클러스터링 기법을 활용한 네트워크 비정상행위 탐지

오상현*, 이원석*

*연세대학교 컴퓨터과학과

Anomaly Intrusion Detection based on Clustering in Network Environment

Sang-hyun Oh*, Won-suk Lee*

Department of Computer Science Yonsei Univ.

요약

컴퓨터를 통한 침입을 탐지하기 위해서 많은 연구들이 오용탐지 기법을 개발하였다. 최근에는 오용 탐지 기법을 개선하기 위해서 비정상행위 탐지 기법에 관련된 연구들이 진행중이다. 본 논문에서는 클러스터링 기법을 응용한 새로운 네트워크 비정상행위 탐지 기법을 제안한다. 이를 위해서 정상 행위를 다양한 각도에서 분석될 수 있도록 네트워크 로그로부터 여러 특징들을 추출하고 각 특징에 대해서 클러스터링 알고리즘을 이용하여 정상 행위 패턴을 생성한다. 제안된 방법에서는 정상행위 패턴 즉 클러스터를 축약된 프로파일로 생성하는 방법을 제시하며 제안된 방법의 성능을 평가하기 위해서 DARPA에서 수집된 네트워크 로그를 이용하였다.

I. 서론

컴퓨터와 통신 기술의 발달로 사용자에게 다양한 정보와 편리성이 제공된 반면, 컴퓨터 침입 및 범죄로 인한 피해가 날로 증가하고 있다. 침입자들의 시스템 공격은 초기에는 침입 기법이 단순하였지만 정보 통신의 발전과 더불어 시스템 침입 기법도 고도화되고 전문적으로 변화해가고 있다. 따라서 이에 대응하는 침입 탐지 기법들도 그 복잡성을 더해 가고 있으므로 과거와 같이 각 침입 방식에 대한 개별적인 대처 방안으로는 충분한 보안 유지를 기대할 수 없다.

기존의 비정상행위 탐지 기법들 [1, 2]은 통계적인 기법을 이용하여 비정상행위 탐지를 수행하였다. 비정상행위 탐지 모델에서 주로 사용하는 통계적인 방법의 경우 실시간 판정에서 사용하게 되는 프로파일 데이타를 최소화 할 수 있다는 장점을 가지고 있는 반면, 감사 데이타를 통계적인 수치 값으로 표현함으로써 데이타의 손실이 발생될 수 있다. 따라서 비정상 행위가 부정확하게 판정될 가능성을 가지게 된다. 또한, 적은 빈도로 발생되는 작업에 대해서 효과적으로 관리하지 못하는 단점을 가지고 있다. 즉, 발생 빈도가 적은 행위가 주기적으로 발생되었다면 상당히 의미 있는 정보라 할 수 있다. 기존의 통계적 방법에서는 이러한 정보들을 희소 카테고리(Rare Category)로 관리하였다. 따라서 서로 관련 없는 행위들을 하나의 단위로 묶어서 관리하기 때문에 효과적인 판정이 이루어지지 않는다. 한편, 기존의 데이터마이닝을 이용한 침입 탐지 기법인 JAM [3]에서는 정상 행위의 frequent-episode [4]를 이용하여 침입을 탐지하였다. 하지만 수치 데이터에 대해서 정확하게 모델링할 수 없다는 단점을 가지고 있다.

본 논문에서는 비정상행위 탐지에서 방대한 데이타 분석을 좀 더 지능적이고 자동적으로 수행하기 위해 DBSCAN [5]와 같은 클러스터링 기법을 활용하였다. 이를 위해서 네트워크 로그로부터 추출된 특징들에 대해서 클러스터링을 이용하여 정상행위 패턴을 생성하며 이를 바탕으로 새로운 행동에 대한 비정상행위를 판별한다. 또한, 생성된 클러스터를 축약된 프로파일로 생성하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 네트워크 비정상행위 탐지 기법들을 소개하고 3장에서는 클러스터링 기법을 이용하여 정상행위 프로파일 생성 방법을 제안한다. 4장에서는 새롭게 수집된 데이터에 대해서 비정상행위 탐지를 수행하는 방법을 소개한다. 5장에서는 비정상행위 판정 시스템의 성능향상을 위한 모의실험 결과를 비교 분석하며 6장에서 최종적인 결론을 맺는다.

