LCSeq를 이용한 변형 웜 시그니쳐 생성 엔진 구현

Implementation of Engine Generating Mutation Worm Signature Using LCSeq

고준상*, 김봉한**, 이재광*
한남대학교 컴퓨터공학과*, 청주대학교 컴퓨터정보공학과**

Joon-Sang Ko(gojs@netwk.hannam.ac.kr)*, Bong-Han Kim(bhkim@cju.ac.kr)**,
Jae-Kwang Lee(Jklee@netwk.hannam.ac.kr)*

요약

본 논문에서는 알려지지 않은 변형 웜을 탐지하기 위한 방법을 제안한다. 그 방법으로, 페이로드 영역에서 시그니쳐 생성 방안들을 패턴인식 알고리즘으로 연구되었던 Suffix Tree중에서 Longest Common Subsequence(LCSeq) 기법을 이용하여 새로운 시그니쳐를 자동적으로 생성할 수 있는 프로그램을 설계하여 구현하였다. 테스트를 통해 코드레드 웜과 님다 웜의 변종을 검출하는 과정을 보여주고 기존 snort의 시그니쳐와 LCSeq를 이용해 생성된 시그니쳐를 비교·평가하였다.

■ 중심어 : | LCSeq | 변형 웜 탐지 | 웜 시그니쳐 |

Abstract

We introduce the way to detect the mutation worm. We implemented the program that can generate signature using LCSeq(Longest Common Subsequence) technique in Suffix Tree studied as pattern recognition algorithm. We also showed the process to detect the mutation of CodeRed worm and Nimda worm and evaluated signatures generated by snort and LCSeq.

■ keyword : | LCSeq | Mutation Worm Detection | Worm Signature |

I. 서 론

웜 바이러스는 인터넷의 발전을 통해 특정 시스템에서 짧은 시간동안 전세계적으로 피해를 확산시킬 수 있게 되었다. 현재 전세계적으로 발생하는 인터넷 웜은 이메일을 통한 전파로 브로드캐스트 또는 트래픽의 급격한 증가, 취약점 공격과 같은 심각한 피해를 네트워크에 발생시키고 있다. 따라서 이러한 웜을 탐지하기 위하여 다양한 침입탐지 기법이 개발되어 상용화 되었다. 그러나 현재 개발되어서 사용되고 있는 웜의 탐지 기법으로는 다양하게 스스로 변형되는 웜에 대해 다음과 같은 문제점을 갖는다; 트래픽이 과중한 네트워크, 스위치 네트워크, 비대칭 네트워크, 탐지 후 대응의 불가능, 대응까지 오랜 시간 소요, 과도한 분석 데이터의 축적, 판정 오류.

변형 웜은 시그니쳐 기반의 침입탐지 기법보다는 비정상행위 탐지 기법으로 탐지 가능하다. 현재 변형 웜을 탐지하기 위해서 많은 연구가 진행되고 있지만, 실시간 탐지에 적용하기에는 많은 무리가 있다. 실시간 탐지를 위해 가장 많이 사용되는 탐지기법으로는 트래픽의 폭주여부를 판단하는 트래픽 비정상행위(anomaly) 탐지 기법과 프로토콜의 적법성 및 비정상

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었습니다. (IITA-2007-C1090-0701-0027)

접수번호 : #070813-001
접수일자 : 2007년 08월 13일
심사완료일 : 2007년 10월 22일
교신저자 : 김봉한, e-mail : bhkim@cju.ac.kr

적 사용 여부를 판단하는 프로토콜 비정상행위 기법이 많이 사용되고 있다[1].

