Journal of Next-genescoren Convergence Information Services Technology Vol.8, No.2, June (2019), pp.187-202
차세대컨버전스정보서비스기술논문지 *http://dx.doi.org/10.29056/jncist.2019.06.04*

네트워크침입탐지에의 랜덤포레스트알고리즘 적용

An Application of Random Forest Algorithm to Network Intrusion Detection

알리타 알폰소나 럭바훈가[1], 심동희[2]*

Alita Alphonsovna Rukubayihunga[1], Shim Donghee[2]*

요 약

랜덤포레스트는 분류나 회귀분석 등에 사용되어왔던 알고리즘이다. 최근 보안이 강조되면서 네트워크보안을 강화하기 위하여 침입탐지시스템이 개발되어 사용되어 왔다. 침입탐지를 위하여 많은 기법들을 개발하여 이용하고 있다. 본 연구에서는 랜덤포리스트알고리즘을 KDD Cup 1999를 이용하여 네트워크침입탐지에 적용하여 다른 분류에 사용되는 알고리즘과 성능을 비교하였다. 그 결과 Naive Bayes, J48 등의 방법들보다 우수한 성능을 나타내었다.

핵심어 : 랜덤포레스트, 침입탐지시스템, 네트워크보안, KDD Cup 1999, J48, 네이브 베이즈

Abstract

Random forest algorithm has been used in classification and regression analysis. Intrusion detection system has been developed for intensifying the network security as the security become more serious. Many techniques have been developed and used for this intrusion detection. In this paper random forest algorithm is applied to this intrusion detection for the KDD Cup 1999 and its performance is compared with other classification algorithms. This random forest algorithm show more efficient compared with other techniques such as Naive Bayes, J48 in performance evaluation.

Keyword : Random Forest, Intrusion Detection System, Network Security, KDD Cup 1999, J48, Naive Bayes

1 Department of Computer Science & Engineering, Jeonju University, Jeonju, Korea [Graduate Student]
e-mail: alita@jj.ac.kr
2 Department of Computer Science & Engineering, Jeonju University , Jeonju, Korea [Professor]
e-mail: dh66shim@naver.com (Corresponding author)
* 이 논문은 Alita Alphonsovna Rukubayihunga 석사학위논문을 기반으로 작성되었습니다.
Received(March 29, 2019), Review Result(1st: April 12, 2019), Accepted(June 03, 2019), Published(June 30, 2019)

ISSN: 2384-101X JNCIST

1. 서론

인터넷의 확산과 더불어 보안이 점차 강조되면서 침입탐지시스템(IDS:Intrusion Detection System)은 방화벽(Firewall), 가상사설망(VPN: Virtual Private Network)과 더불어 네트워크보안의 3대 도구로 널리 활용되고 있다[1]. 그리하여 대부분의 기업에서 네트워크보안을 강화하기 위하여 3대 보안도구를 설치운영하고 있다.

한편 랜덤포레스트(Random Forest) 알고리즘은 기계학습 분야에 속하는 것으로서 분류, 회귀 분석 등에 사용되는 앙상블 학습 방법의 일종으로, 훈련 과정에서 구성한 다수의 결정 트리로부터 분류 또는 평균 예측치를 출력함으로써 동작한다[2].

본 논문에서는 랜덤포레스트 알고리즘을 침입탐지에 적용하였다. 그리하여 KDD Cup 1999 데이터 세트[3]를 이용하여 성능평가를 하였다.

본 논문의 구성은 2장에서 침입탐지시스템의 유형과 특징을 요약하여 설명하였으며, 또한 랜덤포레스트 알고리즘과 Naive Bayes[4], J48[5][6] 등을 설명하였다.

3장에서는 침입탐지에 랜덤포레스트알고리즘을 적용하기 위한 의사코드를 제시하였으며, 4장에서는 KDD Cup 1999 데이터세트를 이용한 성능평가를 하였으며 5장에서는 결론을 서술하였다.

