음을 실행
시킨다
    for i in `echo "$INTERFACES"`; do
```

```
          if [`echo $i | /bin/grep -c "^eth"` = "1" -a \
              `/sbin/ifconfig $i 2>&1 | /bin/grep -
c "inet addr:"` = "0" ] ; then
            /sbin/ifconfig $i down
          fi
        done
        echo
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  status)
        status snort
        #status swatch
        ;;
  *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
esac
exit 0
```

4.1.3. /etc/snort/snort-check

이 쉘 스크립트는 **smbclient** 를 통해 윈팝업을 생성시키거나 주어진 사람에게 이메일을 보내는데 사용된다. 이는 snort 홈페이지에 발표된 Bill Richardson 의 스크립트에 고무되어 작성하였다.

윈팝업 부분은 snort 1.8 에 도입된 **smb** 출력 모듈로 인해 쓸모없을지도 모르며 저자는 이를 아직 테스트하지 않았다.

```
#!/bin/sh

# 다수의 포맷으로 경보를 보내기 위해 swatch 내에서 실행되는 스크립트
# www.snort.org 에 있는 Bill Richardson 의 스크립트에 영감을 받았다
# 윈팝업을 보내는 워크 스테이션의 이름을 갖고 있는 "hosts" 파일을 읽어들이도록
# 확장되었다. 구문은 snortd 옵션 -M 과 같다.
# Poppi, 02.05.2001

# 선결조건:
# Samba 가 정확히 설치되어 있어야 한다.
# 각자 시스템에 맞게 다음 변수들을 변경시킨다 (레드햇 7.x 을 사용한다면 무방하다)

# hostfile 은 윈팝업을 위한 워크 스테이션을 포함하고 있는 파일명을 갖고 있다.
hostfile="/etc/snort/hosts"
```

```
# recipientfile 은 모든 recipient (수령자) 의 주소를 갖고 있다.
# 각 recipient 는 한줄에 하나이다.
recipientfile="/etc/snort/recipients"

# recipient 파일이 존재한다면
if [ -s "$recipientfile" ] ; then
  # 이메일 주소를 갖는 recipientlist 를 생성한다
  for i in `cat $recipientfile` ; do
    recipients="$recipients "$i
  done

  echo "$*" | mail -s "Snort-Alert!!!" "$recipients"
fi

# hostfile 존재한다면 윈팝업을 보낸다
if [ -s "$hostfile" ] ; then
  for i in `cat $hostfile` ; do
    echo "Snort-Alert! $*" | smbclient -M $i > /dev/null 2>&1
  done
fi
```

4.1.3.1. /etc/snort/hosts

이 파일에 snort 메시지를 받는 호스트의 워크 스테이션의 이름을 한줄에 하나씩 넣어라.

```
ws001
ws002
ws003
```

4.1.3.2. /etc/snort/recipients

/etc/snort/recipients 에 snort 경보를 받길 원하는 (또는 받도록 되어있는) 수령자의 이메일 주소를 한줄에 하나씩 넣어라.

```
jane@internal.local.com
henk@snort.info
sandro@snort.info
```

두파일중 하나라도 생략된다면 해당 특징은 금지된다.

4.1.4. Snort 내부 통계자료

Snort 는 약간의 내부 통계자료를 출력할 수 있는 기능이 내장되어 있는데 이는 다음 명령을 사용하여 기록될 수 있다:

/bin/kill -SIGUSR1 <pid of snort>

또는 동일 머신에 한개 이상의 snort 프로세스가 있고 동시에 모든 정보를 얻고 싶다면 다음 명령을 사용할 수 있다:

/bin/killall -USR1 snort

위의 명령을 사용한 경우 syslog (**/var/log/messages**) 에서 다음과 같은 내부 통계자료를 얻는다.