본 논문에서는 능동적으로 변형 웜을 탐지할 수 있는 방안과 변형 웜의 탐지를 위한 시그니쳐 생성 엔진을 연구하고자 한다. 시그니쳐 생성 엔진은 트래픽 비정상행위 탐지 엔진과 프로토콜 비정상행위 탐지 엔진에서 전달받은 공격 패킷을 시그니쳐 규칙으로 생성하는 기능을 담당한다. 시그니쳐 규칙은 IDS와 정의되는 필드 영역, 패킷 헤더, 페이로드 영역으로 구성된다. 본 논문에서 적용하고자 하는 시그니쳐 생성 엔진은 웜이 변형할 때 주로 변경되는 페이로드 영역을 기반으로 설계하고 변형 웜 시그니쳐 생성을 위해 Suffix Tree에서 Longest Common Subsequence(LCSeq) 기법을 이용하여 구현하고자 한다.

II. 관련 연구

현재 능동적인 시그니쳐 생성을 위해 다양한 국제적인 연구가 진행되고 있다. Kai Hwang은 CAIDS(Cooperative Anomaly and Intrusion Detection System)을 개발하였다. CAIDS는 시그니쳐 생성기를 통하여 대화식으로 운영되는 네트워크-기반 침입탐지 시스템과 비정상행위 탐지 시스템으로 구축된다. CAIDS은 NIDS와 ADS가 결합하였을 때의 장점을 보여주고 있다[2].

Jian Zhang은 RAIRS(Rollbackable Automated Intrusion Response System)을 개발하였다. RAIRS의 대응 롤백 메커니즘은 대응 조치가 롤백 되는지 아닌지를 자동적으로 결정한다. RAIRS는 오탐지(False Positive)에 의한 오경보를 처리할 수 있고, 불필요한 대응 조치라도 롤백할 수 있도록 한다. 이것은 전통적인 대응 시스템의 단점을 효과적으로 극복할 수 있다[3].

Hyang-Ah Kim은 Autograph를 개발하였다. Autograph는 수상한 흐름 선택 부분과 시그니쳐 생성 부분을 통해 자동적으로 웜의 시그니쳐를 탐지하는 방법을 제안하였다[4]. Ke Wang은 PAYL을 통해 비정상적인 페이로드 기반 웜 탐지와 시그니쳐 생성 방안을 제안하였다[5].

III. 페이로드 영역에 대한 규칙 생성 방안

트래픽 비정상행위 탐지 엔진에서 전달받은 여러 개의 패킷에 대한 페이로드 중에서 특정 문자열을 시그니쳐 규칙의 시그니쳐 필드에 저장하기 위해서는 여러 개의 패킷 중에서 일치하는 문자열을 추출해야 한다. 동일 문자열 추출을 위해서 문자열을 전 처리한 후, 결과를 자료 구조로 만들어 두는 인덱스 데이터 구조 중에서 Suffix Tree를 이용한다.

Suffix Tree는 m 길이를 가지는 스트링 S는 1~m으로 기록된 m개의 노드가 직접적으로 루트에 연결된 트리 형태를 의미한다. 하나의 노드로부터 생성된 자식 중에서 동일한 문자로 시작된 Edge label은 존재하지 않는다. Suffix Tree를 이용하여 스트링에 대한 전처리 결과를 이용하여 두 문자열 간에 동일 스트링을 추출하는 접근 방법에는 3가지로 정의할 수 있다. 다음은 Suffix Tree의 종류들이다. 각 방법들의 특징을 분석하여, 단편화 및 변경(Mutation)된 패킷에 대해서도 동일 시그니쳐를 생성할 수 있고, 오탐지를 최소화할 수 있는 알고리즘을 선택하였다[6][7].

1. String equality(SE)

SE는 가장 직관적인 접근방법이다. 의심스러운 패킷들에서 연속된 동일한 문자열만을 추출하는 방법이다. 이 방법은 매우 정확한 동일 문자열을 추출할 수 있고, 오탐지를 줄일 수 있다는 점에서 매우 유용하게 사용가능하다.

