제6장 보 안

1. 보안의 개요

(1) 보안에 대하여
(2) 일반적인 침입 수법

2. X Window 보안

(1) 문제점
(2) 주요 공격 패턴
(3) 해결책

3. 보안 도구

(1) Tripwire
(2) Snort
(3) Nmap
(4) Nessus
(5) SSH(Secure Shell)

4. 최소한의 보안

(1) 계정 보안
(2) 시스템 내부 보안
(3) 네트워크 보안
(4) 기타 보안

1. 보안의 개요

(1) 보안에 대하여

1960~1970년대에 컴퓨터 보안이라는 말은 요즘처럼 주위에서 흔히 들을 수 있는 용어가 아니였습니다. 당시의 컴퓨터에도 나름대로 중요한 자료들이 들어 있었고, 그에 대한 위협도 없지는 않았지만 당시에는 컴퓨터의 가격이 비싸서 일반사람들이 휴대하기에는 현적으로 불가능 했고, 컴퓨터를 사용할수 있는 기관은 원래부터 상당한 수준의 보안을 추었기 때문에 일반인들에게 시스템 보안이라는 용어는 생소했습니다.

하지만, 1990년대를 지나 현재 2001년이 되면서 상황은 상상하기 힘들정도로 변하였고, byte단위였던 메모리가 Mbyte를 지나서 Gbyte로 향하고 있고, CPU는 이미 GHz 로 넘어가는 추세이고, 하드디스크 또한 Gbyte를 지나서 1~2년 안으로 Tbyte 로 넘어갈 것입니다.

성능의 상승곡선과는 반대로 가격은 계속 하락하는 추세입니다. 이로 인해 일반 컴퓨터 사용자가 많아졌으며 국가 기관이나 연구소뿐만 아니라 일반 기업에서도 자료를 컴퓨터에 저장해 두는 것이 일반화되었습니다. 더욱이 인터넷의 보급은 특정 컴퓨터에 접근할 수 없는 사람들로 하여금 멀리 떨어진 곳에서 그 컴퓨터에 침투하여 정보를 빼내거나 망가뜨리는 것을 가능하게 만들었습니다.

예전에는 '패스워드를 책상에 붙여두지 말 것.', '전화번호나 주민등록번호등의 번호를 패스워드로 사용하지 말 것.' 이라는 말로 적절한 보안 조치가 취해졌다고 할 수 있었는지 모르지만 이제는 이 말로는 더 이상 보안을 보장할 수 없게 되었습니다. 위에서 언급하였듯이 컴퓨터의 성능이 좋아짐으로 하여, 밝은면과 어두은면의 이중성이 생겼습니다. 밝은면으로는 빠른 연산으로 인한 고성능의 프로그램들이 실생활에 많은 도움을 주고 있고, 어두은면 역시 고성능의 컴퓨터로 인해 더욱 빠른 Cracking(이후 다른 시스템에 침투, 데이터의 변조, 파괴등의 행위를 Craking이라 간주하고, 이 Cracking을 하는 사람을 Cracker라 정의하겠습니다)이 가능해졌다는 것입니다. 시스템 보안과 시스템 Cracking은 모두 시스템이라는 장비를 사용하는 행위이므로, 시스템의 발전은 곧 보안과 Cracking의 발전이라고 봐도 될 것입니다.

한명의 도둑을 열 경찰이 잡기 힘들다는 말처럼, 여러 보안 정책이 집요하게 파고드는 한 Cracker를 당하지 못합니다. 완벽한 보안도 완벽한 Cracking 또한 있을수 없습니다. 좋은 보안툴은 Cracking 툴로 이용할수도 있고, 반대로 Cracking 툴을 보안에 이용할 수도 있습니다. 앞으로의 미래는 지금으로서는 상상도 하기힘들정도의 고성능의 시스템들이 나올 것입니다. 이말은 더 막강한 보안툴과 더욱 뛰어난 Cracking 툴이 나온다는 말과도 같을 것입니다. 그러기에 보안이란 분야는 시스템에서 빼 놓을수 없는 용어가 될 것입니다.

현대 사회에 있어 정보는 중요한 재산입니다. 컴퓨터 및 네트워크 장비의 발달에 힘입어 과거에는 상상하기도 힘든 정도의 엄청난 양의 정보들이 실시간으로 처리, 보관, 전송되고 있습니다. 뿐만 아니라 컴퓨터 통신망의 확대는 수많은 사용자들에게 편리하고 다양한 정보를 제공할 수 있게 하였으며 고급 정보의 전송 또한 날로 증가하고 있습니다. 그러나 이와 같이 정보가 밀집화되고 통신망의 이용이 증가함에 따라 이에 연결된 정보 시스템들은 비단 전문적인 공격자들뿐만 아니라 단순한 호기심에 의한 공격자들까지 여러 공격자들의 표적이 되고 있습니다. 통신망이 거대해지고 정보 자산의 가치가 높아지며 사용자 집단이 다양해지는 것은 한편으로는 공격자의 수가 늘어나고 공격 기회가 많아지며 공격 동기가 높아지고 그리고 공격에 의한 피해의 규모가 커진다는 것을 의미합니다. 따라서 정보 시스템에 대한 보안은 시급한 문제라 하지 않을 수 없습니다.시스템의 발전과 지식의 발전으로 보안 도구들도 많이 있지만, 바이러스등과 같은 악성 코드들도 많이 있습니다.

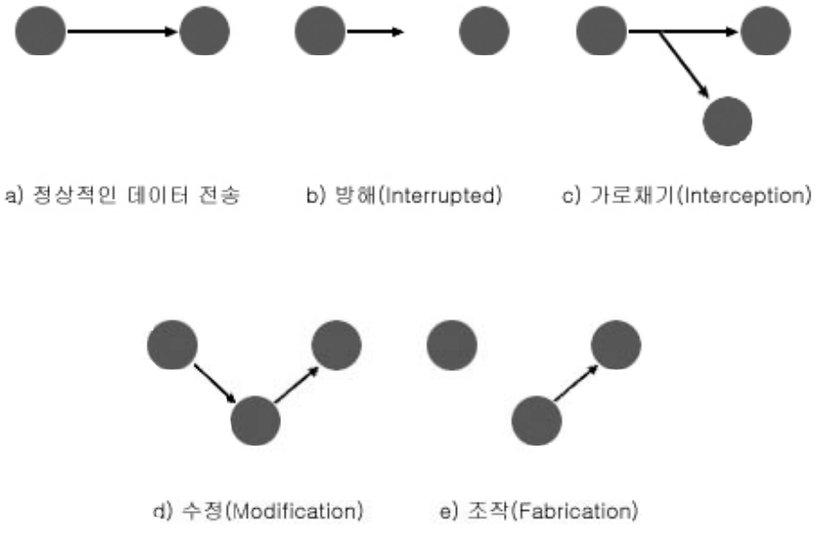
예전의 서버/클라이언트 식의 시스템에서 개인이 저장매체를 관리하는 체계인 지금 각각의 개인 시스템은 Cracker들의 목표대상으로 존재할 수 있습니다. 그러기에 개인이 보안에 신경을 쓰지 않는다는 것은 개인 정보를 포기한다는 의미와도 같을 수 있습니다. 현재는 리눅스 바이러스가 그리 많이 존재 하지 않고, 혹자는 리눅스에서는 바이러스가 존재할 수 없다는 말도 합니다. 하지만 이건 어디까지나 지금의 상황에서 말한 것 뿐입니다. 그 혹자의 리눅스 바이러스가 없다는 생각은 어디 까지나 리눅스를 서버용으로 생각했을 때의 일이고, 만약 리눅스를 클라이언트로 많이 쓰는 시대가 오면 개인개인은 root(슈퍼유저)로 모든 일을 하게 될 것입니다. 물론 일반계정으로 사용을 한다면야 그 혹자의 말처럼 리눅스 바이러스의 전파는 쉽지 않겠지만, 그건 쉽지 않을 것입니다. 이 글을 쓰는 사람도 root로 행하는 일이 많기 때문입니다. 그렇게 되면 리눅스 바이러스는 충분이 가능한 일입니다.

이런 위험을 피하기 위해서 개인이나 기업은 그 상황에 맞는 보안 정책을 책정하고 수행해야 할 것입니다. 이런 보안 정책을 객관적으로 판단하기 위해서 여러 단체에서 보안 정도를 판별하는 기준을 세워 놓았습니다. 그 가운데 가장 권위 있는 것이 Trusted Computer System Evaluation Criteria(TCSEC)입니다. 흔히 'The orange Book' 이라고 부르며 *http://www.nsi.org/computer/govt.html* 에서 그 전문을 볼 수 있습니다. 위처럼 객관적인 정책을 통해 세울수도 있고, 자신에게 맞는 알맞은 정책을 세울수도 있겠습니다. 가장 기본적인 보안 정책이라 할 수 있는 몇가지는, 우선 필요 없는 데몬들을 서비스에서 내리는 것입니다. 특히 얼마전까지 유명세를 올렸던 bind, rpc.statd의 데몬들은 필요가 없으면 반드시 내리던지 보안패치가 있다면 패치를 돌리고 서비스에 올려야 할 것입니다. 이것만으로도 상당한 보안등급을 올릴수 있을 것입니다. 서비스를 올리고 내리는 것은 /etc/rc.d/init.d/에서 할수도 있고, 쉘에서 ntsysv를 입력하면 각종 서비스를 관리할수 있습니다. 그리고, 보안패치등의 뉴스를 항상 접하면서 보안에 관계된 패치를 개을리 해서는 안될 것입니다. 보안 패치는 앞으로 wowlinux.com에서도 신속히 대응해서 패치를 빠르게 올릴것입니다. 솔직히 이 두가지만 제대로 해준다고 해도 거의 완벽한 정도의 보안정책일 것입니다. 다음은 일반적인 침입수법에 대해 논해보겠습니다.

