TCP Stream 분석을 통한 리눅스 기반의 웹 서버 IDS

정해진*, 문정훈*, 이명선*, 변옥환*

*한국과학기술정보연구원, 슈퍼컴퓨팅센터

Linux based IDS for Web Server through TCP Stream Analysis

Hae-jin Jung*, Jeong-hoon Moon*

Myung-sun Lee*, Ok-Hwan Byeon*

*Supercomputer Center, KISTI

요 약

NIDS 의 보급이 보편화됨에 따라 NUDS를 우회하기 위한 공격 기법 역시 많이 개발 되고 있다. 이런 공격들 중 일부는 NIDS 구조의 근본적인 결함을 이용하기 때문에 NIDS 구조에서는 해결될 수 없다. NIDS의 많은 장점들을 유지하면서도 NIDS의 한계를 극복하는 새로운 HIDS 모델을 제시한다. HIDS는 시스템에 많은 부하를 준다는 것이 가장 큰 문제점이지만, Web 서버는 특성상 모든 곳에서의 접속을 허용하므로 보안에 취약하기 때문에 어느 정도 HIDS에 의한 부하를 감수하더라도 보안을 강화해야만 한다. 또한, Web 서버는 Web 서비스라는 특정 목적만을 위해 운영되기 때문에 HIDS를 설치하더라도 Web 공격에 대해서만 고려함으로써 HIDS의 부하를 상당히 줄일 수 있다. 본 논문에서 제안하는 HIDS는 Linux 운영체제의 Kernel에서 TCP Stream을 추출하여 이를 감사 자료로써 사용하여 침입탐지를 한다.

I. 서론

20세기 말의 대표적인 패러다임으로 대두되는 인터넷은 정보 산업의 서비스를 드라이브하는 가장 큰 주체이다. 인터넷의 특성인 개방성과 표준성은 그 사용 주체의 속성에 상관없이 정보 교환과 정보 공유의 벽을 허물어 버렸다. 반면 정보 보호와 안전한 커뮤니케이션에 있어서는 인터넷의 특성상 신뢰감이 떨어지고 있다. 따라서 인터넷이라는 공간에서 이루어지는 각 서비스와 어플리케이션에 대해 적절한 정보 보호 체계가 구축되어야 한다.

정보 보호 기술에는 여러 가지 형태와 목적을 지니고 있다. 방화벽, 침입탐지, 취약성 분석, 접근 제어, 바이러스 백신에 이르기까지 많은 분야와 정보 보호만의 독특한 시장이 형성되어 있다. 현재 인터넷의 보안이 많은 정보 보호 기술 중에 IDS(Intrusion Detection System)에 많은 부분을 의존하고 있다. IDS는 침입탐지시스템으로 단순히 접근 제어 기능을 넘어서 침입의 패턴 데이터베이스와 Expert System을 사용해 네트워크나 시스템의 사용을 실시간 모니터닝하고 침입을 탐지하는 보안 시스템이다. IDS는 모니터링의 대상에 따라 네트워크 기반 IDS(NIDS)와 호스트 기반 IDS(HIDS)로 나눌 수 있다. 본 고에서는 이런 IDS들을 분석하고 인터넷의 보안에 있어 IDS의 한계를 알아보고 이를 대처할 수 있는 방안에 대해 제시하고자 한다.

따라서 본 논문의 동기에서는 NIDS의 한계를 분석하고 이를 극복할 수 있는 방안에 대해 제시 한다. 또한 Related Work에서는 본 논문에서 제안하는 HIDS와 기존의 연구되어 널리 쓰이고 있는 Packet Filtering과 Socket Filtering 과의 차이점을 알아보고 NIDS의 한계를 극복하고자 했던 다른 연구들에 대해서 알아본다. Proposed Solution에서는 NIDS의 한계를 극복하면서 Web 서버를 보안 할 수 있는 새로운 IDS 의 모델을 제시한다. 또 새로운 IDS(TIDS) 의 침입 탐지 기법 설계와 감사 자료 추출해 내는 부분을 제시하며 현재 구현 진행 중인 과정을 제시했다. 마지막 결론으로는 본 논문에서 제안한 IDS가 어떤 것들을 해결할 수 있으며 기존 IDS의 성능과 비교해 봤다.