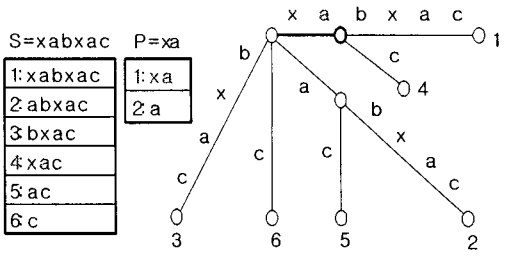


그림 1. String Equality

그러나 패킷을 미세하게 변경시키기만 해도, 동일 문자열을 찾을 수 없다. 웜이 전파하면서 미세하게 자신의 페이로드를 변경시키거나, 단편화를 발생시키면 이

접근 방법으로 시그니쳐를 추출할 수 없게 된다. [그림 1]은 두 문자열에 대해서 SE를 적용한 예를 보이고 있다.

2. Longest Common Substring(LCS)

LCS는 SE처럼 정확하게 동일문자열을 추출한다는 점에서 동일하다. 하지만, SE는 동일한 문자열이 다수개 일 경우 SE는 동일한 문자열은 나열하는데 그치지만, LCS는 다수개의 동일 문자열 중 가장 긴 문자열을 추출해 준다는 데 그 차이점이 있다.

가장 긴 문자열은 추출해 주면 일부 패킷이 단편화되거나 미세하게 변경되어도 시그니쳐를 추출 할 수 있다. LCS로 연속된 동일 문자열을 찾는 과정을 Suffix Tree로 나타내면 [그림 2]와 같다.

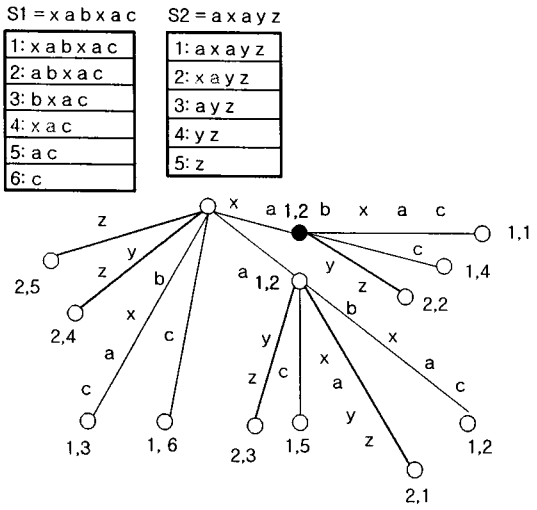


그림 2. Longest Common Substring

3. Longest Common Subsequence(LCSeq)

LCSeq는 LCS와 유사하다. 하지만, LCS는 동일 스트링의 순서에 상관없이 CS를 찾는 Order-insensitive 방법이지만, LCSeq는 동일 스트링의 순서를 고려하는 Order-sensitive 방법이다. 또한, LCS는 동일한 연속된 값을 찾아주지만, LCSeq는 연속되지 않은 동일한 스트링을 찾아 준다.

따라서 스스로 복제하며, 변형된 형태를 지니는 다형성(Polymorphic) 웜의 시그니쳐를 찾는데 유용하다. 그러나 LCS보다 오탐지가 높은 단점이 있다. 다음은 LCS로 연속된 동일 문자열을 찾는 과정의 예이다. 두 개의 주어진 문자열 S1= 'ABCDEFCBGH', S2= 'FECBGAGFHE'에서 동일한 문자열을 추출한다. [그림 3]과 같이 LCSeq를 이용하여 시그니쳐를 추출하면 동일한 문자 'FCBGH'가 추출된다.