2. 침입탐지시스템과 기계학습알고리즘

2.1 침입탐지시스템

침입탐지시스템(IDS: Intrusion Detection System)은 네트워크나 호스트에 대한 원하지 않는 처리(Processing)나 조작(Manipulation)을 발견해 준다[1]. 이러한 IDS는 수 많은 종류들이 존재하며, 이들은 몇 가지 기준에서 분류될 수 있다. 원하지 않는 처리나 조작은 악의를 가진 해커 또는 자동화된 툴을 사용하는 스크립트에 의한 공격으로 이루어질 수 있다. 전통적으로 사용되던 방화벽(Firewall)이 차단하지 못한 악의적인 패킷을 탐지하는 것이 IDS 의 목적이다. 이러한 패킷은 취약한 서비스에 대한 네트워크 공격과 응용프로그램의 데이터 처리 공격, 권한 상승, 불법 로그인, 중요 파일에 대한 불법적인 접근, 악성 소프트웨어(컴퓨터 바이러스, 트로이 목마, 웜)를 이용한 공격을 포함한다[7].

IDS는 보안이벤트를 발생시키는 센서, 이벤트를 모니터링하고 센서를 제어하는 콘솔, 이벤트를 데이터베이스에 기록하고 규칙을 이용하여 경고를 생성하는 엔진으로 구성된다. 이러한 IDS를 분류하는 방법은 센서의 종류와 위치 그리고 엔진이 경고를 만드는 데 사용하는 방법론 등에 따라

여러가지가 있다. 많은 IDS들은 위의 세 가지 요소들을 하나의 장치 또는 설비로 구현하고 있다.

(1) 신호기반 IDS와 이상기반 IDS

신호기반(Signituare-Based) IDS로도 알려진 악용 탐지시스템(Misuse Detection System)은 악의적인 것으로 추정되는 트래픽 또는 응용데이터의 패턴을 감시하여 침입을 인식한다. 이러한 시스템은 알려진 공격만 탐지할 수 있다고 볼 수 있다. IDS는 수집된 정보를 분석하여, 공격신호를 저장하는 데이터베이스를 통해 그것을 비교한다. IDS는 본질적으로 이미 알려진, 문서화된 특정 공격을 찾는 것이다. 바이러스 탐지 시스템에서처럼, 악용 탐지 소프트웨어는 단지 패킷을 비교하기 위해 사용되는 공격 신호 데이터베이스와 유사한 것이다.

한편 이상기반(Anomaly-Based) IDS는 네트워크 또는 호스트의 일반적인 동작과 다른 것으로 추정되는 트래픽 또는 응용 컨텐트를 시스템 운영자에게 알려준다[7]. 이러한 이상기반 IDS는 일반적으로 이것을 스스로 학습하여 이룬다. 이상탐지에서, 시스템 관리자는 네트워크의 트래픽 로드, 고장, 프로토콜, 그리고 일반적인 패킷 크기에 대한 기준선 또는 일반 상태를 정의한다. 이상탐지자(Anomaly Detector)는 네트워크 세그먼트를 모니터하여 정의된 기준과 그들의 상태를 비교하여 이상을 찾는다.

(2) 네트워크 기반 IDS와 호스트기반 IDS

네트워크기반(Network-Based) IDS에서 센서는 감독할 네트워크 또는 종종 DMZ나 네트워크 경계에 위치한다[8]. 센서는 악의적 트래픽 탐지를 위해 모든 네트워크 트래픽의 패킷을 캡처하여 각각의 내용을 분석한다. 호스트 기반(Host-Based) IDS의 경우 센서는 보통 그것이 설치된 호스트의 모든 활동을 감시하는 에이전트 역할을 한다. 이 두 가지 방식이 혼합된 하이브리드 시스템도 사용되고 있다.

(3) 수동적 IDS와 반응적 IDS

수동적 IDS에서 센서는 가능성 있는 보안 침해 사항을 탐지하여, 정보를 로그로 기록하고 콘솔을 통해 경고 신호를 보낸다. 반응적 IDS에서 센서는 의심스러운 동작에 대해 자율적으로 또는 시스템 운영자에 의해 사용자를 로그 오프 시키거나, 방화벽을 다시 프로그래밍하여 의심스러운 악의적 출처로부터 네트워크 트래픽을 차단하도록 대응한다[9].