II. 관련 연구

NADIR [6]는 Los Alamos에서 개발한 규칙기반 전문가 시스템이다. 이 시스템은 네트워크상에서 침입 시도 및 대한 비정상행위를 탐지한다. 이 시스템에서는 클라이언트/서버 모델이 사용되며 각 서버로 Sybase가 동작하는 유닉스 기반 워크스테이션이 이용된다. 정상행위 프로파일과 사건 히스토리들은 각 감시 대상 시스템과 사용자별로 유지된다. NADIR에서 생성된 규칙들은 비정상행위를 탐지하는 프로파일에 적용된다.

EMERALD[2]는 네트웍에 독립적으로 분산되어있어 있는 모니터(distributed surveillance monitor)들을 사용하여 대규모 네트워크에서의 비정상 활동을 감지한다. EMERALD에서는 기존의 호스트기반 침입 기법인 NIDES [1]의 판정 메카니즘을 그대로 활용하였다. EMERALD는 분산 환경에서 실용적인 시스템 구현을 위해 기존의 침입탐지 시스템에서보다 작고 분산적이며 상호 협력적인 구성요소를 채택하는데, 이들 요소들은 계층적인 구조를 이루어 광범위한 네트워크를 감찰한다. 즉, 각 사용자의 활동 내용에 대한 분석과, 분석 결과가 부적합한 것으로 판명됐을 때의 대응이 병렬적으로 처리되어, 실시간 침입 탐지(real-time intrusion detection)에 보다 가까운 감찰을 할 수 있는 것이다.

AAFID[7] 시스템은 기존의 IDS에 대하여, 계층적이고 분산된 에이전트의 구조를 가짐으로써 하나의 에이전트가 서비스를 중지해도 다른 에이전트들이 수행을 계속할 수 있도록 하며 각 에이전트들이 독립적으로 수행되므로 전체의 시스템을 다시 시작해야 하는 번거로움을 해결한다. 또, 각 계층에 존재하는 에이전트들은 수집한 정보를 간단하게 정리하여 상위 계층으로 전달하므로 침입자가 잘못된 데이터 발생하려는 시도를 할 때 쉽게 감지될 수 있다.

JAM [3] 시스템에서는 여러 탐지 모델이 결합될수 있으며 각 탐지 모델에서는 분류화 모델에 의해서 생성된 기본 분류자를 이용하여 시스템의 일면을 감시한다. 각 탐지모델은 서로 다른 기술들이 채용될수 있고 서로간에 독립적이다. 따라서 침입은 기본 분류자들을 결합한 메타 분류자를 이용하여 탐지하게 된다. 한편, JAM 시스템에서는 frequent-episode 모델에 의해서 데이터가 수집되고 다양한 특징들이 추출된다. 탐지 결과를 향상시키기 위해서는 충분한 정상 및 비정상 감사 데이터가 수집되어야 한다.

III. 정상행위 패턴 생성

일반적으로 정상행위 패턴을 생성하기 위해서는 일정 기간 내에 생성된 네트워크 로그 단위로 분석될 수 있다. 이를 위해서 본 논문에서는 tcpdump [8]를 이용하여 패킷 기반 네트워크 로그를 분석하였다. 이와 같이 생성된 네트워크 패킷 기반 로그는 tcp-reduce [9]를 이용하여 다음과 같이 발생시간(Timestamp), 지속시간(Duration), 프로토콜(Protocol), 전송 바이트 수(Sent bytes), 수신 바이트 수(Received Bytes), 지역 호스트(Local host), 원격 호스트(Remote host) 및 상태(State)로 표시될 수 있다.

(Timestamp, Duration, Protocol, Sent bytes, Received bytes, Local host, Remote host, State)
$State = \{SF, REJ, S0, S1, S2, S3, S4, RSTOSn, RSTRSn, SS, SH, SHR, OOS1, OOS2\}$

예를 들어, 지역 호스트(local host) *www.yonsei.ac.kr*에 대해서 원격호스트 *165.132.121.31*로부터의 웹 세션이 정상적으로 이루어졌을 경우, *(896702851.979051, 0.092381, www, 257, 4027, www.yonsei.ac.kr, 165.132.121.31, SF)*와 같이 축약된 로그가 생성된다. 이때, JAM에서와 마찬가지로 시간 윈도우를 적용하여 시간 윈도우를 하나의 네트워크 행위로 설정한다. 본 논문에서 하나의 네트워크 행위로부터의 특징들은 시간 윈도우에 포함된 각 상태의 발생 빈도로 추출하였다.