II. Motivation

Web Server는 그 속성상 모든 곳에서 접속이 가능해야 하기 때문에 일반 서버들에서 쓰이는 보안 툴 예를 들면 TCP Wrapper나 Packet filtering 의 기능을 거의 사용할 수 없다. 이런 특성 때문에 특정 호스트로부터의 접근을 제어할 수 없고, 모든 서비스들을 제공되어야 하기 때문에 서비스 요청에 대한 패킷들을 일일이 거를 수도 없다. 따라서 다른 서버들에 비해 Web Server는 보안이 상당히 취약할 수 밖에 없다. 현재 인터넷과 eBusiness의 급성

장에 따라 Web Server에서 보안의 중요성은 더욱 커지고 있다.

현재는 Web 서버의 보안이 IDS에 많은 부분을 의존하고 있다고 앞에서도 말했다. IDS는 감시 대상에 따라 NIDS와 HIDS로 나뉜다고 했다. HIDS(Host-based IDS)는 호스트 기반 IDS로 시스템의 내부에 설치되어 하나의 시스템 내부 사용자들의 활동을 감시하고 해킹 시도를 탐지해 내는 시스템이다. NIDS(Network IDS)는 네트워크의 패킷 캡처링에 기반하여 네트워크를 지나다니는 패킷을 분석해서 침입을 탐지해낸다. HIDS는 모니터링하려는 시스템 마다 하나씩 설치가 되어야하지만, NIDS는 네트워크 단위로 하나만 설치하면 된다. 따라서 NIDS는 하나의 시스템으로 네트워크에 물려있는 모든 시스템을 보안할 수 있고 비교적 구현이 쉽다는 여러 장점을 가지고 있어 현재 상업용 제품이 가장 많이 나와있는 침입탐지 모델이다.

따라서 현재 웹 서버의 보안도 시스템에 부하를 많이 주는 HIDS보다는 전체적으로 하나의 시스템으로 많은 Host들을 보안할 수 있는 NIDS에 의존하고 있다.[7] 그러나 많은 NIDS가 개발되어 사용되고 있지만 NIDS의 근본적인 한계에서 기인하는 보안의 사각지대가 발생하고 있다. NIDS의 한계는 크게 다음과 같다.

a. 우회공격이 가능하다.

보호하고자 하는 서버와 NIDS의 TCP/IP Stack의 구현이 달라 똑 같은 패킷들을 받는다 할지라도 패킷들을 reassemble 하여 만든 Stream들은 달라질 수 있다. 이를 이용한 Insertion, Evasion Attack으로 인해 False Negative Alarm Rate와 False Positive Alarm Rate가 높고, 공격자의 의도에 의해 급격히 높아 질 수 있다.[3]

b. Fail Open하다.

즉, NIDS가 DOS 혹은 DDOS 공격에 의해 작동이 멈춘다 하더라도, 다른 서버와 네트워크는 정상적인 활동을 계속하기 때문에 NIDS의 자원을 고갈시킨 후에 다른 호스트를 해킹함으로써 NIDS에 탐지되지 않고 해킹하는 것이 가능하다.

c. 패킷 손실이 가능하다.

네트워크 장비의 발전으로 인해 네트워크의 속도가 빨라짐에 따라 NIDS의 패킷 손실률은 큰 문제로 떠오르고 있다. 실제로 한 두 패킷만 놓친다 하더라도 해킹을 전혀 탐지할 수 없기 때문에 패킷 손실은 심각한 문제일 수 밖에 없다.

이와 같은 단점으로 인해 False Negative 와 Positive Alarm이 많기 때문에 적극적인 Response를 하기 어렵다.[3]

따라서 정확한 Intrusion Detection을 하기 위해서는 해당 Protocol에 대한 정확한 정보가 필요하다. 하지만 대부분의 IDS들은 범용 IDS의 특성을 띄기 때문에 리소스의 한계로 인해 각 Protocol에 기반한 정밀한 검사가 불가능하며 각 Segment와 공격 Signature의 Pattern Matching 기법에만 의존하고 있다.