2.2 분류 알고리즘

기계학습에서 분류알고리즘(Classification Algorithm)으로 많은 종류가 있지만 랜덤포레스트[2], Naive Bayes[4], J48[5][6][10][11] 등의 알고리즘이 유용하게 널리 사용되고 있다.

2.2.1 랜덤포레스트 알고리즘

랜덤포레스트 알고리즘은 2001년 Breiman에 의하여 개발되었다[2]. 랜덤포레스트 알고리즘은 아래와 같다.

L은 데이터 케이스 N개로 구성된 훈련집합, M는 분류자의 변수갯수, B는 앙상블에 있는 분류자 수, Ly는 L의 클래스 멤버십이라고 하자.

for b=1 to B

단계1: 훈련집합 L에서 B개의 부트스트랩샘플 L1, L2, .. LB을 생성한다

단계2: 부트스트랩샘플로부터 랜덤특성선택(Random Feature Selection)을 이용하여 랜덤포레스트 트리를 작성한다.각 부트스트랩샘플에서 랜덤특성선택시에는 가장 좋은 성능을 나타내는 (M)**0.5 또는 M/3개의 변수를 선택한다.

단계3: 훈련집합의 분류자 Cb(x)를 생성한다.

단계4: 다수투표를 이용하여 훈련집합의 분류자 B를 취합한다.

C*(x)= arg max y sigma(b=1..B) I [Cb(x)=y]

2.2.2 ID3와 C4.5 알고리즘

ID3알고리즘은 1986년 Quinlan에 의하여 개발된 속성기반 기계학습알고리즘으로 주어진 훈련세트를 이용하여 top-down 방식으로 모든 노드에서 각 속성을 테스트하며 greedy 탐색을 한다[11][12]. 각 분류 트리에서 어떤 속성을 선택할 지를 결정할 때 정보획득(Information Gain)을 이용한다. 속성의 정보획득(Information Gain)은 훈련예제를 타켓분류에 따라 얼마나 잘 분리시키는 지를 나타낸다. 그리하여 ID3알고리즘은 다음과 같이 작동한다.

1) 먼저 훈련세트에서 정보획득이 가장 큰 속성을 선택한다.
2) 이 속성을 트리의 출발로 한다. 그리고 이 속성의 각각의 값으로 노드를 분할한다.
3) 이 프로세스를 훈련세트를 이용하여 속성이 없어질 때까지 반복한다.

ID3알고리즘은 어떤 속성의 입력값이 너무 많은 경우에 성능이 저하될 수 있다.

Quinlan은 이 ID3알고리즘을 개선하여 C4.5 알고리즘[5]으로 만들었는데 이를 자바로 구현하여 J48로 명명하였다[6].

2.2.3 Naive Bayes 알고리즘

이 알고리즘은 통계학에서 널리 알려진 Bayes 규칙을 기본적으로 이용한다. 그리하여 특성들 사

이의 독립을 가정하는 베이즈 정리를 적용한 확률 분류기의 일종으로 1950년대 이후 광범위하게 연구되고 있다[4]. 그리하여 이 알고리즘은 텍스트 분류에 사용됨으로써 문서를 여러 범주중 하나로 판단하는 문제에 대하여 효율적으로 사용되고 있다.

2.3 성능평가 척도

일반적으로 분류에서 성능평가는 [표 1]에 나타낸 바와 같은 척도들이 사용된다[12].

[표 1] 성능평가 척도
[Table 1] Measures of Performance Evaluation

척도	의미
TP:True Positive 또는 DR:Detection Rate	올바른 예를 양으로 판정. (침입탐지에서는 공격이고 경고가 발생함)
FP:False Positive	틀린 예를 양으로 판정. (침입탐지에서는 공격이 아니고 경고가 발생)
TN: True Negative	틀린 예를 부로 판정. (침입탐지에서는 공격이 아니고 경고가 발생하지 않음)
FN:False Negative	올바른 예를 부로 판정. (침입탐지에서는 공격이고 경고가 발생하지 않음)

[표 1]에 나타낸 4가지 척도들간의 관계는 다시 [표 2]에 나타낸 바와 같이 정리된다[12]. 어떤 분류결과를 평가할 때 정밀도, 재현율, 특별성, 정확도 등을 [표 2]에 나타낸 바와 같이 계산할 수 있다.