(2) 일반적인 침입 수법

보안 파트에서 침입 수법을 설명한다는게 다소 의아해할수도 있지만, 보안과 Cracking은 서로 상반되지만 일맥이 상통이 되는 부분도 많고, 도둑을 잡으려면 우선 도둑이 되라는 말도있듯이 상대편에서 생

각을 해보아야 진정한 보안도 되지 않을까 해서 다뤄보겠습니다. Cracker들의 일반적인 침투 유형은 공격에 관계되는 호스트의 수와 그들 사이의 관계에 따라 그림 1과 같이 나눌 수 있습니다. 그림 1에서 각각 (a)는 공격이 없을 때의 정상적인 상태를 나타내고, (b)는 공격자가 정상적인 통신을 방해하는 경우, (c)는 공격자가 통신의 일부를 엿듣는 경우, (d)는 공격자가 통신 내용을 가로채어 위조된 통신문을 전송하는 경우, (e)는 통신이 없을 때 허위로 통신문을 전송하는 경우를 나타낸 것입니다.



[그림 VI-1]

(b)의 예는 서비스 거부(Denial Of Service)를 들수 있고, (c)의 예로는 엿듣기(sniffering), (d)와 (e)의 예로는 IP 주소 속이기(IP address spoofing), DNS 속이기(DNS spoofing)등의 각종 속이기(spoofing) 공격을 들 수 있습니다. 공격 유형은 다시 그 침입 수법에 따라 다음과 같이 나눌 수도 있습니다.

① 서비스 거부(Denial Of Service)

시스템의 정상적인 동작을 방해하는 공격으로, 최근 인터넷 웹사이트가 유명세를 타므로서 이 공격이 부각이 됐는데, 웹사이트로서는 서비스를 하는 것이 최고의 목표이고, 이 목표에 충분히 해를 끼칠수 있는 공격이기 때문입니다. 이제까지의 공격이 시스템에 대한 직접적인 침입을 시도하던 것에 비하여 이 공격은 목표 시스템에 짧은 시간에 큰 부하를 주므로써 시스템의 정상적인 작동을 방해하는 방법을 이용합니다. 하부 구조를 이용한 공격만큼이나 막기 어려운 공격 유형이지만 시스템을 파괴하지는 않으므로 복구는 상대적으로 간단하게 이루어질 수 있습니다.

② 훔쳐보기(Sniffering)

Sniffering은 말 그대로 정상적이 패킷을 가로채서 훔쳐보는 것입니다. password같은 것도 훔쳐낼수가 있습니다. 외부나 내부 모두 가능한 공격으로 빼내가는 정보에 따라 피해는 심할수도 있습니다. 이를 막기 위해서는 패킷을 암호화하여 보내는 방법을 쓰면 됩니다.

③ 신뢰받는 기계로 위장(transitive trust)

목표물이 신뢰하는 기계나 네트워크로 위장하는 수법으로 주로 옛 버전의 썬워크스테이션에 있던 허점을 이용한 공격으로, 사용자의 편의를 위하여 같은 네트워크 그룹 안의 기계들을 서로 신뢰하는 기계로 설정하여 암호없이 로그인하는 것을 허용하는 점을 이용하는 것이며 자신의 기계가 마치 신뢰받는 기계 중의 하나인 것으로 위장하여 목표 기계에 대한 접근권한을 얻는 방법이 이용됩니다. 이 공격도 역시 다른 공격을 위한 발판이 되는 공격이라 할 수 있습니다.

④ 부당 이용(exploits)

시스템이나 데몬들의 취약점을 이용하여 목표 시스템에 대한 접근 권한을 얻고 난 후 관리자 권한을 얻기 위해 시도하는 공격입니다. 예전에는 race condition, IFS환경변수등의 공격이 형태를 띄였으나 근래에는 대부분이 버퍼오버플로우(buffer overflow)를 이용하는 방법이 주류를 이루고 있습니다.

⑤ 자료 이용(data driven)

공격 프로그램(attack program), 트로이 목마(trojan horse), 뒷문(backdoor), 바이러스(virus)등이 이에 속합니다. 좀더 체계적인 공격을 위하여 공격 프로그램을 이용할 수도 있고, 트로이 목마 프로그램을 심어 두었다가 해당 시스템의 사용자가 그 프로그램을 실행하면 암호를 빼낸다던가 시스템을 파괴하던가 할 수도 이따. 또한 한 번 침입에 성공한 시스템에는 뒷문을 만들어 두어 다음 번 침입을 용이하게 하기도 합니다.

⑥ 하부 구조 이용(infrastructure)

프로토콜이나 시스템등 하부구조의 취약성을 이용한 공격입니다. IP주소 속이기가 대표적인 예로 이 공격을 막으려면 상당한 노력이 필요하며 방화벽을 이용하는 방법이 일반적입니다.
이러한 공격의 유형에 따라 여러 가지 침입 방법이 있을 수 있습니다. 그 중에서 가장 일반적인 침입의 순서를 정리하면 아래와 같습니다.

■ 호스트에 침입을 합니다. 주로 관리가 허술한 일반 사용자 계정을 이용합니다. 혹은 처음부터 데몬을 공격하여 데몬을 실행하고 있는 권한을 얻습니다. 만약 유저가 root(슈퍼유저)라면 이미 침입이 완

료가 된것이고, 그렇지 않다면 다음 단계로 진행합니다.

■ 프로그램이나 시스템의 버그를 이용하여 관리자(root)권한을 얻습니다. race condition, 버퍼오버플로우등의 방법을 이용합니다.

■ 뒷문을 설치하여 다음 번 침입을 쉽게 할 수 있도록 합니다. 뒷문을 통하면 다음에는 복잡한 과정없이 쉽게 관리자 권한을 얻을 수 있기 때문에 뒷문 설치는 침입자들에게 애용되는 절차중 하나입니다. 교묘하게 설치된 뒷문은 좀처럼 찾기 어려우며 요새는 rootkit이라는 프로그램을 이용하여, 관리자가 침입자를 알아내는 프로그램들을 바꿔치기 하여 안전하게 이용할수 있게 합니다. 점점 교묘한 방법이 개발되므로 이미 침입당한 시스템은 다시 설치하는 것이 차라리 간편한 방법일 수 있습니다. 따라서 한번 침입당한 시스템은 그 보안성을 심각하게 의심해 볼 필요가 있습니다.

■ 마지막으로 로그 감추기를 통하여 침입 흔적을 지웁니다. utmp, wtmp, lastlog등의 파일에 누가 언제 로그인 했는지에 대한 정보가 기록 되므로 이를 지웁니다. 위의 3개는 아주 기본적인 파일이기 때문에 관리자는 저것이외에 개인적인 로그 시스템을 갖추어 관리를 하는 것이 흔적을 찾기위해 아주 중요합니다.

2. X Window 보안

(1) 문제점

X 윈도우는 클라이언트/서버 모델에 입각한 윈도우 환경으로 사용자는 이를 사용함으로써 원격호스트와 로컬 디스플레이 사이에서 자유로운 통신을 주고받을 수 있는 반면, 네트워크를 통해서 어느 특정한 X 디스플레이의 스크린을 저장 한다거나 혹은 그곳 사용자의 키 입력을 가로채는 등의 행위가 가능하여 보안상의 문제를 야기할 수 있습니다. 실제로, 'xkey' 나 'xhack' 과 같은 유틸리티들은 공개적으로 널리 알려져있어 조금만 방심하면 해커들에게 중요한 정보들이 유출될 염려가 있습니다.

(2) 주요 공격 패턴

■ 디스플레이 상에 있는 특정 윈도우 파괴하기
■ 원격 스크린의 내용을 몰래 훔쳐보기
■ 의심스러운 X 이벤트 생성하기
■ 새로운 윈도우 생성하기
■ 사용자의 키입력 (패스워드 포함) 가로채기

(3) 해결책

X 윈도우를 보다 안전하게 하기 위하여 호스트 인증방식과 토큰 인증방식이 지원되고 있습니다.

1 호스트 인증방식 (Host Authentication)

호스트 인증방식을 위한 프로그램으로 xhost가 있는데, 이것은 단일사용자시스템에 적합한 방식으로 다중사용자 시스템에서는 보안상에 중대한 문제점이 있게 됩니다.
사용중인 X 서버를 다른 호스트에 열어둘 이유가 없을 때에는 아래와 같이 disable 해둡니다.

```
$ xhost -
```

위와같은 방법만으로도 해커들의 공격으로부터 어느정도 안전해 질수 있으며, 만일 꼭 필요한 호스트에 X 서버를 열어둘 필요가 있다면 아래와 같이 반드시 그 호스트에만 X 서버를 열어두도록 합니다.

```
$ xhost +www.anysite.com
```

www.anysite.com에 로긴한 후에는 DISPLAY라는 환경변수를 자신의 로컬 디스플레이로 다시 설정해 주어야 하는데, 이러한 행동은 그 당시 www.anysite.com에 접속하고 있는 모든 사용자에게 자신의 로컬 디스플레이를 열어주는 것으로 다른 사용자에게 악용될 소지가 있으며, 이것이 바로 다중사용자 시스템에서 호스트 인증방식이 부적합한 이유입니다.