```
Sep 29 07:51:48 ids01 snort[8000]:  ===============================================================================
Sep 29 07:51:48 ids01 snort[8000]: Snort ana-
lyzed 27316 out of 27316 packets,
Sep 29 07:51:48 ids01 snort[8000]: dropping 0(0.000%) packets
Sep 29 07:51:48 ids01 snort[8000]: Breakdown by proto-
col:               Action Stats:
Sep 29 07:51:48 ids01 snort[8000]:     TCP: 27152      (99.400%)         ALERTS: 0
Sep 29 07:51:48 ids01 snort[8000]:     UDP: 0          (0.000%)          LOGGED: 0
Sep 29 07:51:48 ids01 snort[8000]:    ICMP: 164        (0.600%)          PASSED: 0
Sep 29 07:51:48 ids01 snort[8000]:     ARP: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:    IPv6: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:     IPX: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:   OTHER: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]: DISCARD: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]: ===============================================================================
Sep 29 07:51:48 ids01 snort[8000]: Fragmentation Stats:
Sep 29 07:51:48 ids01 snort[8000]: Fragmented IP Pack-
ets: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:     Fragment Trackers: 0
Sep 29 07:51:48 ids01 snort[8000]:    Rebuilt IP Packets: 0
Sep 29 07:51:48 ids01 snort[8000]:    Frag elements used: 0
Sep 29 07:51:48 ids01 snort[8000]: Discarded(incomplete): 0
Sep 29 07:51:48 ids01 snort[8000]:    Discarded(timeout): 0
Sep 29 07:51:48 ids01 snort[8000]:   Frag2 memory faults: 0
Sep 29 07:51:48 ids01 snort[8000]: ===============================================================================
Sep 29 07:51:48 ids01 snort[8000]: TCP Stream Reassembly Stats:
Sep 29 07:51:48 ids01 snort[8000]:         TCP Pack-
ets Used: 27152      (99.400%)
Sep 29 07:51:48 ids01 snort[8000]:          Stream Trackers: 1
Sep 29 07:51:48 ids01 snort[8000]:           Stream flushes: 0
Sep 29 07:51:48 ids01 snort[8000]:            Segments used: 0
Sep 29 07:51:48 ids01 snort[8000]:    Stream4 Memory Faults: 0
Sep 29 07:51:48 ids01 snort[8000]: ===============================================================================
```

그러나 다음을 기억해라: 1.8.3 이전 버전에서는 새로운 통계 자료를 얻기위해서는 snort 를 재구동시켜야 한다. 따라서 이전 버전이라면 늘 **kill -SIGUSR1** 과 snort restart 를 함께 실행시켜라.

우선 처음의 두 라인을 보아야 한다. snort 가 버려지는 (dropped) 패킷이 있다고 말한다면 snort 설정뿐만이 아니라 snort 박스의 설정도 매우 자세히 조사해야 한다.

예를 들어 박스에 필수적이지 않은 모든 불필요한 서비스를 정지시켜라. 그리고 **top** 명령의 출력을 조사해라. 유휴 카운터가 매우 낮다면 어떤 프로세스들이 cpu 시간을 소비하며 결국 해당 프로그램 패킷을 outsource 하고 있는지 알아내야 한다. 이는 적은 메모리 및/또는 사양이 낮은 cpu 를 갖는 동일 머신에서 ACID, 데이타베이스 및 snort 를 사용할 때에 해당된다.

다른 데이타 라인들은 전처리기들 및 이들의 작업에 대한 개요를 제공한다. 또한 메모리 fault 부분을 조사해야 한다. 숫자가 0 이 아니라면 메모리 사용을 조사해야 하며 결국 더욱 많은 메모리를 사용하도록 전처리기를 설정해야 한다 (**/etc/snort/snort.conf** 내의 적절한 부분을 조사해라).

다음은 Greg Sarsons 에 영감을 받은 snort 의 내부 통계자료를 얻기 위한 짧은 스크립트이다. 이를 파일로 저장한 후 snort 를 재시작해라.

통계자료 파일은 **/var/log/snort/archive** 에 저장될 것이며 따라서 이 디렉토리를 우선적으로 생성해야 한다.

```
#!/bin/bash
# syslog 또는 kill -USR1 <snort-pid> 실행 후 생성된 주어진 파일로부터
# 특별한 snort 통계 자료를 생성 및 추출하기 위한 스크립트
#
# 이 스크립트는 pid 가 로그파일에 기록됨을 가정한다.
# 이는 snort.conf 파일에 다음 라인을 사용하면 가능하다:
# output alert_syslog: LOG_AUTH LOG_ALERT LOG_PID
#
# (c) Sandro Poppi 2001
# Released under GPL

echo "Starting gathering snort internal statistics. Please be patient..."

if [ "$1." == "." -o ! -e "$1" ] ; then
  # 주어진 파일이 없거나 존재하지 않는 경우 다음 디폴트 파일을 사용한다
  log_file="/var/log/messages"

else
  # 로그파일의 위치가 표준적이지 않을때는 반드시 snort 가 이 로그 파일을 사용하도
록 해라
  # 그렇지 않다면 USR1 시그널을 보낼때 이 스크립트는 작동하지 않을 것이다
  log_file="$1"
fi

# snort pid 를 알아낸다
snort_pid=`/sbin/pidof snort`
```

```
# 모든 snort 프로세스에 대해 내부 통계자료를 얻는다
# 이미 정렬된 출력을 얻기위해 killall 을 사용하지 않는다
for i in `echo $snort_pid` ; do
  kill -USR1 $i

  # snort 가 통계자료를 syslog 에 보내도록 2초동안 휴면한다
  sleep 2
done

# USR1 시그널을 보낸후 즉각적으로 snort 를 재시작한다
# 이는 2001년 1월 11일경 이후의 snort CVS 버전 또는 1.8.2 버전 이상의 모든 버전을
# 사용할 때 생략될 수도 있다.
/etc/rc.d/init.d/snortd restart

for i in `echo $snort_pid` ; do
  # process logfile

  filename=/var/log/snort/archive/snort.`date "+%Y-%m-%d"`.$i.log

  # 기존 파일을 검사하고 존재한다면 파일명을 변경한다
  if [ -e "$filename" ] ; then
    mv "$filename" "$filename.bak"
  fi

  egrep "snort\[$i\]:" $log_file > "$filename"

  # 다음 라인과 같은 라인들을 사용하여 버려지는 패킷이 있는지 조사한다
  # Oct 22 18:02:06 xbgh17183 snort[573]: dropping 0(0.000%) packets
  if [ "`egrep "dropping" $filename | awk -
F "[ (]" ’{ print $7 }’`" != "0" -a \
       "`egrep -c "dropping" $filename`" != "0" ] ; then
    echo "Snort’s dropping packets!!! Take a look on the configura-
tion and/or the system’s performance!!!"
  fi