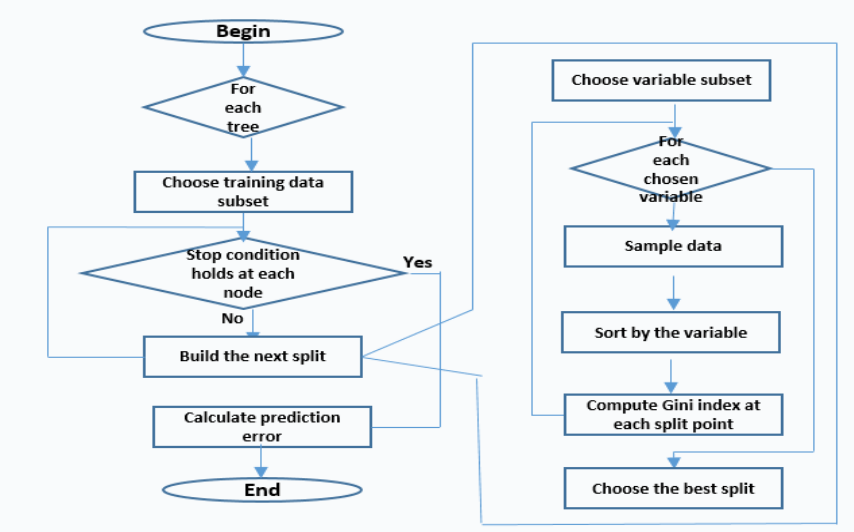
[표 2] 척도들간의 관계
[Table 2] Relationship Between Measures

구분		실제 정답		척도
		올바른 예 (True)	틀린 예 (False)	
실험결과	양(Positive)	TP 또는 DR	FP	정밀도 (Precision) = TP/(TP+FP)
	부(Negative)	FN	TN	
척도		재현율(Recall), 감도율(Sensitivity)= TP/(TP+FN)	특별성(Specificity, True Negative Rate)= TN/(TN+FP)	정확도(Accuracy)= (TP+TN)/(TP+TN+FP+FN)

3. 랜덤포레스트 알고리즘

3.1 랜덤포레스트 처리과정과 의사코드

2.2절에서 설명한 랜덤포레스트 알고리즘은 다시 [그림 1]과 같이 처리흐름을 나타낼 수 있다 [13].



[그림 1] 램덤포레스트 부트스트래핑내에서 처리과정

[Fig. 1] Flowchart in Bootstraping of Random Forest

이 처리를 위한 전체적인 의사코드는 다음과 같이 총 10개로 구성되어 있다[14].

D: 입력자료 차원

매개변수 λ, m: 구조상의 포인트 τ: 정보획득량 (Information Gain)

변수의미 L:리프, S: 분할, X,Y:좌표

의사코드1: BuildTree : 초기에는 전체공간을 다루는 TreeRoot 하나만 보유함.

```
BuildTree(λ, m , τ){
SelectCandidateDivideDimensions(TreeRoot, min(1 + Pr(λ), D))
for t = 1 . . . do
    Receive (Xt, Yt, It) from the environment
    Lt ← leaf containing Xt
```

```
    if It = estimation then
        UpdateEstimateStatistic(Lt, (Xt, Yt))
        for all S ∈ CandidateDivides(Lt) do
            for all L ∈ CandidateChildren(S) do
                if Xt ∈L then
                UpdateEstimateStatistic(L, (Xt, Yt))
                end if
            end for
        end for
    else if It = structure then
            if Lt has fewer than m candidate Divide points then
                for all d ∈ CandidateDivideDimensions(Lt) do
                  CreateCandidateDivide(Lt, d, πdXt)
                end for
            end if
            for all S ∈ CandidateDivides(Lt) do
                for all L ∈ CandidateChildren(S) do
                    if Xt ∈L then
                    UpdateStructuralStatistics(L, (Xt, Yt))
                    end if
                 end for
            end for
            if CanDivide(Lt) then
                if ShouldDivide(Lt) then
                Divide(Lt )
                else if MustDivide(Lt) then
                Divide(Lt )
                end if
            end if
    end if
  end for
  }
```