② 토큰 인증방식 (Token Authentication)

다음에는 생성된 Magic cookie를 접근제어에 사용되는 .Xauthority라는 파일에 추가시켜야 합니다.

```
$ set cookie = `md5    /.Xauthority | awk '{print $4}'`
$ xauth add ${HOST}:0 . $cookie
$ xauth add ${HOST}/unix:0 . $cookie
$ xauth add localhost:0 . $cookie
```

위의 과정을 통해서 사용자의 홈디렉토리에 .Xauthority라는 파일이 생겼다면 제대로 입력되었는지 확인해 보도록 합니다.

```
$ xauth list
www.anysite.com/unix:0 MIT-MAGIC-COOKIE-1 38ad9729c
www.anysite.com:0 MIT-MAGIC-COOKIE-1 38ad9729c
localhost:0 MIT-MAGIC-COOKIE-1 38ad9729c
```

이 과정까지 끝났으면 X 서버에게 토큰 인증방식을 쓰겠다고 알립니다. 홈 디렉토리의 .xserverrc라는 파일을 다음과 같은 내용이 들어가도록 새로 만듭니다.

```
#!/bin/sh
exec /usr/X11R6/bin/X :0 -auth $HOME/.Xauthority
```

위에서 '-auth' 라는 옵션이 토큰 인증방식을 쓰겠다고 알리는 부분입니다. 여기까지 한 다음 보통 때와 마찬가지로 startx 명령으로X 윈도우를 띄우게 되면 호스트 단위의 접근 제어가 아닌 사용자 개개인마다의 접근 제어를 할 수 있는 상태가 되는 것입니다.
토큰 인증방식을 씀으로써 다른 원격 호스트에 접속했을 때 DISPLAY 환경변수를 매번 설정하고, 로컬에서는 xhost 프로그램을 실행시킬 필요가 없다는 편리함이 있습니다.
Magic cookie - 128bit로 이루어진 16진수 숫자. X 서버는 이것을 기억하고 있다가 어느 특정한 X 클라이언트가 자신과 접속하려고 할 때 그 접속을 허용할지의 여부를 판단하는 데 사용합니다.

3. 보안 도구

(1) Tripwire

① 개요

Tripwire는 파일 무결성 보관 도구로, 이전에 생성된 데이터베이스에 저장된 정보와 지정된 파일과 디렉토리를 비교하여 무결성을 검사합니다. Tripwire는 모든 변화를 감지하고, 목록의 추가와 삭제도 포함됩니다. 시스템 파일에 대하여 동작중일 경우 중요한 파일들의 어떠한 변화도 감지할 수 있으며, 그 즉시 적절한 손해 정도를 측정할 수 있습니다. Tripwire가 이상이 없다고 보고를 하면, 주어진 파일 목록 중에는 권한이 없는 사용자에게 수정된 파일이 없다는 것을 명백히 확신할 수 있습니다.

② 사용법

우선 tripwire를 설치를 해야 합니다. 설치 스크립트를 실행 시킵니다.

```
$ /etc/tripwire/twinstall.sh
---------------------------------------------------------------------------
The Tripwire site and local passphrases are used to sign a variety of files, such as the
configuration, policy, and database files.

Passphrases should be at least 8 characters in length and contain both letters and numbers.

See the Tripwire manual for more information.

---------------------------------------------------------------------------
Creating key files...

(When selecting a passphrase, keep in mind that good passphrases
typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)

Enter the site keyfile passphrase:
```

최소 8자 이상의 패스워드를 설정합다.

Verify the site keyfile passphrase:
최소 8자 이상의 패스워드를 반복 설정합니다.

Generating key (this may take several minutes)...Key generation complete.

(When selecting a passphrase, keep in mind that good passphrases
typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)

Enter the local keyfile passphrase:
최소 8자 이상의 패스워드를 설정합니다.

Verify the local keyfile passphrase:
최소 8자 이상의 패스워드를 반복 설정합니다.

Generating key (this may take several minutes)...

Signing configuration file...
Please enter your site passphrase:
처음 설정한 site 패스워드를 입력합니다.

Wrote configuration file: /etc/tripwire/tw.cfg

A clear-text version of the Tripwire configuration file
/etc/tripwire/twcfg.txt
has been preserved for your inspection. It is recommended that you
delete this file manually after you have examined it.

Signing policy file...
Please enter your site passphrase:
두번째 설정한 local 패스워드를 입력합니다.

Wrote policy file: /etc/tripwire/tw.pol

A clear-text version of the Tripwire policy file

/etc/tripwire/twpol.txt
has been preserved for your inspection. This implements
a minimal policy, intended only to test essential
Tripwire functionality. You should edit the policy file
to describe your system, and then use twadmin to generate
a new signed copy of the Tripwire policy.

이것으로 설치는 완료된 것입니다.

다음은 tripwire의 설정을 합니다.

$ mv twpol.txt twpol.txt.org 하여 원본을 백업합니다.
그런후에 twpol.txt파일을 다음과 같이 작성합니다.

```
#모든 파일은 read-only해야합니다.
/etc -> $(ReadOnly);
/bin -> $(ReadOnly);
(emailto="yoururl@site.net",severity=90);
```

작성이 끝났으면, 새설정파일을 만듭니다.

```
$ twadmin —create-polfile /etc/tripwire/twpol.txt
Please enter your site passphrase:
```

site 패스워드를 입력합니다.

```
Wrote policy file: /etc/tripwire/tw.pol
데이터베이스를 초기화합니다.
$ tripwire --init
Please enter your local passphrase:
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
Wrote database file: /var/lib/tripwire/neon2060.twd
The database was successfully generated.
```

이제 모든 설정이 완료 되었습니다. 이제 테스트를 해서 어떻게 사용하는지를 설명하겠습니다. 위에서

/etc 디렉토리만 CheckSum하기로 설정했으므로 확인해봅니다. 확인은 다음과 같은 명령으로 할 수 있습니다.

```
$ tripwire --check
Parsing policy file: /etc/tripwire/tw.pol
*** Processing Unix File System ***
Performing integrity check...
Wrote report file: /var/lib/tripwire/report/neon2060-20010617-
333519.twr

Tripwire(R) 2.3.0 Integrity Check Report

Report generated by: root
Report created on: 2001년 06월 17일 (일) 오후 11시 35분 19초
Database last updated on: Never

==========================================================
Report Summary:
==========================================================

Host name: neon2060
Host IP address: 127.0.0.1
Host ID: None
Policy file used: /etc/tripwire/tw.pol
Configuration file used: /etc/tripwire/tw.cfg
Database file used: /var/lib/tripwire/neon2060.twd
Command line used: tripwire -check

==========================================================
Rule Summary:
==========================================================

----------------------------------------------------------
Section: Unix File System
----------------------------------------------------------

  Rule Name       Severity Level       Added       Removed       Modified
--------------  ----------------  ----------  ------------  ------------
* etc                        90           0             0             1
 (/etc)

Total objects scanned: 1403
Total violations found: 1   ——→   총 1개가 변화가 있음.

==========================================================
Object Summary:
==========================================================

----------------------------------------------------------
# Section: Unix File System
----------------------------------------------------------

----------------------------------------------------------
Rule Name: etc (/etc)
Severity Level: 90
----------------------------------------------------------

Modified:   ——→   수정되었음
"/etc/tripwire"   ——→   /etc/tripwire 디렉토리가 수정됨

==========================================================
Error Report:
==========================================================

No Errors

---------------------------------------------------------------------------
*** End of report ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a
registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO
WARRANTY;
```

for details use —version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
Integrity check complete.

위에서 보듯이 Database를 초기화한후 처음 체크한것이므로 변화는 한곳밖에는 없는것을 알수 있습니다. 그 한곳은 tripwire의 설정 폴더가 /etc/tripwire 이기 때문입니다. 그럼 테스트로 /etc 에 neon2060.txt라는것을 생성한 후에 다시 테스트 해보겠습니다.

```
$ tripwire --check
Parsing policy file: /etc/tripwire/tw.pol
*** Processing Unix File System ***
Performing integrity check...
Wrote report file:/var/lib/tripwire/report/neon2060-20010617-333823.twr

Tripwire(R) 2.3.0 Integrity Check Report

Report generated by: root
Report created on: 2001년 06월 17일 (일) 오후 11시 38분 23초
Database last updated on: Never

==========================================================
Report Summary:
==========================================================

Host name: neon2060
Host IP address: 127.0.0.1
Host ID: None
Policy file used: /etc/tripwire/tw.pol
Configuration file used: /etc/tripwire/tw.cfg
Database file used: /var/lib/tripwire/neon2060.twd
Command line used: tripwire —check

==========================================================
Rule Summary:
==========================================================
```



```
----------------------------------------------------------
Section: Unix File System
----------------------------------------------------------

  Rule Name       Severity Level      Added      Removed      Modified
--------      ----------      -----      --------      --------
* etc                    90                1            0           2
(/etc)

Total objects scanned: 1404
Total violations found: 3   ——→  총 3개가 변화가 있음.

==========================================================
Object Summary:
==========================================================

----------------------------------------------------------
# Section: Unix File System
----------------------------------------------------------


----------------------------------------------------------
Rule Name: etc (/etc)
Severity Level: 90
----------------------------------------------------------

Added:   ——→  추가됨
"/etc/neon2060.txt"  ——→  /etc/neon2060.txt가 추가 되었음.

Modified:   ——→  수정됨
"/etc"
"/etc/tripwire"

==========================================================
Error Report:
==========================================================
```

No Errors

*** End of report ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use —version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
Integrity check complete.

위와 같이 해당 폴더가 변화가 있고, 파일이 추가 되었다는것을 알수 있습니다.이와 같이 파일들이 변조됨을 알수 있기 때문에 관리자로 하여금 중요 파일의 변조를 확인할수 있습니다. 만약 사용중간에 폴더 밑 파일을 추가나 변경을 하고 싶다면, /etc/tripwire/twpol.txt의 파일을 수정한 후에 밑에 과정을 거치면 됩니다.

```
$ twadmin --create-polfile /etc/tripwire/twpol.txt
...
$ tripwire --init
...
```

처음 것은 정책변경을 설정한것이고, 두번째 것은 비교 원본을 현재로 초기화 시킨것입니다. 위에서도 설명 했다시피 tripwire는 데이터베이스의 원본을 설정한 정책에 대해 현재와 비교를 하는것입니다. 참고로 정책을 설정할때, 너무 많은 폴더를 하게 되면 시간이 오래 걸립니다. (/ 폴더를 설정하게 되면 상당히 오랜동안 걸릴것이고, 자칫 다운도 될수 있습니다.) 그러니 관리자는 중요 파일들 위주로 설정을 하게 된다면 아주 유용하고 신뢰도 놓은 툴이 될수 있을것입니다. 또한 crond을 이용하여 체크를 수시로 해주게 되면 아주 유용할 것입니다.