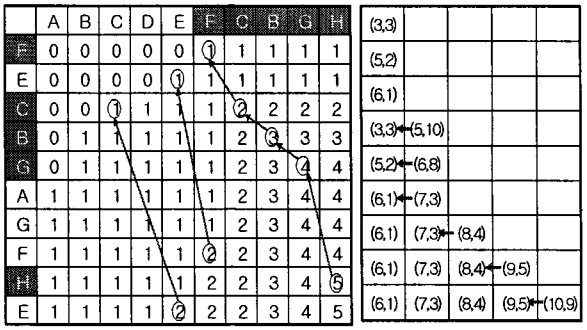


그림 3. LCSeq를 이용한 시그니쳐 추출

IV. 변형 웜 시그니쳐 생성 엔진 구현 및 테스트

1. 시그니쳐 생성 엔진 구현

본 논문에서는 스스로 복제하며, 변형된 형태를 지니는 다형성 웜의 시그니쳐를 찾는데 유용하고 연속되지 않은 동일한 스트링도 탐지할 수 있는 LCSeq를 이용하여 변형 웜 시그니쳐 생성 엔진을 구현하였다. 구현하기 위해 사용된 언어는 Java JDK 6u1이고 개발 툴은 Netbeans 5.5를 사용하였다. 연동 프로그램으로 WinPcap과 Jpcap을 사용하였다. 구현된 프로그램의 동작원리는 [그림 4]와 같다.

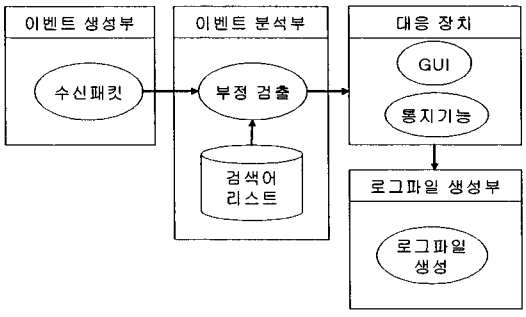


그림 4. 프로그램의 동작원리

[그림 5]는 구현된 프로그램의 메인 화면을 보여주고 있다. 구현된 프로그램의 주요기능은 분석, 통계, 탐지, 경고, 로그생성 등 5부분으로 구성되었다.

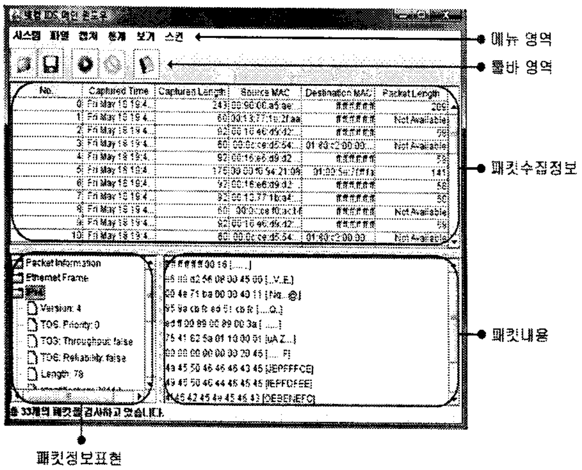


그림 5. 프로그램의 메인 화면

[그림 6]은 변형 웜을 추출하기 위한 LCSeq를 계산하는 소스 코드를 보여주고 있다.

```
import java.io.*;
public class Program {
  static final int MAX = 100;
  static char x[]=new char[MAX];
  static char y[]=new char[MAX];
  static int c[][]=new int[MAX][MAX];
  static int b[][]=new int[MAX][MAX];
  static int I,j;
  static int m = 0;
  static int n = 0;
public static int LCSlength() {
  for(i=1;i<=m;i++) c[i][0]=0;
  for(j=0;j<=n;j++) c[0][j]=0;
  for(i=1;i<=m;i++) {
     for(j=1;j<=n;j++) {
        if(x[i-1]==y[j-1]) {
           c[i][j]=c[i-1][j-1]+1;
           b[i][j]=1;   }
        else if(c[i-1][j]>=c[i][j-1]) {
           c[i][j]=c[i-1][j];
           b[i][j]=2; }
        else {
           c[i][j]=c[i][j-1];
           b[i][j]=3; } }           }
  return c[m][n];        }
public static void printLCS(int i, int j) {
  if(i==0 || j==0) return;
  if(b[i][j]==1) {
     printLCS(i-1,j-1);
     System.out.print(x[i-1]);}
```

```
  else if(b[i][j]==2)
     printLCS(i-1,j);
  else
     printLCS(i,j-1); }
public   static   void   main(String   args[])   throws
java.io.IOException {
  while(true) {
     int loop=0;
     int loop2=0;
     while(true) {
        x[loop]=(char)System.in.read();
        m++;
        if(x[loop++]=='\n') break; }
     m-=2;
  while(true) {
     y[loop2]=(char)System.in.read();
     n++;
     if(y[loop2++]=='\n') break;}
  n-=2;
  System.out.println("LCS length -> " + LCSlength());
  printLCS(m,n);
  System.out.println();
  m=0;
  n=0; } } }
```