의사코드 2: Divide
리프 L, L에는 타당한 분할이 적어도 하나 존재

```
Divide(L){
for L
D ← BestDivision(L)
L′ ← LeftChild(L)
SelectCandidateDivideDimensions(L′ , min(1 +Pr(λ), D))
L“ ← RightChild(L)
SelectCandidateDivideDimensions(L“ , min(1 +Pr(λ), D))
return L′, L“
}
```

의사코드 3: CanDivide

```
CanDivide( L){
d ← Depth(L)
for all S ∈ CandidateDivides(L) do
        if DivideIsValid(L, S) then
        return true
        end if
end for
return false
}
```

의사코드 4: DivideIsValid
리프 L, 분할 S

```
DivideIsValid ( L , S){
d ← Depth(L)
L′ ← LeftChild(S)
L“ ← RightChild(S)
return Ne(L′) ≥ α(d) and Ne(L“) ≥ α(d)
```

```
}
```

의사코드 5: MustDivide

```
MustDivide (L){
d ← Depth(L)
return Ne(L) ≥ β(d)
}
```

의사코드 6: ShouldDivide

```
ShouldDivide {L} {
for all S ∈ CandidateDivides(L) do
    if InformationGain(S) > τ then
        if DivideIsValid(L, S) then
        return true
        end if
    end if
end for
return false
}
```

의사코드 7: BestDivision

```
BestDivision(L) {
best Divide ← none
for all S ∈ CandidateDivides(L) do
    if InformationGain(L, S) ≥ InformationGain(L, best_Divide)
    then
        if DivideIsValid(L, S) then
        best_Divide ← S
        end if
    end if
end for
return best_Divide
```

```
}
```

의사코드 8: InformationGain

```
InformationGain (L, S) {
L′ ← LeftChild(S)
L“ ← RightChild(S)
return Entropy(Y S(L))−[NS(L′)/NS(L)] Entropy(Y S(L′))− [NS(L)/NS(L“)] Entropy(Y S(L“))
}
```

의사코드 9: UpdateEstimateStatistic

```
UpdateEstimateStatistic (L ,Y(point) {
Ne(L) ← Ne(L) + 1
Ye(L) ← Ye(L) + Y
}
```

의사코드 10: UpdateStructureStatistics

```
UpdateStructuralStatistics (L ,Y) {
NS(L) ← NS(L) + 1
YS(L) ← YS(L) + Y
}
```

4. 성능평가

4.1 KDD Cup 1999

성능평가를 위하여 KDD Cup 1999를 이용하였다. KDD Cup 1988은 MIT 대학에서 수행한 DARPA 1998 침입탐지시스템 평가 프로그램에서 수집된 데이터를 기반으로 만들어진 데이터집합이다[3]. 이 데이터집합은 네트워크보안의 연구에 아주 유용하게 사용되고 있다. 이 데이터는 [표 3]에 나타낸 바와 같이 지속기간, 프로토콜종류 등 총 41개의 속성으로 구성된다. 각 레코드가 어떤 공격인 지를 나타내는 클래스까지 합하면 총 42개의 속성으로 이루어진다. 각 공격은 [표 4]에 나타낸 바와 같이 총 22개로 분류되는 데 주요 공격유형은 DoS(Denial of Service), Probe, R2L(Remote to User), U2R(User to Root)이다. 그리고 정상적인 Normal로 분류된다.

