효율적 실행을 위한 개선 기능을 갖춘 안티바이러스 엔진의 플랫폼 독립적 구현에 관한 연구

김미애*, 박유미, 최주영, 유주영, 박은옥, 최은정, 김윤정, 김명주

서울여자대학교 대학원 컴퓨터학과

Platform-independent Implementation of Anti-Virus Engine with Enhanced Features for Execution Efficiency

Mi-Ae Kim.*, Yu-Mi Park, Ju-Young Choi,, Ju-Young Yoo

Eun-Ok Park, Eun-Jeong Choi, Yoon-Jeong Kim, Myuhng-Joo Kim

Department of Computer, Seoul Women's University

요 약

본 논문에서는 현재 공개되어 있는 ClamAV 안티바이러스 엔진의 소스를 분석하여 플랫폼 독립적인 구현을 시도했다. 구체적으로 유닉스 기반에서 Java 프로그래밍 언어를 사용했으며, 사용자의 편의를 위한 GUI 환경을 SWING을 사용하여 구축했다. 아울러 공개된 안티바이러스 엔진보다 효율성을 높이기 위하여 효율적인 매칭 알고리즘 선택 및 바이러스 패턴 데이터베이스의 재구성에 관하여 제안한다.

I. 서론

최근 정보보호 분야의 변화 추세 가운데 두드러진 현상은 "컴퓨터 바이러스"로 통칭되는 "악성 에이전트(malicious agent)"들의 기술 고도화 현상이다. Win32.Nimda나 Code Red 바이러스에서 볼 수 있듯이 초창기 바이러스와 달리 해킹 기법과 결합된 하이브리드 형태가 더욱 많이 출현하기 시작했으며, Ms SQL 웜 슬래머(Slammer)의 경우 MS SQL의 취약점을 공격, 트래픽 증가, DNS 서버의 다운을 통해 전세계 인터넷을 마비시키는 등 그 피해 규모가 엄청나다. 또한 특정 운영체제(MS windows 2000/NT/XP)의 취약점을 공격하는 Blaster, Welchia 웜과 같은 형태도 생겨나고 있다.

따라서 이러한 컴퓨터 바이러스들의 공격으로부터 인터넷 사회의 기반이 되는 인터넷 서버들을 보호하기 위한 연구가 이루어져야 한다.

이를 위해 본 논문에서는 우선 컴퓨터 바이러스들을 탐지 및 제거 기능을 가지는 기본 안티 바이러스 엔진의 구성을 알려진 안티 바이러스 소스를 통해 이해하고, 그것을 적용하여 안티 바이러스 엔진을 구현한다. 구현된 안티바이러스 엔진은 단순히 컴퓨터 바이러스를 탐지하고 삭제한다는 의미에서 다른 엔진들과 차이가 없지만, 서버로 사용되는 유닉스 기반에서 구현되었고, 플랫폼에 독립적인 JAVA를 프로그래밍 언어로 사용했으며, 사용자 편의를 고려한 GUI 환경을 구축했다는 측면에서 의의를 가진다.

또한 이런 안티바이러스 엔진의 효율성을 위해 안티 바이러스 엔진에서 사용되는 매칭(matching) 알고리즘과 바이러스 패턴(pattern) DB 구성에 대한 개선된 방법을 제안한다.

II. 안티바이러스 엔진의 구현

1. 기존 안티바이러스 엔진 분석

대부분의 안티바이러스 엔진은 상용화되어 그 내부 소스 구성에 대한 공개를 하지 않기 때문에, 본 논문에서는 현재 공개되어 있는 안티바이러스

엔진인 ClamAV를 기반으로 다룬다.

1) ClamAV의 전체적 구성

ClamAV는 Linux와 Unix에서 실행되는 안티바이러스 엔진으로 컴퓨터 바이러스 패턴 DB 파일과 탐색 한 파일의 패턴 매칭(pattern matching)을 통해 바이러스 검색 여부를 진단하게 된다.