Ⅲ. Related Work

NIDS의 근본적인 한계에 기인하는 보안의 사각지대를 극복할 수 있는 본 고에서 제안될 새로운 IDS와 기존에 널리 사용되고 있는 Packet Filtering과 Socket Filtering을 분석하고 그들의 한계에 대해 알아보겠다. 또한 새로 제안될 IDS와 비슷하게 연구된 논문들을 참고했다.

a. Packet Filtering

패킷 필터는 지나가는 패킷의 헤더를 살펴보고 그 전체 패킷을 시스템으로 보낼 것인지 버릴 것인지를 결정하는 소프트웨어이다. 하지만 패킷의 헤더 부분만을 점검하기 때문에 TCP이상의 프로토콜에서 침입이 있을 경우에는 탐지 할 수 없다. [5]

b. Socket Filtering

Socket Filter는 사용자 프로그램의 소켓에 filter를 붙여 소켓을 통해 교환되는 데이터를 제한하기 위해 사용된다. 하지만, TCP 레벨을 지원하지 않으며, 사용자 프로그램에 직접 코드를 첨가하여야 하기 때문에 IDS와 같은 관리 프로그램에서 사용자 프로그램의 이런 기능을 사용할 방법이 없다.[5]

c. Transport and Application Protocol Scrubbing

TCP/IP Stack의 차이로 인해 NIDS와 End System이 똑 같은 패킷들을 다른 Stream으로 조합할 수 있는 문제를 해결하기 위한 논문이다. Firewall의 위치에 Scrubber를 두고, Scrubber 가 모든 시스템들이 같은 Stream으로 조합할 수 있는 형태로 바꾸어 패킷을 보내준다. 이 방식은 Scrubber가 Bottle Neck이 될 수 있기 때문에 Performance가 좋지 않고, Scrubber뒤의 모든 시스템들이 같은 Stream으로 조합하도록 할 수 있는 rule의 개발이 어렵고, 새로운 기법이나 시스템들의 TCP/IP Stack이 변하게 되면 rule 전체가 수정되어야 한다. [1]

d. Identification of Host Audit Data to Detect Attacks on Low-level IP Vulnerabilities

네트워크를 통해 교환되는 패킷들을 분석하여 Low-level 네트워크 공격이 일어나는지를 감시하는 HIDS에 관한 연구이다.[2]

Ⅳ. Proposed Solution

NIDS의 한계를 극복하면서 Web Server에 대해 일어나는 공격에 대해 최적의 침입 탐지를 할 수 있는 해결책은 Host-based Intrusion Detection System 이다. Host-based Intrusion Detection System(HIDS)는 일반적으로 보호하고자 하는 호스트 내부에서 감사 자료를 수집해서 침입을 탐지한다. 따라서 NIDS가 모니터링 하지 못하는 호스트에서 일어나는 모든 상황을 모니터링할 수 있다. 또한 Switched Network에서도 사용이 가능하고 호스트 내부에서 동작을 하기 때문에 네트워크가 암호화 되어있어도 영향을 받지 않는다.

일반적으로 기존 HIDS는 감사 자료를 내부에서 얻기 때문에 감사 자료로는 시스템이 남기는 로그들과 각 Process에서 일어나는 System Call 만 보고 침입을 탐지하고자 한다. 따라서 네트워크 상에서 Remote로 일어나는 공격에 대해서는 탐지하기가 쉽지않다.

본 고에서는 운영체제 Kernel에서 TCP Stream을 추출하여 침입이 일어나고 있는지를 판단하는 HIDS를 제시했다. 본 고에서는 운영체제로 개인 PC

와 공용 서버로 많이 쓰이는 리눅스를 사용하여 이론을 제시했다.

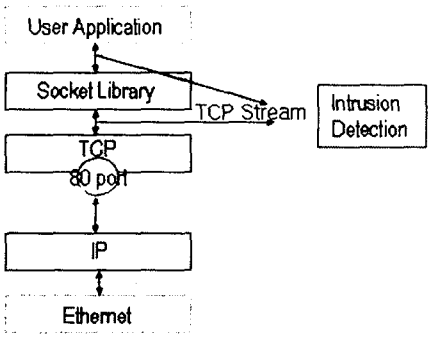
구현은 크게 세 가지 부분으로 나누어 진다. 첫째는 커널에서 Stream을 뽑아내는 부분이고, 둘째는 뽑아낸 Stream 을 이용하여 침입이 일어나는지 Intrusion Detection을 하는 부분이고, 셋째는 Intrusion Detection을 통해 침입을 발견했을 때, 이에 대한 Response를 책임지는 부분이 필요하다.