(2) Snort

① Snort의 개요

침입차단시스템(Firewall)은 보안제품으로써 가장 널리 알려져 있으며, 국내에서도 많은 기관에서 침입차단시스템을 이용하여 보안을 강화하고 있습니다. 하지만 모든 보안제품을 100% 신뢰할 수 없는

것과 마찬가지로 침입차단시스템 만으로 완벽한 보안을 구축하였다고 장담할 수 없습니다.
보안은 방어(protect), 탐지(detect), 그리고 대응(react)으로 이루어진 defense-in-depth 접근법을 적용하여야만 한다고들 합니다. 침입차단시스템만으로는 완전한 방어가 이루어질 수는 없으므로 침입시도나 침입에 대한 신속한 탐지와 대응이 동시에 이루어져야 합니다. 이러한 공감대로 인해 최근 침입탐지시스템이 보안분야에서 이슈화되고 있습니다. 국내에서도 많은 보안업체에서 침입탐지시스템 개발에 노력하고 있습니다. 하지만 실제 국내 해킹 피해기관들은 대부분 중소규모의 업체들로써 고가의 상용 보안제품을 구매할 여력이 없는 경우가 많습니다.
침입탐지시스템은 국내외에 많은 상용제품이 나와 있지만, 공개 S/W 중에서도 활용 가능한 침입차단시스템들이 많습니다. 그 중에서 가장 대표적인 공개 침입탐지시스템이Snort일 것입니다. 국외에서도 Snort를 이용한 침입탐지 결과를 SANS 등에 알려 공격동향 정보를 공유하는 경우가 많다. Snort의 개발자인 Marty Roesch에 의하면 "Snort는 실시간 트래픽 분석과 IP 네트워크 상에서 패킷 로깅이 가능한 가벼운(lightweight) 네트워크 침입탐지시스템"이라고 합니다. Snort는 패킷 수집 라이브러리인 libpcap에 기반한 네트워크 스니퍼인데, 쉽게 정의할 수 있는 침입탐지 rule들에 일치되는 네트워크 트래픽을 감시하고 기록하고 경고할 수 있는 도구입니다. Snort는 프로토콜 분석, 내용 검색/매칭을 수행할 수 있으며 오버플로우, Stealth 포트스캔, CGI 공격, SMB 탐색, OS 확인 시도 등의 다양한 공격과 스캔을 탐지할 수 있습니다. 또한 이러한 탐지 rule들은 보안 Community를 통해 지속적으로 업데이트되고, 본인이 쉽게 rule을 작성하여 추가할 수 있으므로 최신 공격에 적응이 쉽습니다.

② Snort의 운영

Snort의 운영에서 실제 네트워크 관리자가 신경을 써야 할 부분은 자신의 네트워크 환경에 맞게 rule들을 Customizing하는 것입니다
snort-1.7에서는 기본적으로 탐지 rule이 630여개 정도 제공되고 있는데 다음 사이트에서 좀더 풍부한 rule을 다운로드받아 설치할 수 있습니다.

http://www.snort.org/Files/03152001/snortrules.tar.gz

이 파일을 snort-1.7이 설치된 디렉토리에 저장하고 압축을 풉니다. 압축파일에는 snort.conf 파일을 포함하여 약 840여개의 탐지 rule이 있습니다.
그러면 snort의 환경설정 파일인 snort.conf 파일에서 몇가지 변수들을 자신의 환경에 맞게 바꾸어 보자.
snort.conf 파일에서 여러 가지 네트워크 변수값들의 설정, preprocessor 환경, Output plug-in 환경, 사용될 rule의 설정 등을 할 수 있습니다.
필자는 다음과 같이 네트워크 변수값 등 기본적인 사항만 변경하였습니다.

```
var HOME_NET IPaddress
```

```
var EXTERNAL_NET IPaddress
#include rulesfile
```

위의 2개는 자신의 아이피주소를 적고 밑에 rulesfile은 만약 fulesfile을 추가하였다면 경로밑 화일명을 적어줍니다.
사용방법은 snort [-options] <filter options>

Options:
-A Set alert mode: fast, full, or none (alert file alerts only)
"unsock" enables UNIX socket logging (experimental).
-a Display ARP packets
-b Log packets in tcpdump format (much faster!)
-c <rules> Use Rules File <rules>
-C Print out payloads with character data only (no hex)
-d Dump the Application Layer
-D Run Snort in background (daemon) mode
-e Display the second layer header info
-F <bpf> Read BPF filters from file <bpf>
-g <gname> Run snort gid as <gname> group (or gid) after initialization
-h <hn> Home network = <hn>
-i <if> Listen on interface <if>
-I Add Interface name to alert output
-l <ld> Log to directory <ld>
-n <cnt> Exit after receiving <cnt> packets
-N Turn off logging (alerts still work)
-o Change the rule testing order to Pass|Alert|Log
-O Obfuscate the logged IP addresses
-p Disable promiscuous mode sniffing
-P <snap> set explicit snaplen of packet (default: 1514)
-q Quiet. Don't show banner and status report
-r <tf> Read and process tcpdump file <tf>
-s Log alert messages to syslog
-S <n=v> Set rules file variable n equal to value v
-t <dir> Chroots process to <dir> after initialization
-u <uname> Run snort uid as <uname> user (or uid) after initialization
-v Be verbose
-V Show version number
-X Dump the raw packet data starting at the link layer

다양한 설정을 할 수 있지만 여기서는 다음과 같이 설정하였습니다.

```
$ snort -d -l /var/log/snort -c /etc/snort/snort.conf -A full -D
```

이 결과 로그 디렉토리(/var/log/snort)에 alert, portscan.log, log 파일 등이 생성되는 것을 확인할 수 있습니다.
여기서 설치한 기본적인 옵션 이외에도 기관의 네트워크 환경에 맞게 다양하게 실행을 바꿀 수 있습니다.
너무 많은 옵션과 환경이 있을 수 있으니 일반적으로 많이 사용될 수 있는 환경과 그때의 옵션에 대해 알아보도록 하자.
먼저, snort에서 기본적으로 지정된 로그디렉토리 이외에도 syslog 데몬에게도 로그를 전달할 수 있습니다.
syslog에는 LOG_AUTHPRIV 장치와 LOG_ALERT 수준으로 전달되는데 syslog.conf 파일을 이용하여 관리자에게 경고 메일을 보내거나 다른 로그서버에 전달하는 것도 좋은 아이디어일 것입니다.
snort에서 syslog 데몬에 alert을 전달하기 위해서는 "-s"옵션을 사용합니다.
또, 대규모 bandwidth의 네트워크(예를들어 100Mbps)에서 트래픽량이 많을 경우 트래픽을 전부 처리 못하거나 로그로 인해 디스크가 full이 날 수도 있습니다.
이때 로그를 ASCII형태가 아닌 binary 형태로 저장(-b 옵션)하고 alert의 내용도 단순한 형태(-A fast)로 바꿀 수 있습니다.

```
$snort -b -A fast -c snort.conf
```

로그는 각 IP 별로 세부적인 패킷이 모두 저장되는데 이는 디스크 full의 원인이 되기 쉽습니다.
이때 "-N" 옵션을 사용하여 IP별 세부로그를 남기지 않을 수도 있습니다.
이외에도 다양한 환경에 적합하게 옵션들을 사용할 수 있는데 또 다른 환경들은 USAGE파일이나 help 화면을 참고하여 설정하기 바랍니다.

3 snort 탐지결과 분석

snort의 설치를 마치고 데몬을 실행하였다면 snort가 실제 공격을 탐지하지 못하는지(false-negative) 확인하고, 공격이 아닌 event에 대해서 공격이라고 경고하는지(false-positive) 확인하여 잘못된 rule을 수정하거나 제거하여야 합니다. 이부분을 넘어가거나 간과하여서는 snort를 100% 신뢰할수 없게 됩니다. 진정으로 snort의 힘을 100% 믿고 싶다면 반드시 꼼꼼하게 해주기 바랍니다.

공격 로그는 snort 실행시 로그 디렉토리로 지정한 /var/log/snort에 남게 됩니다. 로그 디렉토리에는 경고 메시지가 저장되는 alert 파일과 포트스캔 결과가 저장되는 portscan.log 파일 그리고, 각 IP 주소별로 좀더 상세한 로그를 저장합니다.
먼저 가장 일반적으로 발생되는 스캔 공격에 대해서 어떻게 탐지하는지 알아보도록 하자. 보안툴이기도 하고 포트 스캔 툴로 많이 이용되는 nmap을 이용하여 half-open stealth 스캔을 하여 보았습니다.

```
$ nmap -O -sS 172.16.2.34
```