done

echo "Gathering snort internal statistics finished..."
```

4.1.5. Snort 테스트하기

snort 를 테스트하기 위해서는 **/etc/rc.d/init.d/snortd** 를 편집하고 인터페이스가 루프백 인터페이스 **lo** 에서 듣게 해야 한다. 네트워크 카드가 설치되어 있는 사람의 경우는 대신 **eth0** 를 사용할 수 있지만 snot 와 snort 가 동일 머신에서 실행되고 있다면 인터페이스를 통해 어떠한 패킷도 전송되지 않기때문에 snot 를 두번째 pc 를 사용해 실행시켜야 한다.

snort 를 테스트하는 아마도 가장 간단한 방법은 http://www.sec33.com/sniph/ 에서 찾을 수 있는 **snot** 를 사용하는 것이다.

Snot 를 위해 libnet 를 설치해야 하는데 레드햇 7.x 에는 사용가능한 RPM 이 없기 대문에 http://rpmfind.net/ 및 맨드레이크 사이트 http://www.mandrake.com/ 에서 찾을 수 있는 맨드레이크 소프트사의 **libnet-1.0.2-6mdk.i586.rpm** 을 사용할 수 있다. 대부분의 맨드레이크 RPM 들은 레드햇 시스템에서 아무 문제없이 사용할 수 있다. 그러나 맨드레이크는 **i386** RPM 들을 제공하지 않음에 주의해라 따라서 예전 펜티엄 P5 이전 프로세서에는 이들을 사용할 수 없다. 이런 경우 http://www.packetfactory.net/projects/libnet 에서 소스를 얻은 후 스스로 컴파일해야 한다.

snot 를 컴파일하기 위해서는 단지 tarball 을 untar 한 후 snot 디렉토리에서 **make** 명령을 실행시키면 된다. 에러없이 컴파일이 종료되면 바로 snot 를 사용할 수 있다. 그렇지 않다면 어떤 개발 패키지가 없는 경우이다.

snot 를 준비하기 위해 **/etc/snort/snort.conf** 파일을 snot 디렉토리로 복사한 후 하나 또는 그 이상의 규칙 파일들을 복사한 **snort.conf** 파일의 끝부분에 **cat** 해주면 된다:

cat /etc/snort/backdoor.rules >> snort.conf

그 후 한 컨솔에서 **tail -f /var/log/messages** 을 실행시키고 동시에 다른 컨솔에서 테스트를 해야 한다.

snortd initscript 에서 인터페이스 이름에 **lo** 를 사용했다면 다음과 같이 snot 를 실행시킬 수 있다.

./snot -r snort.conf -d localhost -n 5

이 명령은 snot 에게 복사한 **snort.conf** 를 사용하며, 목적지는 **localhost** 이고 너무 많은 경보를 유발하지 않도록 이를 최대 5로 제한하라고 한다.

추가적인 변수를 무시하라고 하는 약간의 메시지를 아마도 받을 것이다. 이는 snot 가 snort 1.8 에 도입된 새로운 변수를 다룰 수 없기 때문으로 허둥되지 말고 그저 메시지를 무시해라. snot 는 잘 실행되고 있다.

/var/log/messages 에서 약간의 snort 경보를 볼 수 있다.

```
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher ac-
cess: 192.168.213.151:6969 -> 127.0.0.1:3170
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher ac-
cess: 192.168.213.151:6969 -> 127.0.0.1:3170
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher ac-
cess: 192.168.155.231:6969 -> 127.0.0.1:57580
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher ac-
cess: 192.168.155.231:6969 -> 127.0.0.1:57580
Sep 10 18:22:33 ids01 snort[1536]: <lo> Deep Throat ac-
cess: 192.168.170.42:2140 -> 127.0.0.1:60521
```

비슷한 경보를 얻는다면 좋다. 그렇지 않다면 위와 유사한 결과를 얻을 때까지 설정을 조사하기 바란다.

이제 **/etc/snort/snort.conf** 를 편집해 **INTERFACE** 변수에 정확한 값을 넣은 후 snort 를 재시작해야 한다.

4.2. MySQL 설정하기

Snort 가 MySQL 에 경보를 보낼 수 있도록 하기 위해서는 우선 MySQL 을 설치해야 한다. 대부분의 리눅스 배포판에는 사용할 수 있는 MySQL 패키지가 있으며 따라서 이를 이용하면 된다. 그렇지 않다면 아마도 http://www.mysql.org/ 에서 타르볼을 다운로드받아 스크래치로부터 이를 컴파일 및 설치해야 할 것이다. 이를 설치하기 위해서 MySQL 에 포함되어 있는 문서를 보라.

MySQL 데몬을 실행시킨 후 (레드햇의 경우 RPM 을 설치한 후 **/etc/rc.d/init.d/mysql start** 를 실행시킨다) snort 데이타베이스를 초기화시켜야 한다. 이는 다음 절에 기술되어 있다:

```
[root@ids01 /root]# mysql -u root
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 133 to server version: 3.23.32

Type ’help;’ or ’\h’ for help. Type ’\c’ to clear the buffer

mysql>create database snort;
Query OK, 1 row affected (0.00 sec)

mysql> connect snort
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Connection id:    139
Current database: snort

mysql> status
--------------
mysql  Ver 11.12 Distrib 3.23.32, for redhat-linux-gnu (i386)

Connection id:          139
Current database:       snort
Current user:           root@localhost
Current pager:          stdout
Using outfile:          ”
Server version:         3.23.32
Protocol version:       10
Connection:             Localhost via UNIX socket
Client characterset:    latin1
```

```
Server characterset:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 1 day 2 hours 6 min 21 sec

Threads: 14  Questions: 4272  Slow queries: 0  Opens: 58  Flush ta-
bles: 1  Open tables: 18 Queries per second avg: 0.045
--------------

mysql> grant CRE-
ATE,INSERT,SELECT,DELETE,UPDATE on snort.* to snort@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

필수 데이타베이스 테이블 구조를 생성시키기 위해 본래의 타르볼 또는 저자의 RPM의 contrib 에서 찾을 수 있는 **create_mysql** 스크립트를 사용한다.