그림 6. LCSeq 계산을 위한 소스 코드

2. 변형 웜 시그니쳐 생성 테스트

비정상 패킷을 위해 코드레드와 코드레드 변종을 시험 데이터로 이용하였다. [그림 7]은 코드레드 웜의 GET 필드와 일부 필드를 보여주고 있다. 특징으로는 피해 시스템의 버퍼를 가득 채우기 위해서 긴 문자열 N을 사용하고 있다.

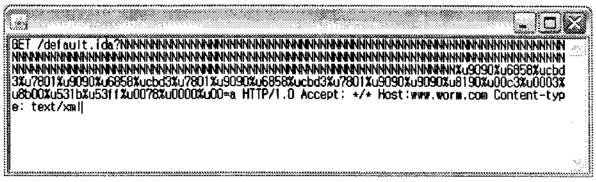


그림 7. CodeRed 웜의 페이로드

다음 [그림 8]은 코드레드의 변종으로 ida 값 사이에 임의의 값을 삽입하고, N 스트링 대신 X 스트링으로 변경한 차이를 가지고 있다. ida 값 내부에 임의의 값을 사용하는 이유는 기존의 시그니처를 우회하기 위하여 NOP(Non Operation) 코드를 문자열 사이에 삽입하여 복제되는 웜의 유형에 대해서도 시그니쳐를 생성하는지에 대한 여부를 테스트하기 위해서이다.

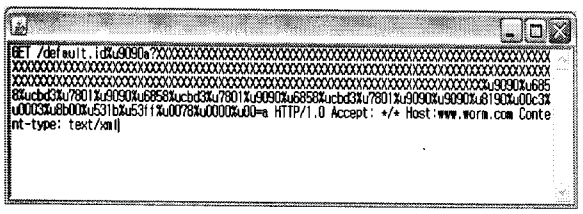


그림 8. CodeRed 웜 변종의 페이로드

다음 [그림 9]는 정상 패킷으로부터 동일 스트링을 추출하기 위한 테스트 데이터이다.

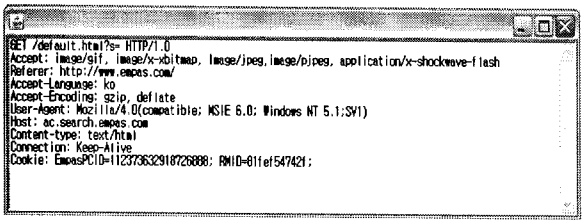


그림 9. HTTP 패킷 테스트 데이터

테스트 방법은 코드레드 패킷과 코드레드 변종에 대해서 LCSeq를 적용하여 첫 번째 시그니쳐 후보를 추출한다. 또한 두 개의 정상 패킷을 이용해 동일 스트링을 추출하여 Common Substring 리스트를 만든다. 마지막으로 첫 번째 시그니쳐 후보 내에서 Common Substring 리스트를 제거하여 새로운 시그니쳐를 생성해 낸다. [그림 10]은 새로운 시그니쳐를 생성하기 위한 테스트 과정을 보여주고 있다.

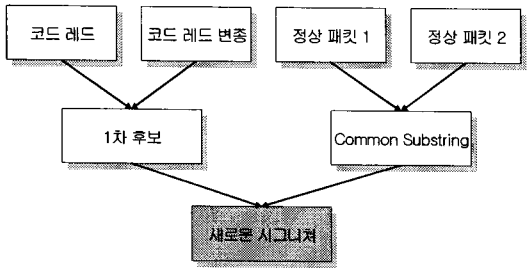


그림 10. New Signature 생성을 위한 테스트 과정

테스트 순서는 다음과 같다. 먼저, 정상 패킷에서 Common Substring 추출하고 코드레드와 코드레드 변종에 대해서 LCSeq를 적용하여 시그니쳐 후보 추출하고 Common Substring과 1차 후보에서 Common String 추출한다. 그리고 1차 후보에서 Common String을 삭제한 새로운 시그니쳐 목록을 생성한다.