[표 3] KDD Cup 1999의 속성
[Table 3] Attributes of KDD Cup 1999

번호	속성명	번호	속성명	번호	속성명
1	Duration	15	su_attempted	29	same_srv_rate
2	Protocol_type	16	num_root	30	diff_srv_rate
3	Service	17	num_file_creations	31	srv_diff_host_rate
4	Flag	18	num_shells	32	dst_host_count
5	src_bytes	19	num_access_files	33	dst_host_srv_count
6	dst_bytes	20	num_outbound_cmds	34	dst_host_same_srv_rate
7	Land	21	is_host_login	35	dst_host_diff_srv_rate
8	wrong_fragment	22	is_guest_login	36	dst_host_same_sre_port_rate
9	Urgent	23	Count	37	dst_host_srv_diff_host_rate
10	Hot	24	srv_count	38	dst_host_serror_rate
11	mun_failed_logins	25	serror_rate	39	dst_host_srv_serror_rate
12	logged_in	26	srv_serror_rate	40	dshost_rerror_rate
13	num_compromised	27	rerror_rate	41	dshost_srv_rerror_rate
14	root_shell	28	srv_rerror_rate		

[표 4] KDD Cup 1999의 공격클래스
[Table 4] Attack Classes of KDD Cup 1999

공격의 주요 클래스	22개의 공격클래스
DoS	back, land, neptune, pod, smurt, teardrop
R2L(Remote to User)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient
U2R(User to Root)	buffer_overflow, perl, loadmodule, rootkit
Probing	ipsweep, nmap, portsweep, satan

4.1.1 데이터 세트의 분류

이 KDD Cup 1999는 3개의 데이터세트로 구성되어 있다. 먼저 가장 큰 전체 KDD 세트는 총 4,898,430 개의 레코드가 있는데 이는 훈련세트로 사용된다. 이 훈련세트가 너무 방대하여 이중에서 약 10%를 추출한 세트를 10%KDD라고 하여 여기에는 494,020 개의 레코드로 구성되어있다. 그리고 수정된(Corrected) KDD로 알려진 311,029개의 레코드로 구성된 테스트세트가 있다. 이들의 주 분류는 [표 5]에 나타낸 바와 같다[15].

[표 5] KDD Cup 1999 구성
[Table 5] Composition of KDD Cup 1999

자료세트 구분	전체	DoS	Probe	R2L	U2R	Normal
전체 KDD (훈련세트)	4,898,430	3,883,370	41,102	11,260	520	972,780
10% KDD	494,020	391,458	4,107	8,606	70	60,593
시험세트	311,029	229,853	4,166	1,126	52	97,277

4.2 전처리

성능평가를 위하여 전처리를 하는데 각 속성들의 값을 텍스트나 기호를 [표 6]과 같이 변환한다.

[표 6] 속성값 변환
[Table 6] Translation of Attribute Value

속성	protocol type	공격 클래스	length	src_bytes, dst_bytes
값	0:TCP 1:UDP 2:ICMP	0:정상 1:Probe 2:서비스거부 3:U2R 4:R2L	log[0,60000]=[0.0, 4.78]	log[0,1.3 billion]=[0.0, 9.14]

4.3 랜덤포레스트알고리즘의 적용

전처리를 거친 10% KDD세트를 이용하여 훈련을 실행하였으며, 이를 테스트세트에 적용하여 성능 측정치를 도출하였다. 또한 같은 방법으로 Weka[16][17]에 공개되어있는 Naive Bayes, J48를 이용하여 성능측정치를 도출하였다.

4.3.1 알고리즘간의 성능비교

4개 알고리즘의 성능을 비교평가하기 위하여 앞서 2장에서 설명한 측정치를 도출하였다. [표 7]에는 TP 비율과 FP 비율을 나타냈고, [표 8]에는 Precision 비율과 Recall 비율을 나타냈다. 4개의 비율 모두에서 랜덤포레스트의 성능이 가장 좋았으며, 다음에는 J48, Naive Bayes 순으로 좋은 성능을 나타내고 있다.