1. TCP/IP Stream 추출

리눅스 커널에서 TCP/IP의 Stream(Reassemble된 후를 말함)을 얻을 수 있는 곳은 INET Socket Layer와 BSD Socket Layer이다. 이 중에서 BSD Socket Layer를 수정하여 TCP/IP Straem을 얻으려고 한다. BSD Socket Layer를 수정하여 TCP/IP Stream을 얻을 때 가질 수 있는 장점은 다음과 같다.

a. BSD Socket Layer는 TCP위의 계층에서 Stream을 다루게 되므로, 다른 구조체로부터 Stream에 대한 정보(Source IP, Destination port 등)를 얻을 수 없기 때문에 프로세스의 정보를 얻는 것이 필수적이다. 사용자 프로세스와 직접 통신하므로, 사용자 프로세스의 정보를 얻을 수 있다.

b. BSD Socket Layer는 표준 인터페이스를 제공하기 위해 모든 BSD 계열의 운영 체제들이 지원하므로, 쉽게 다른 운영체제로 포팅이 가능하다.



[그림 1] 리눅스 커널 구조에서 TCP Stream 추출

커널에서 TCP/IP Stream을 뽑아내는 부분은 다음과 같은 방법으로 구현할 수 있다.

a. 커널 내부에서 printk를 호출하여 커널의 메시지를 user level program에 보낼 수도 있다. /var/log/message에 저장되는 내용들도 printk에 의해 출력되는 메시지들이다. 따라서 syslogd와 klogd의 옵션을 조정하여 출력되는 printk중 원하는 내용만 저장하여 이를 이용할 수 있다.

b. 시스템 호출은 user level program에게 커널에 접근할 수 있는 interface를 제공해준다. 따라서, 커널에 새로운 시스템 호출을 추가하여 user level program이 이를 호출하도록 해서, 커널로부터 TCP stream을 얻을 수 있다.

c. 리눅스에서는 커널이나 모듈(커널과 같은 level에서 돌아가지만, rebooting 없이 커널에 적재하거나 커널로부터 삭제가 가능함)이 user level program과 통신할 수 있는 수단이 proc filesystem이다. 일반 파일에 대한 억세스와 비슷한 방식이지만, 메모리에서 모든 연산이 이루어지므로 속도가 빠르다.

시스템 호출은 bug가 발생할 확률이 대단히 높고, 시스템 호출의 일부분만 수정해도 커널 전체를 다시 컴파일해야 하기 때문에 개발 효율도 떨어지는 방법이다. Proc filesystem을 이용하는 방법 역시 속도면에서는 좋은 편이나, 커널의 많은 부분을 직접 수정하여야 하기 때문에 개발이 용이하지만은 않다. 따라서, 이미 커널에서 상당 부분을 지원해 주고 있는 klogd를 이용하는 방법으로 구현했다.

2. TCP Stream analysis

여기까지가 제안되는 HIDS가 감사자료를 추출하는 부분이다. 이제는 제안된 HIDS가 추출된 감사자료를 어떻게 침입탐지를 하는지 논의하겠다.

일반적으로 Stream에서 침입 탐지를 위해 사용되는 기법은 Patten Search와 Protocol analysis 방법이다. Patten Search와 Protocol analysis는 다음과 같다.

a. Patten Search

Pattern Search기법은 단순히 독특한 Pattern을 Network Traffic에서 찾아서 기록하는 것 이다. 일종의 Virus scanning 기술을 Network traffic에 적용한 것이라고 생각하면 된다. Pattern searching system의 가장 좋은 예는 open-source로 개발되고 있는 Snort이다. Snort는 설치 시에 기본적으로 500개의 패턴 목록이 있다. 패킷들 중에서 패턴과 일치하는 것이 있으면 event file에 capture 되어 저장이 된다. 이러한 패턴들의 signature들을 rule이라고 한다.