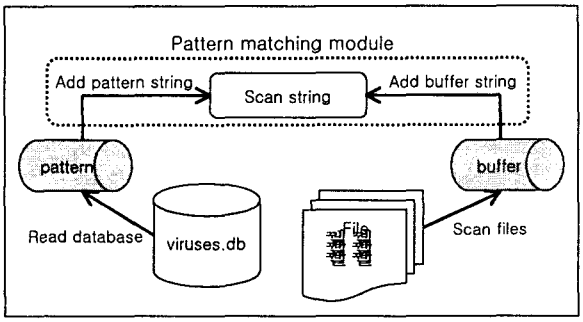

그림 1: ClamAV 구성

그림1은 ClamAV의 전체적인 구성을 대략적으로 나타내고 있다. 버퍼(Buffer) 두 개를 이용한 방법으로 하나의 버퍼에 컴퓨터 바이러스의 데이터베이스 값을 pattern 버퍼에 읽어 들이고, 다른 하나의 버퍼에 진단 대상이 되는 탐색 파일을 읽어 들인다. 그 중에 두 개의 값을 pattern matching module에서 비교함으로 컴퓨터 바이러스 감염 여부를 확인 할 수 있다.[1]

2) ClamAV의 바이러스 패턴 DB 구성

ClamAV는 현재 버전 6.0까지 발표되었고, 7000여개가 넘는 컴퓨터 바이러스 패턴(pattern)을 가지는 DB 파일을 가지고 있다. 또한 ClamAV 내부의 업데이트 모듈을 통해 지속적으로 컴퓨터 바이러스 패턴 DB 파일이 업데이트 되고 있다.

컴퓨터 바이러스 패턴(pattern)은 바이러스 실행 부분을 헥사 코드화한 시그너처(signature)를 말하는 것으로, ClamAV의 기본 바이러스 패턴 DB 파일의 구성은 "바이러스명", "핵사 시그너처", 두 가지를 분리하는 분리자인 "=", 이렇게 세부분으로 이루어진다.

```
Unix/Addy=56697275733d3c46696c653e3bda2040566972757 33d4056697
275735b302e2e2e32395d3bda20636c6f73652846696c65293bdada20666f72
6561636820244669 6c654e616d6520283c2f7661722f73706f6f6c2f6d61696c2
f2a3e29207bda2020696620282d722024
```

표 1: ClamAV 바이러스 패턴 DB 구성

2. 안티바이러스 엔진 구현

위에서 살펴본 공개 안티바이러스 엔진인 Clam-AV의 기본 개념인 패턴 매칭 방법을 적용한 안티바이러스 엔진을 구현한다.

구현된 안티바이러스 엔진에서 사용하는 안티바이러스 패턴 DB 파일은 ClamAV에서 사용하는 바이러스 패턴 DB 파일의 형식을 그대로 따른다.

1) 엔진의 구현 환경

유닉스 운영체제인 Sun Microsystems Solaris 8 환경에서, ClamAV가 C언어를 사용하여 구현된 것에 비해, 플랫폼에 독립적인 Java를 사용해 안티바이러스 엔진을 구현한다. 또한 사용자의 편의 제공을 위한 GUI환경 구현은 Java에서 지원하는 SWING을 사용하게 된다.

2) 엔진의 주요 모듈

구현된 안티바이러스 엔진의 주요모듈은 크게 두 부분으로 나뉘게 된다. 그중 첫 번째 모듈은 탐색 및 보고모듈이다. 탐색 방법은 하위 디렉터리와 파일의 순차적인 탐색을 위해 재귀호출 방을 사용했으며, 탐색 중 폴더인 경우는 경로명만을 출력해주고, 파일인 경우 바이러스 패턴 DB와의 매칭 여부를 확인 후 감염여부, 감염파일 수, 전체 탐색 파일 수를 보고하게 된다.

```
public void scan(File dir){
            ...
            ...
try{
  if(f1.isDirectory()){ //디렉토리인 경우
  System.out.println(f1.getAbsolutePath());
  scan(f2);
  }else{                                    //파일인  경우
System.out.println(f1.getAbsolutePath());
   String fn = f1.getPath();
   String shex = hex(fn);
   int t= fRead1(shex , fn);
// 바이러스DB 파일과 매칭(matching)여부 판별
   fnum++;// 탐색 파일 수
   num+=t;// 감염 파일 수
}
```

표 2: 탐색 및 보고 모듈

두 번째 모듈은, 진단 및 삭제 모듈이다. 진단 및 삭제 모듈에서는 바이러스 패턴 DB를 로드(load)해서 탐색한 파일과 위의 탐색모듈에서 탐색