이때 snort에서 탐지된 alert의 내용입니다.

```
[**] ICMP Nmap2.36BETA or HPING2 Echo [**]
03/27-01:57:51.301388 172.16.4.80 -> 172.16.2.34
ICMP TTL:40 TOS:0x0 ID:34166 IpLen:20 DgmLen:28
Type:8 Code:0 ID:56994 Seq:0 ECHO
[**] spp_portscan: PORTSCAN DETECTED from 172.16.4.80 (THRESHOLD 4
connections exceeded
in 0 seconds) [**]
03/27-01:57:51.644686
[**] SCAN Proxy attempt [**]
03/27-01:57:52.302744 172.16.4.80:38992 -> 172.16.2.34:1080
TCP TTL:58 TOS:0x0 ID:14930 IpLen:20 DgmLen:40
******S* Seq: 0x318068AA Ack: 0x0 Win: 0x1000 TcpLen: 20
[**] INFO - Possible Squid Scan [**]
03/27-01:57:52.522835 172.16.4.80:38992 -> 172.16.2.34:3128
TCP TTL:58 TOS:0x0 ID:45114 IpLen:20 DgmLen:40
******S* Seq: 0x318068AA Ack: 0x0 Win: 0x1000 TcpLen: 20
[**] SCAN nmap fingerprint attempt [**]
03/27-01:57:52.780479 172.16.4.80:39001 -> 172.16.2.34:21
TCP TTL:58 TOS:0x0 ID:44201 IpLen:20 DgmLen:60
**U*P*SF Seq: 0x580E33D1 Ack: 0x0 Win: 0x1000 TcpLen: 40 UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[**] SCAN nmap TCP [**]
03/27-01:57:52.780526 172.16.4.80:39002 -> 172.16.2.34:21
TCP TTL:58 TOS:0x0 ID:28513 IpLen:20 DgmLen:60
***A**** Seq: 0x580E33D1 Ack: 0x0 Win: 0x1000 TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[**] SCAN nmap TCP [**]
03/27-01:57:52.780621 172.16.4.80:39004 -> 172.16.2.34:1
TCP TTL:58 TOS:0x0 ID:19632 IpLen:20 DgmLen:60
***A**** Seq: 0x580E33D1 Ack: 0x0 Win: 0x1000 TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[**] spp_portscan: portscan status from 172.16.4.80: 1530 connections across
1 hosts: TCP(1529),
UDP(1) STEALTH [**]
03/27-01:58:00.073859
[**] spp_portscan: End of portscan from 172.16.4.80: TOTAL time(2s) hosts(1)
TCP(1529) UDP(1)
STEALTH [**]
03/27-01:58:04.551291
```

그리고 portscan.log 파일에 다음과 같은 로그가 남습니다.

```
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:7326 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:1417 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:481 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:163 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:1022 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:1993 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:336 SYN ******S*
Mar 27 01:57:51 172.16.4.80:38992 -> 172.16.2.34:718 SYN ******S*
...
```

FTP 데몬의 site exec 버그를 이용하여 원격에서 시스템 관리자 권한을 취득하려는 공격을 시도하였을 경우 snort에서 alert 파일에 남긴 공격 메시지는 다음과 같습니다.

```
[**] EXPLOIT x86 NOOP [**]
03/27-01:38:25.743974 172.16.4.80:2561 -> 172.16.2.34:21
TCP TTL:63 TOS:0x0 ID:8846 IpLen:20 DgmLen:558 DF
***AP*** Seq: 0x76E783D2 Ack: 0x499461D5 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 242635123 77742672
[**] FTP site exec [**]
03/27-01:38:27.773483 172.16.4.80:2561 -> 172.16.2.34:21
```

```
TCP TTL:63 TOS:0x0 ID:8850 IpLen:20 DgmLen:478 DF
***AP*** Seq: 0x76E785CC Ack: 0x49946486 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 242635327 77742675
```

이 결과 FTP 공격에 대해 정상적으로 탐지하고 있음을 알 수 있습니다.
그리고, 자세한 공격 메시지는 공격자 주소인 172.16.4.80이라는 디렉토리에 다음의 파일이 생성되어 있습니다.

```
# cat TCP:2561-21
[**] EXPLOIT x86 NOOP [**]
03/27-01:38:25.743974 172.16.4.80:2561 -> 172.16.2.34:21
TCP TTL:63 TOS:0x0 ID:8846 IpLen:20 DgmLen:558 DF
***AP*** Seq: 0x76E783D2 Ack: 0x499461D5 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 242635123 77742672
50 41 53 53 20 90 90 90 90 90 90 90 90 90 90 90 PASS ...........
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
90 90 90 90 90 90 90 90 90 90 90 90 31 C0 50 50 ............1.PP
50 B0 7E CD 80 31 DB 31 C0 43 43 53 4B 53 53 B0 P.   ..1.1.CCSKSS.
```



```
5A CD 80 EB 77 5E 31 C0 8D 5E 01 88 46 04 66 68 Z...w^1..^..F.fh
FF FF 01 53 53 B0 88 CD 80 31 C0 8D 5E 01 53 53 ...SS....1..^.SS
B0 3D CD 80 31 C0 31 DB 8D 5E 08 89 43 02 31 C9 .=..1.1..^..C.1.
FE C9 31 C0 8D 5E 08 53 53 B0 0C CD 80 FE C9 75 ..1..^.SS......u
F1 31 C0 88 46 09 8D 5E 08 53 53 B0 3D CD 80 FE .1..F..^.SS.=...
0E B0 30 FE C8 88 46 04 31 C0 88 46 07 89 76 08 ..0...F.1..F..v.
89 46 0C 89 F3 8D 4E 08 8D 56 0C 52 51 53 53 B0 .F....N..V.RQSS.
3B CD 80 31 C0 31 DB 53 53 B0 01 CD 80 E8 84 FF ;..1.1.SS.......
FF FF FF FF FF 30 62 69 6E 30 73 68 31 2E 2E 31 .....0bin0sh1..1
31 76 65 6E 67 6C 69 6E 0D 0A 1venglin..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] FTP site exec [**]
03/27-01:38:27.773483 172.16.4.80:2561 -> 172.16.2.34:21
TCP TTL:63 TOS:0x0 ID:8850 IpLen:20 DgmLen:478 DF
***AP*** Seq: 0x76E785CC Ack: 0x49946486 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 242635327 77742675
73 69 74 65 20 65 78 65 63 20 78 78 BC C6 BF BF site exec xx....
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
```

25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 64 25 63 %.f%.f%.f%.f%d%c
25 63 25 2E 66 7C 25 70 0D 0A %c%.f|%p..
=+

이 흔적에서 많은 수의 NOP(Null Operation)과 코드 중에 /bin/sh이 포함되어 있는 것을 볼 수 있는데 이로써 쉽게 FTP site exec 취약점을 이용한 버퍼오버플로우 공격 사실을 알 수 있습니다. 이제 공격이 아닌데 공격으로 탐지하는 rule들에 대한 정리를 할 필요가 있습니다.
snort 운영중에 실제 공격이 아닌 정상적인 웹서핑인데도 불구하고 IIS Unicode 버그를 공격하는 시도로 계속 탐지된 적이 있었습니다.

〔**〕 spp_http_decode: IIS Unicode attack detected 〔**〕
03/28-22:08:54.155195 172.16.2.149:1245 -> xxx.xxx.xxx.xxx:80
TCP TTL:128 TOS:0x0 ID:6395 IpLen:20 DgmLen:731 DF
AP Seq: 0xE2EB1A8C Ack: 0x9BEE182B Win: 0x43E4 TcpLen: 20

이는 http_decode preprocess에 의해 매칭되어 나타난 것입니다.
실제 Unicode 공격을 위해서는 "/"나 "\"의 Unicode인 %c1%1c, %c0%af, %c1%9c패턴을 매칭하여야 하지만 http_decode프로그램 코드(spp_http_decode.c)에서는 %c0, %c1, %e0, %f0, %f8, %fc만을 비교하는 것으로 보입니다.
이외에도 몇몇 rule들이 false-positive하게 탐지하는 경우들이 있었는데 이는 네트워크 관리자의 판단하에 해당 rule에 주석(#)처리하여 제거하거나, 탐지 패턴을 수정할 필요가 있습니다.
Snort를 더욱 빛내는 많은 보조도구 중에서 가장 많이 사용되는 것은 아마 SnortSnarf일 것입니다. Snort에 의해 생성된 로그화일을 웹페이지로 바꿔서 보여줍니다.
참고. http://www.silicondefense.com/software/snortsnarf/index.htm

(3) Nmap

① 개요

NMAP에서 가능한 스캔은 세가지로 나누어 볼 수 있습니다.

■ Ping으로 호스트가 존재하는지 조사합니다.
■ Port 스캔으로 호스트가 어떤 서비스를 제공하는지 검색합니다.
■ 호스트의 OS를 조사합니다.

② 사용법

nmap [Scan Type(s)] [Options] 〈host or net #1 ... [#N]〉

③ 타겟 선택

Nmap의 스캔 대상이 되는 호스트나 네트워크의 주소는 hostname/mask나 직접 IP주소를 이용할 수 있습니다.CERT의 C클래스 전부를 스캔하는 예. cert.org/24 192.88.209.5/24 192.88.209.0-255 128.88.209.*

④ 스캔 타입

■ sT tcp connect() port scan

가장 기본적인 port scanning 입니다. connect() 함수는 상대 host의 요청한 port들과 연결해 줍니다. 만약 port 가 열려있다면 connect() call은 성공할 것입니다. 이 방법의 좋은 점은 어떤 권한 없이 누구나 connect() 함수를 호출할 수 있으며, 따라서 어떠한 user라도 port scan을 할 수 있다는 것입니다. 또 하나의 장점은 속도가 빠르다는 것입니다. socket들을 병렬적으로 호출할 수 있으며 따라서 scan에 걸리는 시간을 단축할 수 있습니다. 그러나 단점을 들자면, 이 방법은 발견되기 쉽고, 또 걸러지기도 쉽습니다. 웬만한 logger들은 connect() 연결을 다 기록하며, scan 후에는 target의 log에 엄청나게 많은 연결 시도와, error message를 남길 것입니다.

■ sS tcp SYN stealth port scan (must be root)

이 방법은 흔히 "half-open" scanning 으로 불리는데, 왜냐하면 이 방법은 하나의 완전한 TCP connection을 열지 않기 때문입니다. target port에 SYN packet을 보내면, SYN|ACK나 RST가 응답으로 오게 되는데, SYN|ACK는 port가 열렸음을, RST는 port가 닫혔음을 나타냅니다. 만약 SYN|ACK가 도착하면 바로 RST를 보내서 연결을 중지 시킵니다. 이 방법은 Tcp-wrapper나 커널에서 지원하지 않는 이상 발견되지 않지만, ipfwadm과 firewall은 이 스캔을 감지합니다. 이러한 스캔을 위한 packet을 만들어 주기 위해서는 root 권한이 필요합니다.