[표 1] PHF 공격에 대한 Snort Rule

```
Alert tcp $EXTERNAL_NET any ->
$HTTP_SERVERS 80 (msg:"WEB-CGI phf
access";flags: A+; uricontent:"/phf"; nocase;
reference:bugtraq,629;
reference:arachnids,128;reference:cve,CVE-1999-00
67; classtype:attempted-recon; sid:886; rev:3
```

예를 들면, [표 1]의 PHF 공격에 대한 Snort Rule은 URL 의 "/phf"에 의해 만들어지는 HTTP 요구가 일어날 때마다 Snort를 행동하게 한다. 만약 hacker 가 PHF를 실행시키는 특별한 URL을 써서 시스템 안에 침입을 해온다면 패턴 서칭 시스템은 '/phf'대한 들어오는 모든 URL을 검사함으로써 침입자를 찾아낼 수 있다. 그러나 이 기법은 최근에 우회공격등에 의해 무력화될 수 있다. 또한 '/phf'패킷은 공격에 전혀 관련되지 않는 곳에서도 일어날 수 있다. 따라서 NIDS의 false positive 가 발생될 수 있다.[4]

a. Protocol analysis

pattern - search NIDS의 HTTP signature 보다 더 많은 정보와 header의 각각의 field를 나누어서 그 의미를 두어 분석한다. 따라서 evasion에 대해 좀 더 꼼꼼하게 분석할 수 있다.

[표 2] 전형적인 http 요청 예 [4]

```
GET / index.html HTTP/1.0
Host: www.robertgraham.com
Referer : http://www.robertgram.com/cgi-bin/phf
User-Agent : Mozilla/2.0
```

[표 3] Pattern Search 예[4]

GET / index.html HTTP/1.0Host: www.robertgraham.com Referer : http://www.robertgram.com/cgi-bin/phfUser-Agent : Mozilla/2.0

[표 4] Protocol Analysis 예 [4]

Method = GET URL = / index.html Version = HTTP/1.0 Fieldname = Host HTTP_Host: www.robertgraham.com Fieldname = Referer HTTP_Referer : http://www.robertgram.com/cgi-bin/phf Fieldname = User-Agent HTTP_Agent : Mozilla/2.0

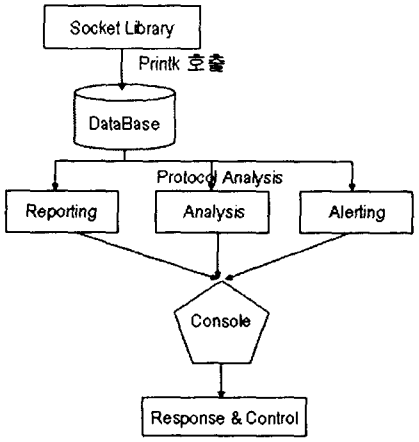
[표 2]과 같은 전형적인 HTTP 요청이 올 경우 Pattern Search 기법을 이용하는 IDS 시스템은 [표 3]와 같은 긴 문자열에서 공격 패턴과 일치하는 문자열이 존재하는지를 검색한다. 이는 실제 공격을 수행하기 위해 이용될 가능성이 전혀 없는 것들에 대해서까지 모두 검색하기 때문에 실제로는 공격이 일어나지 않는 부분인 HTTP_REFERER 부분에서 나오는 '/phf' 문자열로 인해 False Positive alarm를 일으키게 된다. 이에 반해 Protocol Analysis 기법은 [표 4]에서처럼 HTTP의 프로토콜에 따라 TCP Stream을 분석하고, 공격에 이용될 가능성이 전혀 없는 부분에 대해서는 공격 패턴이 존재하는지 찾아 보지 않고, 공격이 일어날 수 있는 부분(Method 와 URL 부분)만 검색한다. 따라서, 예제 패킷에 대해서False Positive Alarm을 내지 않고 정상적인 HTTP 요청으로 간주한다.

Protocol-analysis는 evasion 에 대한 대응 방법에 있어 pattern-search 기법 보다 우월하다고 볼 수 있다. 보통 padding에 의해 pattern이 smudged 되었을 때 protocol-analysis는 data에 얼마나 많이 padding 되었는지에 상관없이 자동적으로 data를 unsmudge하고 침입에 대해 올바르게 행동을 취한다. 이것은 대부분의 protocol 이 얼마간의 encoding이나 intrusion의 참된 signature 를 숨기는 *smudging*을 유사하게 인정하기 때문이다. 따라서 protocol-analysis NIDS는 이런 것들을 손 쉽게 다룰 수 있다.[4]

3. Response

본 논문에서는 BSD Socket layer로부터 추출된 Stream을 감사 자료로 이용하고 Protocol Analysis 기법으로 침입탐지를 하고 침입 탐지에서 침입으로 판단될 경우 소켓 파괴나 해당 Stream을 삭제하는 방식을 고려하고 있다. 차후 이에 대한 연구를 할 계획이다.