된 파일을 핵사코드로 변환한 것의 일치 여부를 판별한 후 바이러스 발견 유·무를 해당 파일의 경로명과 함께 화면에 출력한다. 그리고 바이러스 감염 파일을 자동 삭제한다

```
public int fRead1(String shex , String fn)
{
  String file = "virusdb.db";//바이러스 DB 파일
  ......
  while((s=is.readLine()) != null)
  {
     s1= s.indexOf("=");
    /*바이러스 DB 파일 로드(load)*/
     s2 = s.substring(s1+1);
    /*바이러스 발견 실패*/
   if(shex.indexOf(s2) == -1)continue;
    /*바이러스 발견 성공*/
  if(shex.indexOf(s2) != -1)
    /*바이러스 발견 출력*/
  System.out.println("  " + fn + " Virus Found!!");
  deleteFile(shex); //감염 파일 삭제
  }
   ...
```

표 3: 진단 및 삭제 모듈

3) 엔진의 동작 결과

엔진을 실행시키면, 제일 먼저 아래 그림2와 같은 화면이 나타나게 된다.

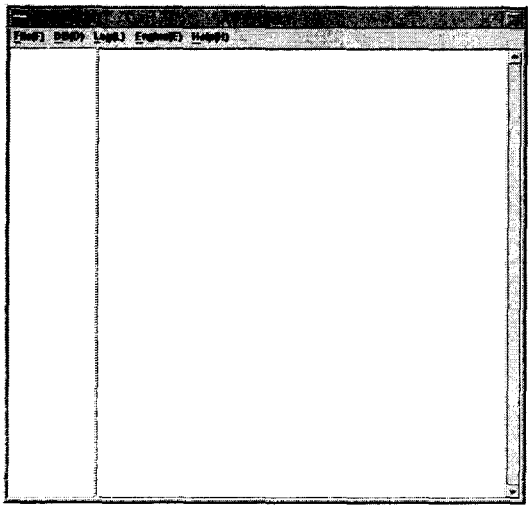

그림 2: 엔진 실행 첫 화면

File메뉴에서 Scatnstart를 선택하게 되면 왼쪽

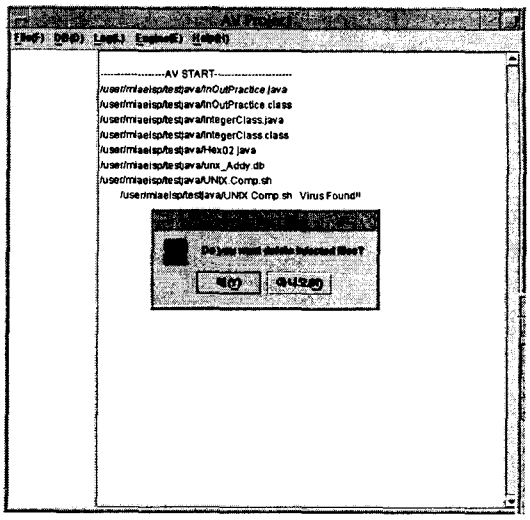

그림 3: 바이러스 발견 시

창에 파일이 탐색되는 과정이 보이면서, 탐색 영역의 파일과 바이러스 패턴 DB 파일의 패턴 매칭을 통해 바이러스 발견 시, 바이러스 삭제 여부를 묻는 창이 나타나게 된다. 그림3.

DB메뉴는 바이러스 패턴 DB 파일의 구성을 보여주고, 바이러스 패턴 DB 파일의 수정 및 갱신에 관한 모든 작업을 수행하게 된다. 그림4.는 바이러스 패턴 DB의 구성을 보여주는 view메뉴를 선택했을 시에 엔진에서 진단할 수 있는 바이러스명이 왼쪽 창에 나타나는 것을 보여준다.