클러스터링을 수행하기 위해 각 행위를 특징에 대한 수치 값으로 표현된다. 즉, i번째 행위를 o_i라 하면, k번째 특징에 대한 행위 값은 $v_k(o_i)$와 같이

표현될 수 있다. 또한, k번째 특징에 대한 행위 값들의 집합을 V_k와 같이 표현될 수 있다. 실질적으로 V_k를 이용하여 클러스터링이 수행된다. 본 논문에서는 DBSCAN 알고리즘[5]을 이용하여 행위 값들의 집합 V_k대해서 1차원 클러스터링을 수행하였다. 이때, 클러스터링에서 사용되는 파리미터는 클러스터링 범위 λ와 최소 밀도 δ이다. 클러스터링에서 클러스터링 범위는 유사한 행위 값들의 범위를 조절하고 최소 밀도는 발생빈도를 조절한다. 집합 V_k에 대한 클러스터링 수행 시 클러스터 분류를 위해서 특정 정상 행위 $v_k(o_i)$에 유사한 행위 집합 $N_\lambda(o_i)$는 다음과 같이 표현될 수 있다.

$$N_\lambda(v_k(o_i)) = \{x | v_k(a_i) - \lambda \le x \le v_k(a_i) + \lambda,\ x \in V_k\}$$

여기에서 V_k내에서 두 행위를 o_i와 o_j이라 했을 때, 두 행위가 동일한 클러스터에 포함되기 위한 조건은 다음과 같다.

① $N_\lambda(v_k(o_i)) \cap N_\lambda(v_k(o_j)) \neq \varnothing$.
② *for all* o *such that* $o_i \le o \le a_j$, $density(N_\lambda(v_k(o))) \ge \delta$

$density(N_\lambda(v_k(o)))$: $N_\lambda(v_k(o))$에 포함된 객체 개수

조건 ①은 클러스터링 범위 λ에 대해서 두 집합 $N_\lambda(v_k(o_i))$와 $N_\lambda(v_k(o_j))$에서 가장 인접한 두 행위 값간의 차가 λ이하임을 나타낸다. 즉, 인접한 두 행위 값이 클러스터링 범위 안에 존재하게 되므로 같은 클러스터로 묶일 수 있다. 조건 ②에서는 두 행위 값 o_i와 o_j 사이에 존재하는 모든 점에 대한 유사행위 집합의 밀도가 최소 밀도 δ이상인지의 여부를 판별한다.

그림 1은 클러스터를 생성하는 알고리즘을 나타낸다. 알고리즘 FNA의 상세한 수행과정은 다음과 같다.

[1단계] 클러스터링을 위해서 주어진 데이타를 오름 차순으로 정렬한다.

[2단계] 클러스터링은 정렬된 데이타의 가장작은 데이타부터 시작하며, 데이타의 유효 반경(λ) 안에 존재하는 데이타들의 집합 N_λ을 구한다.

[3단계] N_λ에 대한 밀도를 구하고 이를 최소 밀도와 비교한다. 이때 최소 밀도보다 작으면 이 데이타의 상태는 *noise*로 설정된다. 만일 최소 밀도 이상이면 이 데이타에 새로운 클러스터 식별자를 부여하고 N_λ에 포함되어 있는 데이타들을 스택(stack)에 넣고 스택이 빌 때까지 다음 단계를 수행한다. 만일 스택이 비어있으면 2단계를 다시 수행한다.

[4단계] 스택의 맨 상단으로부터 데이타를 읽어와서 이 데이타에 대한 N_λ을 구한다. 만일 N_λ의 밀도가 최소 밀도보다 크면 N_λ내에 존재하는 unclassified /noise 데이타에 현재의 클러스터 식별자를 부여한다.