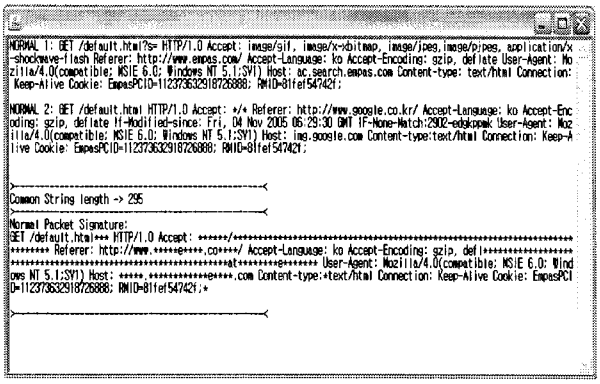


그림 11. 정상 패킷에서의 Common Substring 추출

[그림 11]은 우선적으로 정상 패킷들의 데이터 부분을 출력한다. 그 다음 두 패킷 간에 동일 스트링의 길이를 출력하고, 동일한 문자열을 출력하고 있다. 동일하지 않는 문자는 '*'로 표시하고 있다.

```
1: GET
2: /default.html
3: HTTP/1.0
4: Accept:
5: Referer:
6: http://www.
7: Accept-Language: ko
8: Accept-Encoding:gzip, deflat
9: User-Agent: Mozilla/4.0(compatible; MSIE 6.0; Windows NT
5.1;SV1)
10: Host:
11: .com
12: Content-type:text/html
13: Connection: Keep-Alive
14: Cookie: EmpasPCID=112373632918726888; RMID=81fef5474 2f;
```

동일 문자열은 다음과 같이 구성된다. 3byte 이하의 동일 문자열은 제외한다.

[그림 12]는 코드레드와 코드레드 변종 패킷에 대해서 Common Substring을 출력하고 있다. [그림 12]를 살펴보면, 정상패킷과 동일하게 코드레드와 변종에 대한 데이터 부분을 출력하고, 동일 문자열의 길이와 1차 후보를 출력하고 있다.

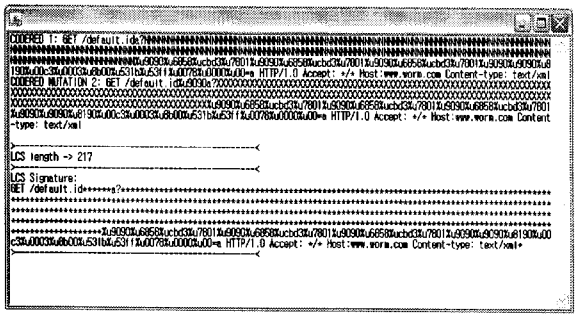


그림 12. CodeRed와 CodeRed 변종을 이용한 1차 후보

동일 문자열은 다음과 같이 구성된다.

```
1: GET /default.id******a?
2: u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u685
8%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53f
f%u0078% u0000%u00=a
3: HTTP/1.0
4: Accept:
5: Host:www.worm.com
6: Content-type: text/xml
```

[그림 13]은 첫 번째 시그니쳐 후보와 정상 패킷에서의 Common Substring을 입력으로 일치하는 동일 스트링을 나타내고 있다.