[root@ids01 /root]# mysql -u root snort < ./contrib/create_mysql

데이타베이스에 대해 사용자아이디/패스워드 쌍을 추가하고 **xxxx** 를 각자의 환경에 적합한 패스워드로 변경하는 것을 기억해야 할 것이다.

```
[root@ids01 /root]# mysql -u root mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 148 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> insert into user (User,Password) values('snort',PASSWORD('xxxx'));
Query OK, 1 row affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

편의를 위해 snort 타르볼과 저자의 RPM 의 contrib 에 있는 약간의 특별한 테이블을 다음 명령을 사용하여 추가해라

zcat snortdb-extra.gz | mysql -u root snort

ACID 의 저장 특징을 사용하려면 **snort** 데이타베이스를 정의했던 방법과 동일하게 다른 데이타베이스 **snort_archive** (또는 원하는 다른 이름의 데이타베이스) 를 생성해야 할 것이다.

지금부터 데이타베이스는 **/etc/snort/snort.conf** 에서 활성화시킬 수 있는 snort 의 database 출력 모듈을 통해 언제든지 기록에 사용될 수 있다.

4.3. ADODB 설정하기

ADODB 는 ACID 의 필수 부분으로 ACID 와 같은 PHP 기반 프로그램에 대해 데이타베이스 연결을 지원한다.

웹 서버에 사용할 수 있는 디렉토리에 ADODB 를 설치해라. 레드햇의 경우 이는 **/var/www/html/adodb/** 이다.

ADODB 버전 1.31 은 **adodb.inc.php** 에 버그가 있는데 더욱 새로운 버전에서도 버그가 존재할 수도 있다. 각자의 로컬 요건을 반영하기 위해 40번째 줄의 경로를 변경시켜야 할 것이다. **dirname()** 명령을 완전히 지우는 것이 필수적이며 따라서 다음과 같다:

```
 if (!defined('_ADODB_LAYER')) {
        define('_ADODB_LAYER',1);

        define('ADODB_FETCH_DEFAULT',0);
        define('ADODB_FETCH_NUM',1);
        define('ADODB_FETCH_ASSOC',2);
        define('ADODB_FETCH_BOTH',3);

        GLOBAL
                $ADODB_vers,            // 데이타베이스 버전
                $ADODB_Database,        // 사용된 마지막 데이타베이스 드라이버
                $ADODB_COUNTRECS,       // 반환된 레코드의 카운트 넘버 - 조회
를 느리게 한다
                $ADODB_CACHE_DIR,       // 캐시 레코드셋에 대한 디렉토리
                $ADODB_FETCH_MODE;      // DEFAULT, NUM, AS-
SOC or BOTH. Default follows native driver default...

        $ADODB_FETCH_MODE = ADODB_FETCH_DEFAULT;
        /**
         * 아래의 값을 이 파일이 놓이는 디렉토리로 설정해라
         * ADODB_RootPath 는 ADODB_DIR 로 이름이 변경되었다
         */
        if (!de-
fined('ADODB_DIR')) define('ADODB_DIR','/var/www/html/adodb');
```

이것이 ADODB 에 행해져야 하는 모든 것이다.

4.4. PHPlot 설정하기

PHPlot 을 다운로드한 후 웹 서버가 인식할 수 있는 디렉토리에 패키지를 놓아라. 레드햇에서 이는 **/var/www/html/phplot/** 이다. 여기서는 설정할 것이 아무 것도 없다.

4.5. ACID 설정하기

앞에서 말했듯이 ACID 는 정확히 작동하기 위해 몇개의 추가적인 프로그램이 설치될 필요가 있다. MySQL 버전 3.23 이상의 데이타베이스 시스템, PHP 모듈 **mod_php** 이 있는 **apache** 와 같이 PHP 4.0.2 이상을 지원하는 웹 서버와 ADODB 버전 0.93 이 필요한 반면 그래픽 라이브러리 **gd** 버전 1.8 이상과 PHPlot 버전 4.4.6 이상은 선택이지만 추천된다. **apache**, PHP 모듈 및 **gd** 는 모든 리눅스 배포판에 늘 포함되어 설치되어 있기때문에 이 문서에서는 이들을 다루지는 않는다.

Snort 1.8 이상에 대해서는 적어도 ACID 0.9.6b13 이 필요할 것이다. ACID 는 저자의 RPM 의 contrib 에 있지만 ACID 는 빠르게 개발되고 있기 때문에 이전 버전일 수도 있다. 따라서 더욱 새로운 버전이 존재하는지 늘 ACID 의 홈페이지를 보아야 한다.

ACID 를 **/var/www/html/acid** 와 같이 웹 서버에서 인식할 수 있는 디렉토리에 설치해라.

/var/www/html/acid/acid_conf.php 에서 각자의 환경에 맞도록 약간의 변수들을 편집해야 할 것이다.

우선 변수 **DBtype** 에 데이타베이스 타입을 정의해라. 다음 **alert_*** 와 **archive_*** 변수들을 정의해라.

ChartLib_path 에 PHPlot 에 대한 경로를 정의해라. 이 문서에서는 **/var/www/html/phplot** 이다.

정의해야 할 마지막 변수는 **portscan_file** 로 snort 의 포트스캔 로그파일의 절대 경로와 파일명을 정의해라.

모든 다른 변수들은 당분간은 충분하다. 물론 요구에 적합하게 이들을 편집할 수 있다.

다음은 저자가 사용한 config 이다:

```
<?php

$ACID_VERSION = "0.9.6b15";