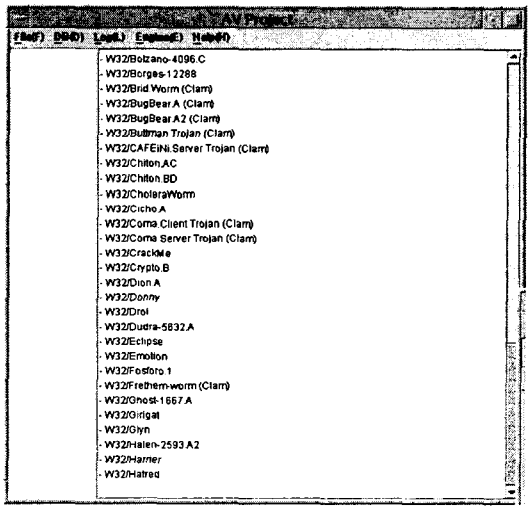

그림 4: 바이러스 패턴 DB

3. 효율적인 안티바이러스 엔진 구성 방법

1) 효율적인 매칭 알고리즘 적용

위에서 살펴본 ClamAV와 구현한 안티바이러스 엔진은 모두 패턴 매칭을 사용하는데, 바이러스 시그너처가 존재 할 경우 바이러스로 진단하게 된다.

일반적으로 패턴 매칭(pattern matching)은 문자열 중 지정된 서브 문자열을 찾는 것으로 서브 문자열이 있으면 참이 되고, 없으면 거짓이 되는 연산을 말한다. 이런 패턴 매칭 알고리즘으로는 Brute-Force, KMP (Knuth-Morris-Pratt), Boyer-Moore 알고리즘 등이 있다.

이 중, Brute-Force 알고리즘은, 길이가 n 인 텍스트T 에서 길이가 m인 패턴 P를 찾는 가장 간단한 방법으로, T의 모든 위치에서 패턴 P가 시작되는지 하나씩 맞춰보는 방법이다. 이 알고리즘의 경우 간단하여 이해하기 쉬운 반면, 경우에 따라서 패턴을 찾는 시간이 길어질 수 있는 단점이 있다. KMP알고리즘의 경우엔, 문자를 비교하는 과정에서 일치하지 않는 경우가 발생하면 다시 비교할 위치를 결정할 때 이미 비교한 패턴에 있는 정보를 활용하여 문서의 문자 위치를 나타내는 변수 i를 결정하는 것으로 반복되는 문자들이 많은 스트링에서 반복되는 문자들이 많은 패턴을 찾을 경우 유리하다. Boyer-Moore 알고리즘은, KMP 알고리즘과 유사한 방식이지만 패턴과 문서를 비교할 때 오른쪽에서 왼쪽으로 비교하는 것으로, 전처리 단계에서는 문장에 나타날 수 있는 각 패턴에 대해 불일치가 발생할 경우 SKIP정도를 결정하게 된다. 또한 Boyer-Moore 알고리즘은 세가지 알고리즘 중에서 가장 시간이 적게 걸리는 장점이 있다.[7],[8]

위에서 살펴본 바와 같이, 패턴 매칭의 방법을 사용할 때엔, 어떤 매칭 알고리즘을 사용하는 가에 따라 매칭 시간이 길어 질 수도 있고, 더 빨라질 수도 있다는 것을 알 수 있다.

본 논문에서 구현한 안티바이러스의 엔진에서 패턴 매칭을 하는 부분을 살펴보면, Java String Class에서 제공하는 indexOf();라는 메소드를 사용하고 있다는 것을 알 수 있다. 표3.

indexOf();라는 메소드는 문자열을 왼쪽부터 검색을 시작하며, 검색하는 문자나 문자열의 존재 여부를 판별하는 기능을 한다. 그런데 이 메소드이 내부 소스를 살펴보면, Brute-Force 알고리즘의 방법을 사용하고 있음을 알 수 있다. 표4.

```
public int indexOf(int ch, int fromIndex) {
        int max = offset + count;
        char v[] = value;
        if (fromIndex < 0) {
            fromIndex = 0;
        } else if (fromIndex >= count) {
            // Note: fromIndex might be near -1>>>1.
            return -1;
        }
        for (int i = offset + fromIndex ; i < max ; i++) {
            if (v[i] == ch) {
                    return i - offset;
            }
        }
        return -1;
}
```

표 4: Java String method indexOf();

따라서, 효율적인 안티바이러스 엔진의 구현을 위해서는 엔진의 속도 향상을 위해, 가장 매칭(matching)시간이 짧은, Boyer-Moore알고리즘을 적용하는 것이 바람직하다.