다음 [그림 2]는 제안된 솔루션 (명칭 TIDS)의 개념도이다.



[그림 2] TIDS 개념도

다음은 TIDS와 기존 IDS들과 의 차이점을 비교해보았다.

[표 5] TIDS와 기존 IDS들과의 차이

	NIDS	HIDS	TIDS
Audit Data	Network Packet	system Log	TCP Stream
Insertion, Evasion Attack	Possible	Impossible	Impossible
System Load	None	High	Medium
Packet Loss	Possible	Impossible	Impossible
Fail-open	Yes	No	No

제안된 솔루션 TIDS는 Host 기반 IDS이므로 HIDS들이 일반적으로 가지는 가장 큰 단점인 시스템의 로드를 줄여야 하는 문제점을 안고 있다. 그러나 TIDS는 논문 제목에서도 알 수 있듯이 웹 서버만을 위한 IDS이다. 따라서 TIDS의 룰셋과 침입 탐지는 웹 요청 트래픽과 플로우에 대해서만 탐지하기 때문에 시스템에 주는 로드가 적을 것이며 감사 자료가 시스템 로그가 아니기 때문에 자연적으로 로드는 줄어들 수밖에 없다.

V. Conclusion

새롭게 제안하는 Stream Analysis를 통한 Intrusion Detection은 다음과 같은 것들을 해결할 수 있다.

a. 위에서 언급한 NIDS의 한계를 극복할 수 있는 새로운 Intrusion Detection기법을 제시한다.

b. 기존의 범용 IDS의 한계를 극복할 수 있는 Web Server에 특화된 Intrusion Detection 기법을 제시한다.

c. False Positive와 False Negative를 현저히 줄임으로써 Active Response가 가능한 IDS를 제시한다.

기존의 범용 IDS는 원래의 취지가 Intrusion이 일어나는지 Detection을 하는 시스템이다. 그러나 침입에 대한 탐지를 이미 시스템이 해킹을 당하고 난 뒤에 침입을 탐지했다고 좋은 IDS는 아니다. 앞으로는 IDS가 침입에 대해 어느 정도의 예방률을 가지고 있고 또한 침입이 일어났을 때 얼마나 신속하게

침입을 오탐 없이 판단하고 침입에 어떻게 대응할 수 있는지가 중요하다.

따라서 앞으로는 IDS가 각종 공격에 대한 통계로 예방, 차후 대응 등을 할 수 있는 종합적인 보안툴이 되어야 한다고 본다. 이미 그러한 Management로써의 IDS가 개발되고 있는 실정이다.[7]

본 고에서는 새로운 HIDS를 개발하기위한 간단한 이론을제시했다. 앞으로는 좀더 많은 IDS를 분석하여 Active한 대응과 오탐이 없는 정확한 탐지 기법을 쓸 수 있는 HIDS로 완벽하게 구현하고자 한다.

참고문헌

[1] G. Robert Malan, David Watson and Farnam Jahanian "Transport and Application Protocol Scrubbing",IEEE INFOCOM 2000, March 26-31, Tel Aviv, Israel

[2] Thomas Daniels, Eugene Spafford "Identification of Host Audit Data to Detect Attacks on Low-level IP Vulnerabilities", July, 1998, Journal of Computer Security

[3] Thomas H. Ptacek , Timothy N. Newsh , "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", January, 1998

[4] Robert Graham, "NIDS-Pattern Serch vs. Protocol Decode", September 2001 Journal of Computer Science

[5] Julia Allen, Alan Christie, William Fithen, John McHugh,

Jed Pickel, Ed Ston, "State of the Practice of Intrusion Detection Technologies", January 200, Networked Systems Survivability Progr

[6] Steven J. Scott, "Threat Management Systems The State of Intrusion Detection" August 9, 2002

[7] Rebecca Bace, Peter Mell " NIST Special public On Intrusion Detection System"

[8] Marcus J. Ranum " Experiences Benchmarking Intrusion Detection Systems" NFR Security Technical Publications, Decenber, 2001