[표 7] TP비율과 FP비율 비교

[Table 7] Comparison of TP Ratio and FP Ratio

항목	TP 비율			FP 비율		
Class	Naive	J48	Random Forest	Naive	J48	Random Forest
back	0.977	1.000	1.000	0.005	0.000	0.000
teardrop	0.995	1.000	1.000	0.001	0.000	0.000
loadmodule	0.778	0.667	1.000	0.000	0.000	0.000
neptune	0.996	1.000	1.000	0.000	0.000	0.000
rootkit	0.700	0.100	1.000	0.003	0.000	0.000
phf	1.000	1.000	1.000	0.000	0.000	0.000
satan	0.954	0.995	0.999	0.002	0.000	0.000
buffer_overflow	0.200	0.933	1.000	0.000	0.000	0.000
ftp_write	1.000	0.500	1.000	0.003	0.000	0.000
land	1.000	0.952	1.000	0.000	0.000	0.000
spy	1.000	0.000	1.000	0.000	0.000	0.000
ipsweep	0.970	1.000	0.998	0.009	0.000	0.000
multihop	0.429	0.571	1.000	0.000	0.000	0.000
smurf	0.999	1.000	1.000	0.000	0.000	0.000
pod	0.909	1.000	0.996	0.037	0.000	0.000
perl	1.000	1.000	1.000	0.000	0.000	0.000
ware2client	0.470	0.983	1.000	0.006	0.000	0.000
nmap	0.442	0.957	0.991	0.001	0.000	0.000
imap	0.917	0.833	1.000	0.000	0.000	0.000
waremaster	0.950	0.900	1.000	0.001	0.000	0.000
portsweep	0.905	0.955	1.000	0.001	0.000	0.000
normal	0.653	1.000	1.000	0.001	0.000	0.000
guess_passwd	0.981	0.943	1.000	0.001	0.000	0.000
가중평균	0.928	1.000	1.000	0.000	0.000	0.000

ISSN: 2384-101X JNCIST

[표 8] 정밀도와 감도율 비교

[Table 8] Comparison of Precision and Recall

항목	정밀도			감도율		
Class	Naive	J48	Random Forest	Naive	J48	Random Forest
back	0.454	1.000	1.000	0.977	1.000	1.000
teardrop	0.638	1.000	1.000	0.995	1.000	1.000
loadmodule	0.029	0.545	1.000	0.778	0.667	1.000
neptune	1.000	1.000	1.000	0.996	1.000	1.000
rootkit	0.004	1.000	0.909	0.700	0.100	1.000
phf	0.114	1.000	1.000	1.000	1.000	1.000
satan	0.578	0.996	1.000	0.954	0.995	0.999
buffer_overflow	0.028	0.875	0.968	0.200	0.933	1.000
ftp_write	0.006	0.800	1.000	1.000	0.500	1.000
land	0.375	0.952	1.000	1.000	0.952	1.000
spy	1.000	0.000	1.000	1.000	0.000	1.000
ipsweep	0.210	0.998	0.998	0.970	1.000	0.998
multihop	0.094	0.800	1.000	0.429	0.571	1.000
smurf	1.000	1.000	1.000	0.999	1.000	1.000
pod	0.014	0.996	1.000	0.989	1.000	0.996
perl	0.750	0.750	1.000	1.000	1.000	1.000
ware2client	0.143	1.000	1.000	0.470	0.983	1.000
nmap	0.157	0.995	1.000	0.442	0.957	0.991
imap	0.149	1.000	1.000	0.917	0.833	1.000
waremaster	0.068	0.900	1.000	0.950	0.900	1.000
portsweep	0.657	0.998	0.999	0.905	0.995	0.999
normal	0.997	0.999	1.000	0.653	1.000	1.000
guess_passwd	0.068	1.000	1.000	0.981	0.943	1.000
가중평균	0.989	1.000	1.000	0.928	1.000	1.000

[표 9]에는 침입유형별 정밀도와 감도율 비교를 나타내고 있는데 역시 랜덤포레스트, J48, Naive Bayes 순으로 성능을 보이고 있다.