/* DB 추상 라이브러리에 대한 경로
 *  (주의: 디렉토리 다음에 백슬래쉬를 포함하지 마라)
 *   e.g. $foo = "/tmp"      [OK]
 *        $foo = "/tmp/"     [WRONG]
 *        $foo = "c:\tmp"    [OK]
 *        $foo = "c:\tmp\"   [WRONG]
 */
$DBlib_path = "/var/www/html/adodb";

/* 기본적인 경보 데이타베이스 타입
```

```
 *
 *  MySQL       : "mysql"
 *  PostgresSQL : "postgres"
 */
$DBtype = "mysql";

/* Alert DB 연결 변수들 */
 *   - $alert_dbname   : Snort 경보 DB 의 MySQL 데이타베이스 이름
 *   - $alert_host     : DB 가 저장되는 호스트
 *   - $alert_port     : DB 에 접속하는 포트
 *   - $alert_user     : DB 사용자
 *   - $alert_password : DB 사용자의 패스워드
 *
 *  이 정보는 Snort 데이타베이스 출력 플러그인 설정에서 수집될 수 있다.
 */
$alert_dbname   = "snort";
$alert_host     = "localhost";
$alert_port     = "";
$alert_user     = "snort";
$alert_password = "xxxx";

/* Archive DB 연결 변수들 */
$archive_dbname   = "snort_archive";
$archive_host     = "localhost";
$archive_port     = "";
$archive_user     = "snort";
$archive_password = "xxxx";

/* 사용할 DB 연결 타입
 *   1  : 영속적인 연결을 사용한다 (pconnect)
 *   2  : 통상적인 연결을 사용한다 (connect)
 */
$db_connect_method = 1;

/* 그래픽 라이브러리에 대한 경로
 *  (주의: 디렉토리 다음에 백슬래쉬를 포함하지 마라)
 */
$ChartLib_path = "/var/www/html/phplot";

/* 차트의 파일 포맷 ('png', 'jpeg', 'gif') */
$chart_file_format = "png";

/* 차트의 디폴트 컬러
 *    - $chart_bg_color_default    : 차트의 백그라운드 컬러
 *    - $chart_lgrid_color_default : 차트의 그리드라인 컬러
 *    - $chart_bar_color_default   : 차트의 막대/선 컬러
 */
$chart_bg_color_default     = array(255,255,255);
$chart_lgrid_color_default  = array(205,205,205);
```

```
$chart_bar_color_default    = array(190, 5, 5);

/* 기준 요소당 최대 줄수 */
$MAX_ROWS = 20;

/* 모든 조회 결과에 대해 표시하기 위한 줄수 */
$show_rows = 50;

/* 스냅샷동안에 반환되는 아이템수
 *  Last _X_ # of alerts/unique alerts/ports/IP
 */
$last_num_alerts = 15;
$last_num_ualerts = 15;
$last_num_uports = 15;
$last_num_uaddr = 15;

/* 스냅샷동안에 반환되는 아이템수
 *  가장 자주 일어난 유일한 alerts/IPs/ports
 */
$freq_num_alerts = 5;
$freq_num_uaddr = 15;
$freq_num_uports = 15;

/* 조회 결과를 표시할 때 사용되는 스크롤 버튼의 수 */
$max_scroll_buttons = 12;

/* 디버그 모드 - 얼마나 많은 디버깅 정보가 보여지는 지를 결정
 * 타이밍 모드 - 타이밍 정보 표시
 * SQL trace 모드 - SQL 문을 기록
 *   0 : 특별한 정보 없음
 *   1 : 디버깅 정보
 *   2 : 확장된 디버깅 정보
 *
 * HTML no cache - no-cache 지시가 브라우저로 보내지는 지를 결정
 *                 익스플로러에 대해서는 1 이다
 *
 * SQL trace 파일 - SQL trace 를 기록하기 위한 파일
 */
$debug_mode = 0;
$debug_time_mode = 1;
$html_no_cache = 1;
$sql_trace_mode = 0;
$sql_trace_file = "";

/* Auto-Screen refresh
 * - Refresh_Stat_Page - 어느 정도의 통계 자료 페이지가 새롭게 보여져야 하는가?
 * - Stat_Page_Refresh_Time - refresh 간격 (초단위)
 */
$refresh_stat_page = 1;
```

```
$stat_page_refresh_time = 180;

/* 경보에 대해 처음/이전/마지막 타임스탬프를 표시하는가 또는
 * 유일한 경보 목록에 대해 그저 처음/마지막 타임스탬프를 표시하는가
 *    1: yes
 *    0: no
 */
$show_previous_alert = 1;

/* Sets maximum execution time (in seconds) of any particular page.
 * 주의: 이는 PHP 설정 파일 변수 max_execution_time 을 무시한다.
 *       따라서 스크립트가 총 ($max_script_runtime + max_execution_time) 초동
안 실행될 수 있다
 */
$max_script_runtime = 180;

/* IP 주소에 대한 기준이 검색 스크린에 어떻게 입력되어야 하는가?
 *   1 : 각 옥텟이 별도 필드이다
 *   2 : 전체 주소가 하나의 필드이다
 */
$ip_address_input = 2;

/* IP 를 FQDN (Fully Qualified Domain Name) 로 바꿀 것인가 (어떤 조회에 대해)?
 *    1 : yes
 *    0 : no
 */
$resolve_IP = 0;

/* summary stats 가 모든 조회 결과 페이지에 대해 계산되어야 하는가
 * (이 옵션을 허가하는 것은 페이지 로딩 시간을 느리게 할 것이다)
 */
$show_summary_stats = 1;

/* DNS 캐시 유효기간 (분단위) */
$dns_cache_lifetime = 20160;