■ sF,-sX,-sN Stealth FIN, Xmas, or Null scan (only works against UNIX)

세가지 방법 모두 Win95/98/NT 그리고 IRIX에서는 동작하지 않습니다. 위의 패킷을 받은 호스트는 닫힌 포트에 대해서는 RST로 응답하고, 열린 포트이면 패킷을 버립니다. 그러나 만약 패킷이 네트워

크상의 문제로 유실 되었을 때, NMAP은 패킷을 받을 수 없으 므로, 열린 포트로 오인 할 수 있습니다. 예전에는 탐지되지 않고 스캔을 시도 할 때 이 방법을 사용했으나, 현재는 발견될 수 있습니다.

■ sP ping "scan". Find which hosts on specified network(s) are up but don't일반적인 Ping을 이용해 스캔할 때 쓰는 옵션입니다.

■ sU UDP port scan (must be root)

이 scan 방법은 UDP를 쓴다는 점에서 위의 TCP scan방법과는 대조됩니다. 확실히 UDP protocol은 더 간단하지만 scan을 구현하는 것은 좀 더 어렵습니다. 왜냐하면 UDP 포트는 TCP 포트와는 다르게 open port일 때, ACK packet을 보낼 필요가 없으며, closed port는 error packet을 보낼 필요가 없습니다. 다행이 많은 host들은 closed UDP port로 packet을 보내면 ICMP_PORT_UNREACH error를 응답해 줍니다. 따라서 무엇이 닫혀있는 port인가를 알 수 있고 이를 제하면 열린 port라는 것을 알 수 있습니다. 그러나 UDP packet이나, ICMP error들이 꼭 도착한다고 보장을 하지 못하므로, UDP scanner들은 packet전달과정에서 누락된 packet의 재전송 부분을 구현해야 합니다. 또 이러한 방식에 scan에서 문제가 되는 부분은 속도입니다. 많은 기계들이 RFC1812 section 4.3.2.8을 참조하여, ICMP error message rate들을 제한하기 때문에 scan이 느리게 동작할 수밖에 없습니다.

■ b ftp "bounce attack" port scan

target을 scan하기 위해, ftp의 PORT 명령을 사용하는 방법입니다. 매개체가 될 호스트(주로 타겟 호스트가 신뢰하는 호스트입니다.)의 FTP로 접속해서 PORT 명령을 이용해 포트가 열려 있는 지를 확인합니다. 이 방법의 장점은 명확합니다. 이 방법을 쓰게 되었을 때, 이를 추적하기 어려우며, 또 firewall이나 packet filter등을 통과할 수도 있습니다. 단점이라면, 이 방법은 느리고 어떤 ftp에서는 아예 PORT기능을 disable시켰을 수도 있습니다.

■ f use tiny fragmented packets for SYN, FIN, Xmas, or NULL scan

SYN, FIN, Xmas, NULL스캔을 할 때 패킷을 쪼개서 보냅니다. 이 방법은 TCP header를 찢어서 보내므로, packet filter나 그외 여러 가지 등이 이 packet이 무었인지를 알기 어렵게 됩니다. 그러나, 어떠한 프로그램들은 이러한 작은 packet들을 잘 다루지 못함으로 에러가 발생할 수 있고, 요즘의 라우터들은 패킷의 fragment를 허용 하지 않습니다. 또, 이 방법은 IP fragment들을 queue하는 packet filter나 firewall에는 소용이 없습니다.

■ P0 Don't ping hosts (needed to scan www.microsoft.com and others)

호스트가 살아 있는지 Ping을 하지 않습니다. Ping을 하지 않음으로써 log파일에 기록될 가능성을 줄일 수 있고, filtering을 피할 수도 있습니다.

■ PT Use "TCP Ping" to see what hosts are up (for normal and ping scans)

일반적인 ICMP Ping은 firewall이나 packet filter에 의해 걸러 지므로 Ping을 할 때 TCP port 80에 ACK 패킷을 보내 Ping을 수행 하는 방법입니다. 만약 호스트가 살아 있다면 RST패킷을 보내올 것입니다.

■ PT21 Use "TCP Ping" scan with probe destination port of 21 (or whatever)

적어준 숫자의 포트에 ACK패킷을 보냅니다. 일반적인 호스트는 높은 숫자(예를 들어32523)의 포트는 필터링을 하지 않기 때문에 이를 이용할 수 있습니다. 그러나 어떤 호스트는 서비스 되고 있는 포트 외에는 막아놓기도 합니다.

■ PI Use ICMP ping packet to determines hosts that are up

일반적인 ICMP ping입니다. 이 방법은 firewall이나 packet filter에 의해 쉽게 걸러 집니다. 그러나 대부분의 시스템 관리자는 ICMP ping에 대한 로그를 신경 쓰지 않으므로, ICMP로 ping을 할 수 있는 호스트는 이 방법을 쓰는 것이 좋습니다.

■ PB Do BOTH TCP & ICMP scans in parallel (TCP dest port can be specified after the 'B')

Ping을 할 때 ICMP ping과 TCP ping을 동시에 사용합니다.

■ PS Use TCP SYN sweep rather than the default ACK sweep used in "TCP ping" TCP

ping을 할 때 ACK 패킷 대신 SYN 패킷을 보냅니다. 어떤 firewall은 스캔을 위해 보낸 ACK패킷이 현재 진행 중인 TCP connection 과정의 일부가 아님을 판단하고, 이 패킷을 차단 할 수 있습니다. 이런 firewall을 통과 하기 위해 SYN패킷을 이용합니다. 그러나, SYN을 이용하므로, 높은 포트 번호는 사용할 수 없을 것입니다.

■ O Use TCP/IP fingerprinting to guess what OS the remote host is running

OS를 판별 하기 위해서는 열린 포트와 닫힌 포트가 필요합니다. 닫힌 포트는 높은 포트 번호 중 랜덤하게 선택 됩니다. 그러나, 높은 포트를 filtering하는 호스트는 OS를 판별할 수 없고 유실된 packet이 생기면 잘못된 판별이 생길 수 있습니다.

■ p ports

특정 포트를 스캔하거나 스캔 할 포트의 범위를 지정해 줄 수 있습니다. '-p 23' 은 호스트의 23번 포트만 스캔을 하고, '-p 20-30,63000-' 은 20에서 30번 포트와 63000포트를 검색할 것입니다.

■ D decoy__host1,decoy2,ME,decoy3[,...]

스캔을 수행한 컴퓨터의 IP를 속이기 위해 임의의 다른 IP속에 자신의 IP를 숨겨 놓습니다. 스캔의 대상이된 호스트의 관리자는 decoy호스트에 나열된 모든 IP주소에서 스캔이 온것으로 오인합니다.

따라서 관리자는 모든 decoy이 호스트를 검색할 수 없으므로 추적을 포기하게 됩니다.

■ F fast scan. Only scans ports in /etc/services, a la strobe(1)
잘알려진 포트인 /etc/services에 있는 포트만을 검색하기 때문에 속도가 빠릅니다. -I Get identd (rfc 1413) info on listening TCP processes. TCP 프로세스의 identd 정보를 가져옵니다.

■ n Don't DNS resolve anything unless we have to (makes ping scans faster)숫자로된 IP 주소를 DNS 서버에 있는 호스트명으로 바꾸지 않기 때문에 속도가 빨라집니다.

■ R Try to resolve all hosts, even down ones (can take a lot of time)
모든 숫자 IP 주소를 호스트명으로 바꾼다. 속도가 상당히 느려집니다.

■ o Output scan logs to in human readable
스캔한 결과를 유저가 읽을 수 있는 텍스트 형식의 원하는 파일 이름으로 저장합니다.

■ m Output scan logs to <logfile> in machine parseable format
스캔한 결과를 다른 프로그램이나, 스크립트에서 읽기 쉬운 형식의 파일로 저장합니다.

■ i Grab IP numbers or hostnames from file. Use '-' for stdin
스캔의 대상이 되는 IP 주소나 호스트 네임을 파일에서 읽어 옵니다. 파일 이름대신 '-'를 적으면 표준 입력인 stdin에서 읽어 옵니다.

■ g Sets the source port used for scans. 20 and 53 are good choices
스캔할 포트번호를 적습니다.

■ S If you want to specify the source address of SYN or FYN scan
만약 스캔할 때 사용하는 패킷의 출발지 IP주소를 바꾸고 싶을 때, 이 옵션을 주고 원하는 IP 주소를 적습니다.

■ v Verbose. Its use is recommended. Use twice for greater effect
스캔할 때 옵션을 줄지를 하나씩 물어보는 질문형 모드로 전환합니다.

■ h help, print this junk. Also see http://www.insecure.org/nmap/
도움말을 본다. 또한 http://www.insecure.org/nmap/에서도 구할 수 있습니다.

■ V Print version number and exit
실행 중인 nmap의 버전을 표시하고 종료합니다.

■ e Send packets on interface (eth0,ppp0,etc.)
원하는 장치로 패킷을 보냅니다. 〈devicename〉에 원하는 장치명을 기입합니다.

■ q quash argv to something benign, currently set to "pine" (deprecated)
결과 값을 원하는 프로그램으로 보냅니다. 기본은 "pine"이지만 현재 사용이 안됩니다.

(4) Nessus

1 개요

Nessus프로젝트는 프리이며 아주 강력하고, 계속 진행중이며 쓰기 쉬운 보안 감시 도구를 개발 하는 프로젝트 입니다. security scanner는 원격 또는 내부네서 크래커들이 들어올 가능성이 있는 곳을 찾아주는 보안 도구입니다. Nessus는 매우 빠르고 신뢰할수 있습니다. 지금 당신의 네트워크의 안전을 테스트해보십시오.(절대 허가 없이 남의 사이트를 스캔 하지 마십시오. Nessus는 흔적을 많이 남기 때문에 후에 생기는 결과에 대해서는 절대 책임지지 않습니다.)