[5단계] 스택을 초기화 한다. N_λ내에서 이전에 잡음(noise)이나 클러스터 식별자가 부여되지 않은 새로운 데이타들을 스택에 넣는다. 이 과정을 모든 데이타가 접근이 될 때까지 반복한다.

```
Find_Normal_Activity(Vk, δ, λ)
Vk:activity value set for kth feature which is
sorted by the ascending order
δ: minimum density
λ: clustering range

for all data v in Vk {
  if (v is unclassified) {
    retrieve Nλ(v);
    if (density(Nλ(v)) < δ) set noise to a and
       return;
    else {
     set new cluster-id to all data in Nλ(v);
     push all data from Nλ(v) onto stack;
     while(not stack.empty()) {
       current = stack.pop();
       retrieve Nλ(current);
       if (density(Nλ(current)) >= δ) {
         select all data unclassified/marked as
         noise in Nλ(current);
         stack.top = -1
         set current cluster-id to these data;
         push the unclassified data onto stack; }
       }
    }
}
```

그림 1: 클러스터링 알고리즘 Find Normal Activity(FNA)

이와 같이 클러스터링을 통해서 생성된 클러스터들은 비정상행위 탐지를 위해서 간략한 프로파일로 축약해야 된다. k번째 특징에서 i번째 클러스터를 C^i_k라 하자. 이때, 이 클러스터의 중심점 μ^i_k과 표준 편차 σ^i_k는 다음과 같이 구할 수 있다.

$$\mu^i_k = \frac{1}{|C^i_k|}\sum_{j=1}^{|C^i_k|} v_k(o_j)\ (o_j \in C^i_k)$$

$$\sigma^i_k = \sqrt{\frac{1}{|C^i_k|}\sum_{j=1}^{|C^i_k|} v_k(o_j)^2 - (\mu^i_k)^2}$$

결과적으로 클러스터 C^i_k의 프로파일 CP^i_k은 다음과 같이 구성된다.

$$CP^i_k = (\mu^i_k, \sigma^i_k)$$

따라서 클러스터의 개수가 m일때 k번째 특징에 대한 프로파일 집합은 $P_k=\{CP^1_k, CP^2_k, ..., CP^m_k\}$와 같이 같다.

IV. 비정상행위 탐지 기법

비정상 행위 판정을 위해서 정상행위 프로파일과 온라인에서 발생되는 행위에 대한 차이가 높을 경우 비정상 행위도가 높아지게 된다. 이때 어떤 특징에 대해서 클러스터가 두개 이상일 경우 행위값으로부터 중심점이 가장 가까운 클러스터가 선택된다. k번째 특징에 대해서 온라인에서의 행위 o의 차와 클러스터사이의 차는 다음과 같이 나타낼 수 있다.

$$diff(P_k, o) = \frac{|\mu^i_k - v_k(o)|}{\sigma^i_k}$$

where C^i_k is a closest cluster from $v_k(o)$.

이 수식에서는 서로 다른 특징들을 정규화 해주기 위해 클러스터의 표준편차로 나누어준다. 결과적으로 특징의 개수가 n일 때, 온라인 데이터 o에 대해서 전체 특징들에 대한 비정상 행위도(Abnormality)는 다음과 같이 계산될 수 있다.

$$abnormality(o) = \frac{1}{n} \cdot \sum_{k=1}^{n} diff(P_k, o)$$

온라인 행위의 비정상행위의 정도를 결정하기 위해서 본 논문에서는 서로 다른 비정상행위 레벨을 두며 이러한 레벨을 과거 네트워크 행위들에 대한 통계를 이용한다. 즉, 네트워크 행위에 대해서 두개의 레벨 (green 및 red)을 설정하여 네트워크 행위를 분류한다. green 레벨은 네트워크 행위가 정상적임을 나타내며 red 레벨은 네트워크 행위가 비정상적임을 나타낸다. 이를 위하여, 과거 정상적인 행위들의 집합을 N이라 할 때, 평균 비정상행위도 Φ와 이에 대한 표준편차 sd로 다음과 같이 나타낼 수 있다.

$$\Phi = \frac{1}{|N|} \sum_{j=1}^{|N|} abnormality(o_j) \; (o_j \in N)$$

$$sd = \sqrt{\frac{1}{|N|} \sum_{j=1}^{|N|} \frac{1}{2} abnormality(o_j) \frac{3}{4}^2 - (\Phi)^2 (o_j \in N)}$$

온라인 행위 o에 대해서, 각 레벨의 범위는 다음과 같이 나타낼 수 있다. 여기에서 ξ는 정상행위의 범위를 조절하는 파라미터이다. 이때, *false alarm rate* 는 전체 정상행위의 개수에 대해서 *red* 레벨에 포함되는 행위의 비율을 나타내고 *detection rate*는 전체 비정상행위의 개수에 대해서 *red* 레벨에 포함되는 행위의 비율을 나타낸다.