/* Whois 정보 캐시 유효기간 (분단위) */
$whois_cache_lifetime = 40320;

/* Snort spp_portscan 로그 파일 */
$portscan_file = "/var/log/snort/portscan.log";

/* 이벤트 캐시 자동 갱신
 *
 *  이벤트 캐시가 모든 페이지 로그에 대해 검증되고 갱신되어야 하는가?
 *  그렇지 않다면 캐시는 'cache and status' 페이지로부터 명시적으로 갱신되어
야 할 것이다.
 *
 *  주의: 이 옵션을 허가하는 것은 캐시되지 않은 많은 경보가 있을때 페이지 로딩 시간을
```

```
 *  상당히 느리게 할 것이다. 그러나 이는 단지 한번 겪는 불편이다.
 *
 *   1 : yes
 *   0 : no
 */
$event_cache_auto_update = 1;

/* 외부 Whois 조회에 대한 링크 */
$external_whois_link = "http://www.samspade.org/t/ipwhois?a=";

?>
```

저자가 패스워드로 **xxxx** 를 사용했는지 의아해할지 모른다. 좋다 여러분은 세상의 모두가 사용할 수 있는 패스워드를 좋아하는가?

브라우저로 ACID 를 처음 불러올 때 선택된 데이타베이스에 ACID 지원을 설치해야 한다는 지시를 얻을 것이다. **Setup** 를 클릭해라 그러면 ACID 가 데이타베이스에 필수 엔트리를 생성할 것이다. 모든 것이 정확히 설정되면 지금 데이테베이스에 있는 모든 정보를 얻을 것이다. 일반적으로 이때에는 아무 것도 없을 것이다.

snot(윗 절을 보라) 또는 **nmap** (http://www.nmap.org/, 매우 많은 능력을 갖고 있는 포트스캐너) 또는 **nessus** (http://www.nessus.org/, 시스템의 취약성을 찾기 위한 스캐너) 를 사용해 어떤 snort 규칙들을 유발해 보라.

이것이 일어날 때마다 바로 ACID 에서 모든 경보를 얻을 것이다.

4.6. SnortSnarf 설정하기

SnortSnarf 는 데이타베이스대신 snort 의 로그파일을 분석하는 다른 도구이다.

SnortSnarf 를 원하는 디렉토리에 tar 한 후 설치해라. 저자의 경우는 **/opt/SnortSnarf** 에 설치했다.

필수 펄 모듈을 SnortSnarf 에 사용할 수 있도록 **/opt/SnortSnarf/Time-modules/lib/Time** 를 **/opt/SnortSnarf/include/SnortSnarf/Time** 에 복사해라.

다음 라인들을 웹 서버의 **cgi-bin** 에 복사해라 (예, **/var/www.cgi-bin/**):

```
/opt/SnortSnarf/cgi/*
/opt/SnortSnarf/include/ann_xml.pl
/opt/SnortSnarf/include/web_utils.pl
/opt/SnortSnarf/include/xml_help.pl
```

SnortSnarf 에서 사고에 대한 메모를 생성할 수 있는 annotation 특징을 사용하고 싶다면 우선 **/var/www/html/SnortSnarf/annotations** 디렉토리를 생성해야한다. 다음

/opt/SnortSnarf/new-annotation-base.xml 를 **/var/www/html/SnortSnarf/annotations** 에 복사한 후 **/opt/SnortSnarf/utilities** 에 있는 다음 명령을 실행시킨다.

./setup_anns_dir.pl -g apache /var/www/html/SnortSnarf/annotations

/var/www/html/SnortSnarf/annotations 의 권한을 검사해서 다음과 같도록 해라.

```
[root@ids01 SnortSnarf]# ll -a /var/www/html/SnortSnarf/annotations/
total 16
drwxrwx---    2 root     apache       4096 May 23 14:31 .
drwxr-xr-x    8 root     root         4096 May 23 14:17 ..
-rw-r--r--    1 apache   apache        478 May 23 14:31 new-annotation-
base.xml
```

저자는 다루기 힘든 @INC 에러 (펄에 대해 더욱 많은 지식을 갖고 있는 누군가가 이 에러들을 없애는 방법에 대한 힌트를 저자에게 줄 수 있다) 들을 없애기 위해 **/opt/SnortSnarf/snortsnarf.sh** wrapper 스크립트를 만들었다. 저자는 오전 6 시부터 오후 6 시까지 매시간 cron 을 통해 **/opt/SnortSnarf/snortsnarf.sh** 을 실행시키고 있다.

저자의 crontab 엔트리는 다음과 같다:

```
# 오전 6시부터 오후 6시까지 매시간 SnortSnarf 통계 자료를 생성한다
0 6,7,8,9,10,11,12,13,14,15,16,17,18 * * * /opt/SnortSnarf/snortsnarf.sh
```

SnortSnarf 는 다섯개의 로그파일 **/var/log/messages*** 을 분석하기 위해 호출되며 생성된 HTML 파일들을 **/var/www/html/SnortSnarf** 디렉토리내에 넣은 후 위에 기술된 annotation 특징을 이용한다.

다음은 **/opt/SnortSnarf/snortsnarf.sh** 파일 내용이다:

```
#!/bin/sh
# @INC 문제를 없애기 위해 crontab 과 함께 사용하기 위한 wrapper
# Poppi, 22.05.2001
cd /opt/SnortSnarf
./snortsnarf.pl -d /var/www/html/SnortSnarf -
db /var/www/html/SnortSnarf/annotations/new-annotation-base.xml -dns -
rulesfile /etc/snort/snort.conf -
ldir "file://var/log/snort/" /var/log/messages /var/log/messages.1 /var/log/messages.2 /var/log/messages.3 /var/log/message
```

snortsnarf.sh 을 실행시켜서 SnortSnarf 를 테스트하고 브라우저로 **/var/www/html/SnortSnarf/** 를 조사해보라.

4.7. Arachnids_upd 설정하기

경고: 어떠한 암호화 또는 인증없이 규칙들을 자동 갱신하는 것은 공격자가 IDS 에서 존재가 발각되지 않을 수 있도록 규칙들이 손상될 수 있기때문에 백도어를 만들 수 있다. 따라서 이를 조심해서 사용해야 한다.

다른 문제는 wwww.whitehats.com 이 종종 오프라인이라는 것이며 따라서 어떠한 규칙들도 다운로드받을 수 없다.

arachnids_upd 패키지를 선택한 디렉토리에 압축해제해라. 저자는 **/opt/arachnids_upd/** 를 사용한다.

Snort 1.8 이상에 대해 **/opt/arachnids_upd/arachnids_upd.pl** 를 편집해서 다운로드할 파일명을 변경해야 할 것이다:

```
     my $url = "http://www.whitehats.com/ids/vision18.rules.gz";   # De-
fault URL.
```

Arachnids_upd 는 **wget** 를 사용하기 때문에 각자의 시스템에 설치되어야 하며 인터넷 연결과 작동되도록 설정되어야 한다.

사용자 인증을 하는 프락시 서버를 통한 연결의 경우 .wgetrc 예제 파일이다:

```
     proxy_user = user
     proxy_passwd = xxxx
     http_proxy = <proxy>:<port>
     ftp_proxy = <proxy>:<port>
     use_proxy = on
```

<proxy> 와 <port> 를 각각 각자의 프락시 이름 또는 ip 주소와 프락시가 사용하는 포트 넘버로 대체해라. 프락시를 사용하지 않는다면 이 엔트리들 중 아무것도 필요하지 않다.

저자는 새로운 규칙을 얻고 **/etc/snort/snort.conf** 의 정의에 맞게 변수 이름 **vision.rules** 을 변경하며 새로운 규칙들이 유효하도록 snort 를 재시작시키는 쉘 스크립트를 작성했다.

```
#!/bin/sh
# arachnids_upd.pl 을 사용하여 vision.rules 파일을 정확히 갱신하기 위한 스크립트
# Poppi 22.05.2001

# 새로운 규칙을 얻는다 (~/.wgetrc 가 인터넷에 접속할 수 있도록 설정되어야 한다)
/opt/arachnids_upd/arachnids_upd.pl -o /opt/arachnids_upd/vision.rules -
b /opt/arachnids_upd/rules.backup/ -c

# /etc/snort/snort.conf 파일내에 사용된 변수이름으로 변경하고 새로운 파일을 정확
한 곳으로 복사한다
cat /opt/arachnids_upd/vision.rules | sed s/EXTERNAL/EXTERNAL_NET/g | sed s/INTERNAL/HOME_NET/g > /etc/snort/vision.rules
```

```
# 규칙들이 유효하도록 snort 를 재시작한다
/etc/rc.d/init.d/snortd restart
```

Arachnids_upd 는 또한 다운로드하는 동안 **vision.rules** 내의 규칙들을 삭제할 수 있기 때문에 원한다면 **/opt/arachnids_upd/arachnids.ignore** 를 편집해 무시되어야 하는 IDS 넘버를 넣을 수 있다.

```
# 규칙들이 금지되어야 하는 IDS 넘버를 넣어라.
# 한줄에 한개의 IDS 넘버.

# Examples:

1       # Ignore IDS1
2       # Ignore IDS2
3       # Ignore ISD3

# I think you get it now :)
```

4.8. Swatch 설정하기

Swatch 는 모든 로그파일을 다루는 우수한 패키지로 공격과 관련된 어떠한 것이 로그파일에 기록된다면 경고해주도록 정규 표현을 이용해 설정될 수 있다.

Swatch 는 다음 펄 모듈이 설치되어야 한다:

```
perl-TimeDate
perl-Date-Calc
perl-Time-HiRes
perl-File-Tail
```

Swatch 는 저자의 소스 RPM http://www.lug-burghausen.org/projects/Snort-Statistics/swatch-3.0.2-1.src.rpm 과 함께 http://www.lug-burghausen.org/projects/Snort-Statistics/swatch-3.0.2-1.noarch.rpm 에서 RPM 으로 얻을 수 있다.

Swatch 는 **/etc/swatch/swatch.conf** 설정 파일을 통해 설정된다.

저자는 원래 swatch 패키지의 약간의 다른 예와 함께 밑에 보이는 snort 메시지와 에러에 대한 두 규칙을 포함하고 있는 데모 **swatch.conf** 를 갖는 소스 RPM 을 작성하고 있다.