2 Nessus의 실행

nessus는 서버와 클라이언트로 구성되어 있습니다. 실행을 하기 위해선 로그인 과정이 필요하고 로그인을 하기 위해선 계정이 필요합니다. 계정을 추가 하기 위해서는 nessus-adduser라는 유틸리티를 이용합니다.

```
$nessus-adduser

Addition of a new nessusd user

Login:
여기에 계정을 쓰시고 엔터를 치세요.
Authentication method (cipher/plaintext) [cipher]:
인증방식을 선택하는 곳입니다. 기본적으로 cipher입니다. 그냥 엔터를 치시면 됩니다.

Source restriction
------------------------------------------------------------
```

You can, if you will, configure this account so that it can only
be used from a given host or subnet. For instance, you may want
neon to be able to connect to this nessusd server only from
his work machine.

Please enter the host (or subnet) neon is allowed to connect from.
A blank entry will allow him to connect from anywhere

The entry format must be an IP address followed by an optional netmask.
Hostnames are *not* accepted

Examples of valid entries :
  192.168.1.5
  192.168.1.0/24
  192.168.1.0/255.255.255.0

Invalid entry :
  prof.fr.nessus.org

Source host or network [anywhere] :

이것은 생성한 계정의 접속가능한 호스트를 지정해주는 곳입니다. IP주소나 넷마스크를 적어주시면 됩고, 공백으로 엔터를 치면 모든 곳에서의 연결을 허가하는 것입니다.

One time password :

패스워드를 넣는 곳입니다.

User rules
---

nessusd has a rules system which allows you to restrict the hosts
that neon has the right to test. For instance, you may want
him to be able to scan his own host only.

Please see the nessus-adduser(8) man page for the rules syntax

Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)

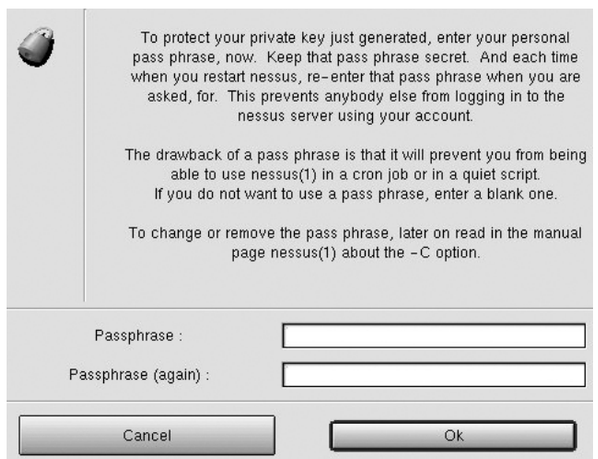
rule을 적는 곳입니다. nessus-adduser의 맨페이지를 확인하시면 됩니다. 공란으로 놔두고 컨트롤 D를 치면 모든곳에 스캔 할 수 있습니다. 우선은 공란으로 놔두시고 컨트롤 D를 누릅니다.

Login : 계정
Auth. method : cipher, can connect from anywhere
One time password : 패스워드
Rules :

Is that ok ? (y/n) [y]

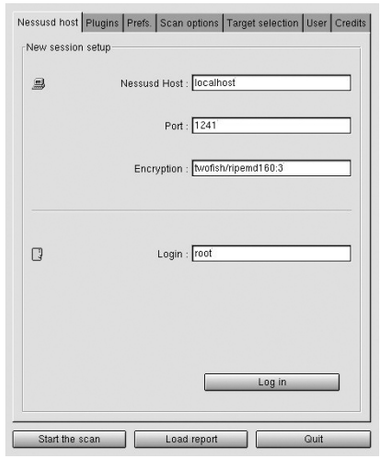
지금까지의 설정이 나옵니다. 올바르다면 y를 누르고 아니라면 n을 누르시고 다시 작업을 하시면 됩니다.
$ service nessusd start 또는 $ /etc/rc.d/init.d/nessusd start 를 하여 nessus 서버를 실행 합니다.
$ ps -ax|grep nessus를 하여 /usr/sbin/nessusd -D가 나오면 제대로 된것입니다.
이제 nessus를 실행 해보겠습니다. X-window 상의 터미널에서
$ nessus 를 쳐서 실행을 합니다. 처음에 실행을 하면 [그림 1]과 같은 창이 뜨면서 패스워드를 두번 입력하는 창이 뜰 것입니다.

거기서 설치시에 생성했던 패스워드를 입력합니다. 그리고 확인을 누릅니다.



[그림 VI-2]

그러면 그림 2와 같은 nessus setup창이 나옵니다.

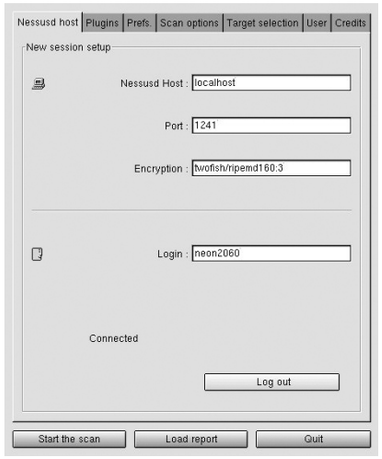


[그림 Ⅵ-3]

그림 2는 Nessusd host 셋업부분입니다. 로그인란에 전에 추가했던 계정을 넣습니다. 그런다음 Log in 버튼을 눌러서 로그인을 합니다.

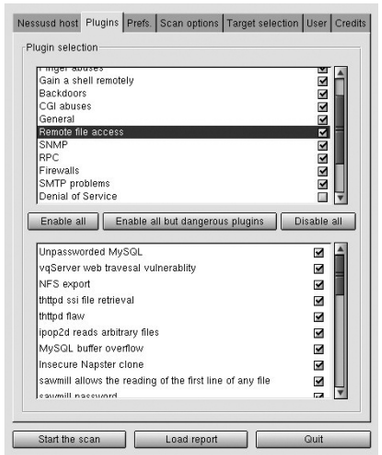
[그림 Ⅵ-4]

그러면 다시한번 패스워드를 물어보는 그림 3에서처럼 창이 뜹니다. 패스워드를 입력하시고 OK를 누릅니다.
그러면 그림 4처럼 변합니다. (참고로 필자는 계정을 neon2060으로 하였습니다.)



[그림 Ⅵ-5]

그다음 플러그 인(Plugins)를 선택합니다. 그림 5의 화면이 나옵니다.

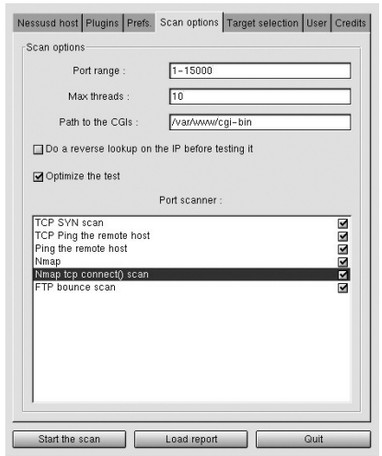


[그림 Ⅵ-6]

플러그인을 선택하는 곳입니다. Nessus가 알아낼수 있는 버그나 홀들의 목록이라고 생각하시면 됩니다.
여기서 어떤한 플러그인도 선택 하지 않는다면 Nessus는 아무것도 검색하지 않습니다.
검색하고 싶은 플러그인만을 선택합니다. 선택은 목록의 뒤에 체크 박스를 선택하면 됩니다. 목록을 선택하면 하위 목록이 밑에 리스트창에 나옵니다.
세부목록도 체크합니다. 디폴트는 전부 선택 되어 있습니다.
선택이 다 되었으면 Target selection항목을 선택 합니다. 그림 6과 같습니다.

[그림 Ⅵ-7]

이 항목에선 스캐닝할 호스트를 적습니다. (경고, 절대 허가되지 않은 호스트를 스캐닝 하지 마십시오. 플러그인중에는 DoS도 있어서 타겟호스트에 많은 로그들을 남기게 됩니다.)
호스트를 입력하였다면 Scan options 항목을 선택 합니다. 그림 7과 같습니다.



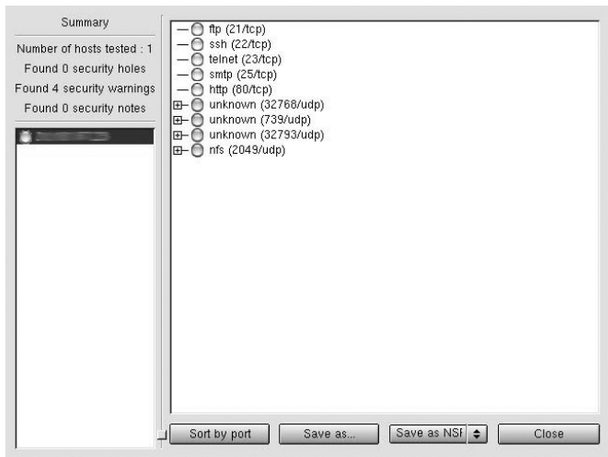
[그림 Ⅵ-8]

이 항목은 스캐닝 옵션을 정해주는 곳입니다. Port range는 스캐닝할 포트의 범위입니다. Max threads는 스캐닝을 할때 생성하는 쓰레드의 수 입니다.
시스템의 사양과도 상관이 되오니 너무 많은 쓰레드는 성능의 저하를 가져 옵니다. 기본적으로 10을 적습니다.

Path to the CGIs 에는 /var/www/cgi-bin 을 적습니다.
이제 설정은 어느정도 되었습니다. 그런 다음 창 하단에 Start the scan이라는 버튼을 눌러서 스캔을 시작 합니다. 그러면 그림 8과 같은 상태창이 나옵니다. prograss bar 가 진행여부를 알려줍니다. nmap과 같은 가벼운 툴보다는 다소 오래 걸립니다.