- *green: if $0 \le abnormality(o) \le \Phi + \xi \cdot sd$*
- *red: if $\Phi + \xi \cdot sd < abnormality(o)$.*

V. 실험 및 결과 분석

본 장에서는 판정시스템의 성능에 대한 검증을 위하여 수행한 모의 실험 환경 및 모의 실험 결과에 관하여 기술한다. 모의 실험을 위해서 1998년에 DARPA에서 수집된 tcpdump 네트워크 로그 데이터 [10]를 이용하였다.

본 논문에서 제안된 방법은 클러스터링을 이용하여 네트워크 정상행위 패턴을 생성하기 때문에 최소 밀도와 클러스터링의 범위는 정상행위 패턴 생성시 상당히 많은 영향을 준다. 만일 최소 밀도가 너무 높게 설정되거나 클러스터링 범위가 너무 좁게 설정되면 false alarm rate가 높게 나타나게 된다. 반면 최소 밀도가 너무 낮게 설정되거나 클러스터링 범위를 너무 크게 설정되면 전체 데이터가 하나의 클러스터로 생성될 수 있기 때문에 공격을 효과적으로 탐지할 수 없다. 따라서 최적의 클러스터링 범위를 설정하는 것은 생성될 클러스터의 개수와 클러스터의 정확도에 상당히 많은 영향을 미치게 된다. 본 논문에서는 클러스터링 범위를 1로 설정하고 최소 밀도의 변화에 따라서 비정상행위 판정에 대한 실험을 수행하였다. 한편, 네트워크 행위의 단위인 시간 윈도우는 10초로 설정하였다.

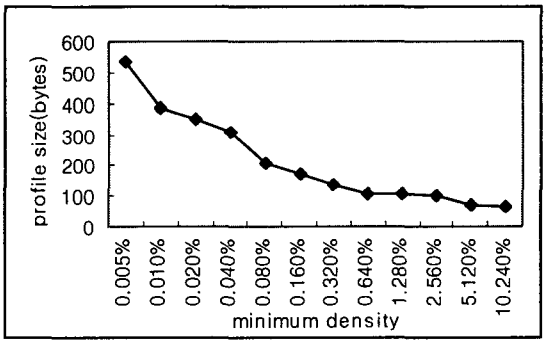

그림 2: 최소 밀도에 따른 프로파일 사이즈

그림 2는 최소 밀도에 따른 프로파일 크기를 나타낸다. 이 그림에서 최소 밀도가 커질수록 프로파일의 크기가 작아짐을 볼 수 있다.

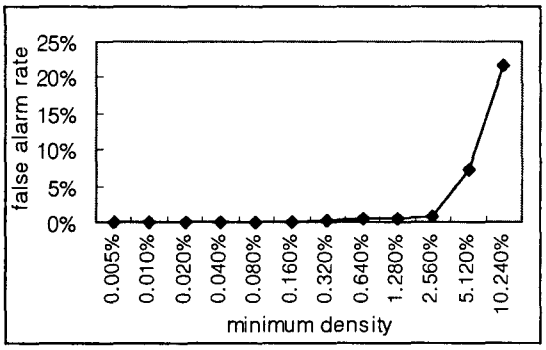

그림 3: 최소 밀도에 따른 *false alarm rate* (ξ=2)

그림 3은 최소 밀도에 따른 *false alarm rate*를 나타낸다. 이 그림에서 최소 밀도가 2.56%이하일 때 false alarm rate가 5% 미만으로 나타났다.

한편, 그림 4에서는 최소 밀도에 따른 *detection rate*를 보여준다. 이 그림에서 최소 밀도가 0.16% 이상일 때 모든 공격을 탐지함을 볼 수 있다. 그림 2에서 그림 4까지의 실험결과로부터 최소 밀도가 너무 작게 설정되면 프로파일의 크기가 커짐으로써 정상행위의 범위가 넓어지게 된다. 따라서 *false alarm rate*가 작아지게 되는 반면 효과적으로 공격을 탐지할 수 없게 된다. 반대로 최소 밀도가 너무 높게 설정되면 *false alarm rate*가 커지게 되어서 신뢰할만한 비정상행위 탐지가 어렵게 된다. 그림 3과 그림 4에 의해서 가장 효과적은 최소 밀도의 범위는 0.160%~2.560%까지로 설정될 수 있다.