```
# global swatch.conf file
# * Poppi, 30.04.2001
# - 최초버전
```

```
#
# * Poppi, 08.06.2001
# - 에러 지원을 추가하였다t; 반드시 snort 보다 swatch 를 먼저 실행시켜라;)
#
# Poppi, 19.09.2001
# - 동일 사건에 대해 너무 많은 경보를 얻지 않도록 throttle 옵션을 추가하였다

# PID 가 있는 정상적인 snort 메시지
# 10초동안 두개가 있는 경보를 제거해라 (예 pings)
watchfor /snort\[/
        bell
        exec /etc/snort/snort-check $0
        throttle 00:00:10

# snort 에러 메시지는 [!] indicator 가 있을 수도 있고 없을 수도 있다
watchfor /snort: (\[\!\])* ERROR/
        bell
        exec /etc/snort/snort-check $0
```

첫번째 규칙은 출력 모듈 **alert_syslog** 를 통해 생성된 모든 경보를 얻기 위한 것이고 두번째 규칙은 (규칙 파일에서의 에러들과 같이) 무엇인가 잘못된 경우 snort 가 생성하는 모든 에러 메시지들을 얻기 위한 것이다.

두 규칙은 pc 에 신호를 보내 알리며 (만약 센서가 운영자가 없는 방에서 사용되고 있다면 아무 의미도 없다) 주어진 사람에게 경고하기 위해 앞에서 기술한 **snort-check** 스크립트를 이용한다. **&0** 내에 swatch 는 swatch 를 유발한 로그파일내의 모든 엔트리를 제공한다.

Swatch 는 snort 보다 먼저 시작되어야 한다. 정확한 **chkconfig** dates 를 갖는 swatch 자신의 initscript 를 생성하는 대신 저자는 이를 **/etc/rc.d/init.d/snortd** 에 포함하였다. 왜냐하면 저자가 사용하는 swatch 의 의존성때문에 이렇게 하기로 결정했다. 저자는 좋은 방식이 아니며 swatch 부분이 비교적 쉽게 initscript 내에서 작성될 수 있음을 알고 있다. 아마도 추후에 이를 변경할 것이다.

5. 보안 문제

Snort 는 자신의 userid/group 쌍인 **snort/snort** 하에서 실행된다. 이는 아직껏 수정되지 않은 모든 버퍼 오버플로우가 단지 snort 사용자가 갖는 권한만을 갖음을 확인한다. 이것이 충분하지 않은 사람들의 경우는 snort 의 커맨드 라인 옵션 **-t** 를 사용해 chroot 된 환경을 사용할 수도 있다. 그러나 저자에게 이 방법을 묻지 말아라. 저자는 해본적이 없으며 아마도 하지 않을 것이다.

모든 보안 관련 시스템에서와 같이 필요이상의 서비스를 허용하지 마라. 모든 리눅스 배포판의 표준 설치를 한 경우 배포판이 예전 inetd 를 사용하고 있다면 **/etc/inetd.conf** 를 **xinetd** 기

반 시스템이라면 **/etc/xinetd.d/*** 를 조사해서 시스템에 실제로 필수적이지 않은 모든 서비스를 금지해야 한다. 예를 들어 텔넷 사용을 원하지 않는다면 이를 ssh 로 대체해라.

또한 레드햇과 같이 System V 기반 시스템에서 **/etc/rc.d/init.d/*** 에 있는 initscript 들을 조사해라. **nfs** 와 **portmap** 와 같이 사용하지 않는 서비스가 있다면 해당 패키지를 완전히 제거해라.

Security-HOWTO, 시스템 관리자 가이드 또는 네트워크 관리자 가이드 와 같은 많은 보안 관련 문서들을 읽어야 한다.

또는 http://www.securityfocus.com/, http://www.linuxsecurity.org/ 또는 http://www.insecure.org/ 와 같은 여러 보안 관련 웹 사이트를 조사해라.

6. 도움얻기

결국 자신의 문제를 해결할 수 없어 누군가의 도움을 필요로 할 수도 있다. 가장 효과적인 방법은 가까운 곳 또는 가장 가까운 리눅스 사용자 그룹내의 누군가에게 묻거나 웹을 검색하는 것이다.

그러나 우선적으로 http://www.snort.org/ 과 snort 메일링 리스트를 살펴보도록 해라. 저자는 이곳의 사람들에게 매우 많은 도움을 받았다.

또다른 방법은 매우 매우 많은 뉴스그룹중에서 Usenet New 에 질문을 하는 것이다. 문제는 질문에 대한 답변이 없을 정도로 이곳은 매우 많은 사람들이 있고 혼란스럽다.

어느 곳에 질문을 하던지 조리있게 질문하는 것이 중요하다 그렇지 않으면 여러분의 질문은 신중히 다뤄지지 않을 것이다. 단순히 **snort** 가 작동하지 않는다 라고 말하는 것은 도움이 되지 않으며 대신 불쾌한 답변만 더욱 증가할 것이다. 물론 운좋다면 누군가가 설명해 줄 것이다.

대신 사람들이 여러분을 도울 수 있도록 약간은 세부적으로 문제를 기술해라. 여러분이 예상하지 못했던 곳에 문제가 있을 수도 있다. 따라서 여러분의 시스템에 대해 다음 정보를 열거하라고 충고한다.

소프트웨어

- /etc/snort/snort.conf
- /etc/swatch/swatch.conf (사용하는 경우)
- /var/log/messages 내에서 단지 연관있는 엔트리들의 발췌
- 사용하는 리눅스 배포판 또는 운영체제와 버전
- 에러가 있는 소프트웨어 (버전 넘버 또는 날짜)

저자에게 직접적으로 질문할 수도 있다. 그러나 기억하기 바란다: 저자는 컴퓨터 작업외에 다른 일도 하고 있으며 그리 한가한 편이 아니니다. 저자는 거의 언제나 이메일에 답장할 것이

지만 약간 시간이 걸릴 수 있다. 또한 저자는 snort 메일링 리스트에 가입했으며 따라서 이를 통해 저자에게 연락할 수 있다.

7. 질문과 답변

This is just a collection of what I believe are the most common questions people might have. Give me more feedback and I will turn this section into a proper FAQ.

* *This part needs tweeking. As questions and answers arrive I will put them in here. Contributions are welcome.*

- Q:

  A:

(rest deleted.)