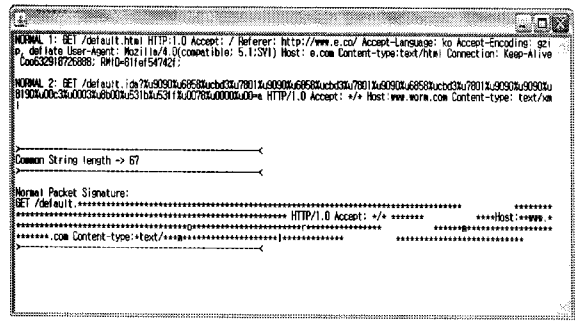


그림 13. 시그니쳐 후보와 Common Substring간에 동일 스트링

동일 문자열은 다음과 같이 구성된다.

```
1: GET /default.
2: HTTP/1.0
3: Accept:
4: host: www.
5: .com
6: Content-type: text/
```

따라서, 1차 후보에서 위 문자열을 제외하고 추출하면 동일 문자열은 [그림 14]와 같은 4개의 새로운 시그니쳐가 생성된다.

그림 14. 새로운 시그니쳐 생성

[표 1]은 현재 침입탐지용으로 광범위하게 사용되고 있는 Snort 시그니쳐와 구현한 변형 웜 시그니쳐 생성 엔진을 통해 생성된 시그니쳐를 비교하였다. 사용된 웜은 코드레드 웜과 님다 웜의 변형을 이용하였다.

표 1. Snort와 구현한 프로그램의 시그니쳐 비교

	Snort	구현한 프로그램
CodeRed	1: .ida 2: ida?	1: ida*a? 2: %u9090%u6858%ucbd3... 3: worm 4: xml
Nimda	1: cmd.exe	1: /..%c%.. 2: /winnt/system32/cmd.exe?/c+dir

코드레드 웜 탐지를 위해 Snort에서의 Uricontent는 '.ida' 또는 'ida?' 등의 스트링을 이용하고 있으며, 새로운 시그니쳐는 4개의 시그니쳐 생성을 보이고 있다.

Snort의 시그니쳐는 웜이 발생된 후, 전문가에 의해 수동으로 만들어진 규칙이지만 새로운 시그니쳐는 LCSeq를 이용하여 만들어진 시그니쳐이기 때문에 정교함은 떨어지지만 새로운 시그니쳐를 자동적으로 생성할 수 있다는 장점이 있다. 또한, 복제되면서 자기 스스로 변형된 형태를 지니는 특성을 갖는 웜의 유형에도 새로운 시그니쳐의 1번 항목처럼 'ida'와 'a' 사이를 '*'로 표기함으로서 다형성 웜들도 탐지가 가능하다.

그러나 결과에서 살펴보았듯이 다수개의 시그니쳐가 발생되기 때문에 오탐지가 발생하였다. 따라서 이러한 시도가 시스템에 적용되기 위해서는 자동으로 생성된 시그니쳐에 대한 오탐지를 얼마만큼 줄일 수 있는지 여부가 매우 중요한 이슈가 될 것이다.

탐지 정확성 시험을 위해 초기 님다 웜을 포함하는

16개의 변형 웜을 이용하였다[8]. 님다 웜과 정상 트래픽을 동시에 전송하여 성능 변화에 따른 탐지율을 측정하였다. 시험환경은 1Gbps까지 전송이 가능한 IXIA 트래픽 생성기를 snort와 구현한 프로그램에 각각 연결하여 탐지율을 확인하였다. 성능 평가는 [표 2]와 같다.

표 2. Snort와 구현한 프로그램의 성능 평가

IDS \ 백그라운드	0Mbps	100Mbps	500Mbps	1Gbps
snort	7 hits	7 hits	5 hits	1 hits
구현한 프로그램	14 hits	14 hits	11 hits	7 hits

백그라운드 없이 웜 트래픽을 전송 하였을 때는 snort와 제안한 프로그램의 탐지율이 높은 것으로 확인되었다. 하지만 1Gbps로 전송하였을 때는 탐지율이 현격하게 떨어진 것을 볼 수 있다. 그 이유는 snort 와 구현한 프로그램에서 사용한 pcap 라이브러리의 성능저하가 원인이 될 수 있다. 따라서, 제안한 방식을 FPGA 형태로 구현된다면 성능의 저하 없이 탐지율을 높일 수 있을 것으로 생각된다.