[그림 Ⅵ-9]

이제는 Nessus스캐닝 결과를 분석창을 보겠습니다. 그림 9는 필자가 허가된 호스트를 스캔해서 얻은 report입니다. 영어로 되어있지만 조금만 자세히 보면 금방 이해가 되실겁니다. 모든 스캐너들이 그렇겠지만, Nessus의 결과를 100% 신뢰할수는 없습니다. 새로운 버그나 홀들이 하루에두 수없이 쏟아져 나오기 때문에 보안에 항상 염두해두고 지켜봐야 합니다.



[그림 Ⅵ-10]

report 파일을 여러가지 포맷으로 저장 가능합니다.

(5) SSH(Secure Shell)

① 개요

SSH는 rlogin이나 rsh와 같은 BSD쪽의 service들이 보안에 아주 취약한 점을 대체하기 위하여 나온 원격 login 프로그램입니다. 일반 login 프로그램들이 packet을 전송할때 평문으로 전달을 하기 때문에 password를 쉽게 가로챌수 있는 것(예를 들어 sniffer를 들수가 있습니다)에 비해 ssh는 packet자체를 암호화를 하여 보내기 때문에 원격 관리에 혁명을 일으킨 프로그램입니다.

② SSH의 기본원리

보안쉘의 기본적인 개념은 데이타의 흐름에 있어서 직접적인 데이타의 송수신이 아니라 데이타를 한번 암호화 해서 송신하여 데이타의 유출이나 변조를 막아서 데이타 송수신시에 보안을 높이는 것입니다.

③ SSH의 방어효과

보안쉘이 주는 대부분의 방어효과는 기본원리인 데이타 암호화에서 생기는 것이며 여러가지가 있습니다.

■ IP 속이기(IP Spoofing)
IP를 속이는 것을 막습니다.

■ IP Source Routing
라우팅 경로가 임의로 지정되어서 보내어지는 데이타 그램을 무용지물로 만듭니다.

■ DNS 속이기(IP Spoofing)
DNS의 정보를 바꾸지 못하게 합니다.

■ 중간 호스트에서 데이타 가로채기
rlogin등을 사용할 경우 보내지는 패스워드가 직접 보내지지 않고 암호화를 하기 때문에 원거리 호스트와 서버 사이에서 데이타를 가로채더라도 암호키로 인해서 쓸모없는 데이타가 되게 됩니다.

■ 중간 호스트에서 데이타 임의 전송
중간 호스트에서 임의로 데이타를 만들어 내기 위해서는 암호키를 알아야 하지만암호키를 알아낼 수 없게 되어 있기 때문에 중간 호스트에서 데이타의 임의 전송이 불가능합니다.

■ X authentication interception, spoofed connection
X 윈도우 시스템에서 인증과정의 데이타 가로채기나 속여진 접속을 막도록 합니다.

④ 사용법

사용법은 간단하다. 일반 telnet 사용과 비슷합니다.

$ ssh -l (계정) (접속할서버주소)
(ex. ssh -l neon2060 www.serverurl.com)

이렇게 하면 비밀 번호를 물어보는데 비밀번호를 입력후에 엔터를 치면, 그이후에는 telnet 과 다를바가 없습니다.

4. 최소한의 보안

(1) 계정 보안

계정 보안은 사용자를 어떻게 관리 할 것인가와 관련이 있습니다. 일반적으로 계정 보안은 계정의 암호를 잘 관리하는 것을 말합니다. 그렇지만 요새는 패스워드를 root만이 볼수 있는 /etc/shadow에서 관리를 하기 때문에 암호화된 패스워드를 잃어버리는 경우는 흔치 않을 겁니다. 그렇기에 실질적으로 계정 보안은 단지 패스워드를 어렵게 하는것만으로는 되지 않습니다. 보다 폭넓게 계정사용자들의 습관들을 고쳐주는것이라 할수 있겠습니다. 암호화가 되지 않는 telnet 보다는 ssh의 사용을 권장하여 좀더 보안에 강한 프로그램을 쓰게끔 해야 할 것입니다.

(2) 시스템 내부 보안

시스템 내부 보안은 많은 부분이 파일 시스템과 관계되어 일어납니다. 특히 루트 소유의 SETUID실행 파일에서 보안 취약성이 존재하는 경우가 많습니다. 크래커들은 이런 SETUID가 걸린 파일의 오작동을 유도해서 원하는 작업을 수행해서 침입하는경우가 많습니다. 그러므로 관리자는 SETUID가 걸린 파일들을 사전에 조사하여, 치밀하게 감시해야 할것입니다. 되도록이면 불필요한 SETUID가 걸린 파일들을 제거 해야 할것입니다. 또한 크래커들은 일단 침투에 성공을 하면 루트킷을 설치해서 기존의 시스템 파일을 변조하는 경우가 많습니다. 그럴경우 크래커의 프로세스를 찾는 것도 힘들게 되고, 심하게는 침투가 되었는지도 모르게 됩니다. 그러므로 관리자는 중요한 시스템 파일들을 Tripwire같은 무결성 검사 도구로 수시로 체크를 해주어야 합니다. 밑에 코드는 필자가 임시로 만든 쉘스크립트입니다.

```
#!/bin/bash
# by Kim Jong-chul (neon2060)
# date : 2001, 06, 19
# user_name 과 email_id를 고쳐서 사용하시면 됩니다. email_id로 메일이 전달되니 필히 작성.
user_name="김종철"
email_id="yoururl@site.com"

Added_count= expr 0
Removed_count= expr 0
Modified_count= expr 0
IFS="
"
result=$(tripwire -check)
for val in $result; do"
        if [ "$val" = "Added:" ];then
                Added_ready="ok"
                Removed_ready=""
                Modified_ready=""
                continue
        fi
        if [ "$val" = "Removed:" ];then
                Added_ready=""
                Removed_ready="ok"
                Modified_ready=""
                continue
        fi
        if [ "$val" = "Modified:" ];then
                Added_ready=""
                Removed_ready=""
                Modified_ready="ok"
                continue
        fi
        if [ "$Added_ready" = "ok" ];then
                Removed_ready=""
                Modified_ready=""
                if [ "$val" != "==========================================
==========================================" ] && [ "$val" != "
Removed:" ] && [ "$val" != "Modified" ];then
                        if [ -n "$Added_data" ];then
                                Added_data=$Added_data,$val
                        else
                                Added_data=$val
                        fi
                        Added_count= expr $Added_count + 1
                        continue
                else
```

```
                Added_ready=""
                continue
        fi
    fi
    if [ "$Removed_ready" = "ok" ];then
        Added_ready=""
        Modified_ready=""
        if [ "$val" != "==========================================
==========================================" ] && [ "$val" != "
Modified:" ];then
            if [ -n "$Removed_data" ];then
                Removed_data=$Removed_data,$val
            else
                Removed_data=$val
            fi
            Removed_count= expr $Removed_count + 1
            continue
        else
            Removed_ready=""
            continue
        fi
    fi
    if [ "$Modified_ready" = "ok" ];then
        Added_ready=""
        Removed_ready=""
        if [ "$val" != "==========================================
==========================================" ];then
            if [ -n "$Modified_data" ];then
                Modified_data=$Modified_data,$val
            else
                Modified_data=$val
            fi
            Modified_count= expr $Modified_count + 1
            continue
        else
            Modified_ready=""
            continue
        fi
    fi
done
error_total_count= expr $Added_count + $Removed_count + $Modified_count
if [ $error_total_count -gt 0 ];then
    subject=$user_name"님 tripwire로 부터 메일이 왔습니다."
    content1="총$error_total_count개의 화일이 변경 및 위조 되었습니다."
    if [ $Added_count -gt 0 ];then
        content2=" $Added_count개의 화일 $Added_data가 추가 되었습니다."
    fi
    if [ $Removed_count -gt 0 ];then
        content3=" $Removed_count개의 화일 $Removed_data가 제거 되었습니다."
    fi
    if [ $Modified_count -gt 0 ];then
        content4=" $Modified_count개의 화일 $Modified_data가 변조 되었습니다."
    fi
    date= date
    mailto $email_id -s "$subject" << !CONTENT!
$content1
$content2
$content3
$content4
$date
!CONTENT!
fi
```

위의 코드를 /etc/cron.daily(하루에 한번) 나 /etc/cron.hourly(한시간에 한번) 에 실행권한(chmod 755)으로 넣습니다. 어떤 디렉토리에 넣을것인지는 관리자가 정한 정책에 따라 선택하면 될 것 같습니다. 그러면 하루나 한시간에 한번씩 파일에 대한 무결성 검사를 합니다(tripwire 가 인스톨 되었고, 설정또한 마무리가 졌다고 가정합니다. 만약 설치 밑 설정이 이루어지지 않았다면, 위의 보안 도구를 참고하시기 바랍니다.). 만약 파일이 변조되었다면 위의 스크립트에 있는 email로 중요 로그들이 정리되어 전해질 것입니다.

(3) 네트워크 보안

네트워크 보안 영역은 너무 크지만, 일반적으로 필요치 않은 데몬들을 서비스에서 내리는 것이 일차적인 작업일 것입니다. 필요는 하지만 가끔 사용하는 데몬(kudzu, sendmail 등..)들은 일단은 서비스에서 내리고, 필요시에 올려서 사용하면 될 것입니다. 또 불가피하게 서비스해야 하는 데몬들은 보안 패치가 있으면 즉시 패치를 해서 사용해야 할 것입니다. 또 Satan, Saint, Nessus등을 이용해서 자신의 서버의 취약점을 사전에 조사해서 패치할 것은 해준다면 더욱 보안에 강한 서버가 될것입니다.

(4) 기타 보안

지금까지의 보안 정책도 중요하지만 무엇보다 빼놓을 수 없는 것은 백업일 것입니다. 일반 자료뿐만 아니라, log파일들 까지 스크립트를 이용하여 주기적으로 백업을 해놓는다면, 불가피하게 피해를 입었을 때 그 피해는 경미해 질 수 있을 것입니다.