[표 9] 침입유형별 정밀도와 감도율 비교
[Table 9] Comparison of Precision and Recall

Class	Naive	J48	Random Forest
Normal	0.6530	1.000	1.0000
Probe	0.8178	0.9868	0.9970
DoS	0.6695	0.9920	0.9993
U2R	0.9927	0.6750	1.0000
R2L	0.8434	0.7163	1.0000

5. 결론

최근 들어 보안의 중요성이 강조되는 상황에서 네트워크보안은 중요한 부분이라고 할 수 있다. 분류분야에서 유용한 랜덤포레스트알고리즘을 KDD Cup 1999를 이용하여 네트워크침입탐지에 적용하였다. 그리고 성능평가를 위하여 이 결과를 분류분야에서 사용되는 Naive Bayes, J48과 비교하였다. 그리하여 Naive Bayes, J48 알고리즘에 대한 Weka 프로그램을 이용하여 적용한 결과와 비교하였다. 분류분야에서 사용되는 평가척도인 TP비율, FP비율, 정밀도, 감도율 등을 이용하여 성능비교한 결과 모든 평가척도에서 랜덤포레스트, J48, Naive Bayes의 순으로 좋은 성능을 나타냈다. 이는 랜덤포레스트알고리즘이 네트워크침입탐지에 적용한 경우에도 유용성을 나타낸다고 볼 수 있다.

향후 랜덤포레스트알고리즘의 네트워크침입탐지에의 적용을 위하여 분산서비스거부공격 등의 자료세트를 이용하여 추가적인 연구가 필요하다.

ISSN: 2384-101X JNCIST

References

[1] http://www.windowsecurity.com/articles-tutorials/intrusion_detection/Intrusion_Detection_Systems_IDS_ art_I__network_intrusions_attack_symptoms_IDS_tasks_and_IDS_architecture.html, Retrieved: September 2 (2014)

[2] Leo Breiman, "Random Forests", Machine Learning, (2001), Vol. 45, Issue 1, pp.5 – 32

[3] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Retrived: September 6 (2014)

[4] Tina R. Patil and S. S. Sherekar, "Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification", International Journal Of Computer Science And Applications, (2013), Vol.6, No.2, pp.256-261

[5] http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html, Retrived: September 8 (2014)

[6] Jay Gholap, "Performance Tuning of J48 Algorithm for Prediction of Soil Fertility", Asian Journal of Computer Science and Information Technology, (2012), Vol.2, No.8, pp.10-14

[7] Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," IEEE Communications Surveys & Tutorials, (2014), Vol.16, No.1, pp. 303-336

[8] S.Vasanthi and Dr. S.Chandrasekar ,"A Study on Network Intrusion Detection and Prevention System Current Status and Challenging Issues", Proc. of Int. Conf. on Advances in Recent Technologies in Communication and Computing, (2011), Sept. 14-15; Bangalore, India

[9] http://netsecurity.about.com/cs/hackertools/a/aa030504.htm, Retrived: September 10 (2014)

[10] A. Chauhan, G. Mishra and G. Kumar, "Survey on Data Mining Techniques in Intrusion Detection" , International Journal of Scientific & Engineering Research, (2011), Vol.2, Issue. 7, pp.1-4

[11] https://www.cs.umd.edu/grad/scholarlypapers/papers/Bahety.pdf, Retrived: September 11 (2014)

[12] https://ko.wikipedia.org/wiki/precision_recall, Retrived: September 12 (2014)

[13] http://home.etf.bg.ac.rs/~vm/os/dmsw/Random%20Forest.pptx, Retrived: September 13 (2014)

[14] http://proceedings.mlr.press/v28/denil13-supp.pdf, Retrived:September 15 (2014)

[15] Safaa O.Al-mamory and Firas S. Jassim, "Evaluation of Different Data Mining Algorithms with KDD CUP 99 Data Set", Journal of Babylon University/Pure and Applied Sciences, (2013), Vol.21, No.8 pp.2263-2681

[16] http://www.cs.waikato.ac.nz/ml/weka, Retrived: September 20 (2014)

[17] http://weka.sourceforge.net, Retrived: September 20 (2014)