V. 결 론

본 논문은 패턴인식 알고리즘으로 연구되었던 Suffix Tree중에서 Longest Common Subsequence 기법을 이용하여 변형된 웜을 탐지할 수 있는 프로그램을 구현한 논문이다.

또한 능동적으로 변형 웜을 탐지할 수 있는 방안과 변형 웜의 탐지를 위한 시그니쳐 생성 엔진을 연구하였고, 테스트를 통해 코드레드 웜과 님다 웜의 변종을 탐지하는 과정을 각 단계별로 출력되는 생성물로 확인하였다. 또한 기존의 snort를 이용한 시그니쳐와 구현된 엔진을 이용한 시그니쳐를 비교・평가하였다.

다양한 변종이 발생하는 인터넷 웜을 탐지하기 위해서는 기존의 수동적인 시그니쳐 중심의 탐지 방법을 탈피하여 능동적으로 탐지할 수 있는 이상탐지 기법을 이용하여야 한다. 그런 관점에서 패턴인식 분야의 알고리즘들은 그 대안이라고 할 수 있다. 향후 연구방향으로는 구현된 프로그램을 대용량 네트워크에서도 적용 가능할 수 있도록 성능을 강화하고, 변형 웜 탐지가 가능한 많은 패턴 인식 알고리즘을 발굴하고자 한다.

참 고 문 헌

[1] C. Endorf, E. Schultz, and J. Mellander, *Intrusion Detection & Prevention*, McGrawHill, 2004.

[2] K. Hwang, Y. Chen, and H. Liu, "Defending Distributed Systems Against Malicious Intrusions and Network Anomalies," IPDPS'05, p.286a, 2005.

[3] J. Zhang, J. Gong, and Y. Ding, "Research on automated rollbackability of intrusion response," Journal of Computer Security, Vol.12, No.5, pp.737-751, 2004.

[4] http://www.cs.cmu.edu/~bkarp/autograph-usenixsec2004.pdf

[5] http://worminator.cs.columbia.edu/papers/2005/raid-cut4.pdf

[6] J. Newsome, B. Karp, and D. Song, "Polygraph: automatically generating signatures for polymorphic worms," Security and Privacy 2005 IEEE Symposium, pp.226-241, May 2005.

[7] G. Navarro and M. Raffinot, *Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences*, Cambridge University Press, 2002.

[8] http://www.linklogger.com/nimda.htm

저 자 소 개

고 준 상(Joon-Sang Ko) 정회원

- 1991년 2월 : 청주대학교 전자계산학과(공학사)
- 1997년 8월 : 한남대학교 전자계산공학과 (공학석사)
- 2004년 3월 ~ 현재 : 한남대학교 컴퓨터공학과(박사과정)

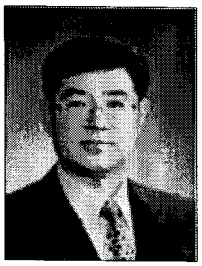
<관심분야> : 정보보호, 컴퓨터네트워크

김 봉 한(Bong-Han Kim) 정회원

- 1994년 2월 : 청주대학교 전자계산학과(공학사)
- 1996년 2월 : 한남대학교 전자계산공학과 (공학석사)
- 2000년 2월 : 한남대학교 컴퓨터공학과(공학박사)
- 2001년 3월 ~ 현재 : 청주대학교 컴퓨터정보공학과 교수

<관심분야> : 멀티캐스트, P2P, 정보보호

이 재 광(Jae-Kwang Lee) 종신회원



- 1984년 2월 : 광운대학교 전자계산학과 (공학사)
- 1986년 2월 : 광운대학교 전자계산학과 (공학석사)
- 1993년 2월 : 광운대학교 전자계산학과 (공학박사)
- 1986년 3월 ~ 1993년 8월 : 군산전문대학 전자계산학과 부교수
- 1993년 8월 ~ 현재 : 한남대학교 컴퓨터공학과 교수

<관심분야> : 컴퓨터네트워크, 정보통신, 정보보호