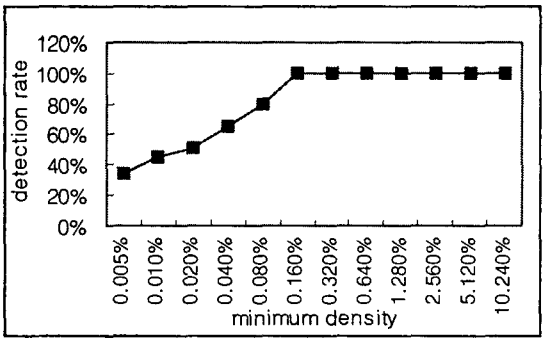

그림 4: 최소 밀도에 따른 *detection rate*(ξ=2)

VI. 결론 및 향후 연구방향

기존의 많은 연구들이 통계적인 기법을 이용하여 호스트 기반 침입을 탐지를 수행하였다. 하지만 통계적인 기법은 사용자 행위에 대한 평균치를 이용하여 사용자의 비정상 행위가 상당히 부정확하게 판정될 가능성을 가지게 된다. 이를 해결하기 본 논문에서는 방대한 데이타 분석을 좀 더 지능적이고 자동적으로 수행하기 위한 데이타마이닝 기법 중에서 클러스터링 기법을 활용하여 네트워크 로그로부터 정상행위 패턴을 분석하였다. 이를 위해서 비정상행위를 다양한 각도에서 분석될 수 있도록 네트워크 로그로부터 여러 특징들을 추출하고 각 특징에 대해서 클러스터링을 수행하여 네트워크 정상행위 패턴이 생성될 수 있다. 또한 생성된 정상행위 패턴을 간략한 프로파일로 구성함으로써 실시간에 보다 효율적인 비정상행위 판정을 기대할 수 있다. 한편, 본 논문에서 제안한 방법의 성능을 평가하기 위해서 DARPA에서 수집된 네트워크 로그를 이용하였으며 네트워크로부터 수행된 공격들에 대해서 효과적으로 탐지를 하였다. 향후 연구로는 클러스터링 수행시 최적의 클러스터링 범위 및 최소 밀도를 자동으로 설정하는 방법을 연구할 예정이다.

참고문헌

[1] Harold S.Javitz and Alfonso Valdes, "The NIDES Statistical Component Description and Justification," Annual report, SRI International, 333 Ravenwood Avenue, Menlo Park, CA 94025, March 1994.

[2] Phillip A. Porras and Peter G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," 20th NISSC, October 1997.

[3] S.J. Stolfo, A.L. Prodromidis, S. Tselepis, W. Lee, D. Fan, P.K. Chan, "JAM:Java agents for Meta-Learning over Distributed Databases," .Proc. KDD-97 and AAAI97 Work. on AI Methods in Fraud and Risk Management), 1997.

[4] H. Mannila, H. Toivonen and I. Verkamo, "Discovery of frequent episodes in event sequences," Data Mining and Knowledge Discovery, Vol. 1, No. 3, 1997, 259-289.

[5] Martin Ester, Hans-Peter Kriegel, Sander, Michael Wimmer, Xiaowei Xu, "Incremental Clustering for Mining in a Data Warehousing Environment", Proceedings of the 24th VLDB Conference, New York, USA, 1998.

[6] Hochberg, J., Jackson, K., Stallings, C.,

McClary, J., DuBois, D., Ford, J. NADIR: An automated system for detecting network intrusions and misuse. Computers and Security 12(1993)3, May, pages 253-248.

[7] Jai Sundar Balesubramaniyan, Jose Omar Garcia-Fernandes, David Isacoff, Engene Spafford, Diego Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents," Technical Report 98-05, COAST Laboratory, Purdue University, West Lafayette, IN 47907-1398, May 1998.

[8] http://www.tcpdump.org/

[9] http://www.cs.unc.edu/Research/dirt/

[10] http://www.ll.mit.edu/IST/ideval/index.html