2) 효율적인 바이러스 패턴 DB 구성 적용

위에서 구현한 안티바이러스 엔진의 경우 ClamAV에서 사용하는 바이러스 패턴 DB 파일을 그대로 따르고 있다. 그러나, 바이러스 이름과 바이러스 시그너처만으로 구성된 바이러스 패턴 DB의 경우 바이러스 패턴 DB 파일에 삽입된 패턴들을 처음부터 끝까지 매칭 시키는 단순한 작업에만 쓰일 수 있고, 처음부터 끝까지 매칭 시킬 경우 시간이 많이 걸리는 단점이 있다. 따라서 좀 더 바이러스 패턴 DB를 효율적으로 구성하여 엔진의 성능을 향상시킬 수 있는 방법을 제안하겠다.

첫째, 바이러스 패턴 DB 파일에서 각 바이러스 패턴에 일정한 플래그(flag) 값을 주어 바이러스 패턴을 바이러스의 종류에 따라 구분한다. 예를 들어 웜(Worm)의 경우 바이러스 이름 앞에 "W"라는 값을 주고, 스크립트(Script) 바이러스의 경우 "S"라는 값을 준다면, 바이러스 탐색 모듈을 실행 할 시에, 전체 바이러스 패턴에 대해 다 매칭하지 않고, 실행 시에 옵션을 주어 특정 바이러스의 종류에 대해서만 탐색할 수 있도록 할 수 있다.

둘째, 최근에 발생한 바이러스의 경우 "N"이라는 값을 준다거나, 위험순위가 높은 바이러스에

대해서 우선 순위 값을 준다면, 모든 바이러스 패턴에 대해 다 매칭하지 않고, 현재 가장 위험하다고 판단되는 바이러스들에 대해서만 탐색할 수 있게 되므로, 부트 바이러스와 같이 현재 PC환경에서 발생하지 않는 바이러스와 같은 것에 대해서는 검색하지 않게 되고, 상대적으로 검색시간을 줄일 수 있게 된다.

또한 바이러스를 위험순위에 따라 분류해서 DB 파일을 구성할 경우 바이러스의 최신 동향에 동적으로 대응하는 성능을 높일 수 있게 된다.

III. 결론

컴퓨터 바이러스를 검색하기 위한 기본적인 안티바이러스 엔진 구현을 위해, 현재 공개된 안티바이러스 엔진인 ClamAV의 구성에 대해 알아보았고, ClamAV가 지원하는 바이러스 진단 방식인 패턴 매칭(pattern matching)과 ClamAV에서 사용하는 바이러스 패턴 DB 파일을 이용한 안티바이러스 엔진을 구현하였다.

기본 구성과, 단순히 바이러스를 탐지해서 삭제한다는 개념적인 면에서 구현 된 안티바이러스 엔진은 ClamAV와 차이점이 없어 보인다. 그러나 구현된 안티바이러스의 경우 향후 서버 단에서의 바이러스 탐지를 위해 UNIX환경에서 구현이 이루어졌으며, 다른 운영체제를 사용하는 서버들에 대한 이식성을 고려하여 플랫폼에 독립적인 Java 프로그래밍 언어를 사용하였고, 사용자 편의를 위해 GUI환경을 구축하였다.

구현된 안티바이러스 엔진은 위에서 제시한 좀 더 효율적인 매칭 알고리즘과 효율적인 바이러스 패턴 DB 파일을 적용하여 안티 바이러스 엔진의 성능을 향상 시켜야 한다.

또한, 안티바이러스의 주요 핵심은 진단 문자열을 이용하여 파일의 감염여부를 확인하는 것이다. 진단 문자열은 인터넷에 공개된 ClamAV의 바이러스 패턴 디피 파일에서 사용하는 바이러스 시그너처를 이용했지만, 바이러스 샘플을 축출하는 기술은 기존 백신 업체의 노하우이기 때문에 계속적인 분석을 통해 독자적인 컴퓨터 바이러스 패턴 DB 생성이 필요하다. 계속 증가하는 컴퓨터 바이러스에 대한 안티바이러스 엔진 최신 버전 및 바이러스 패턴 DB에 대한 On-Line Update 기능 및 인터페이스 보완이 이루어져야 할 것이다.

Acknowledgement

본 연구는 산업자원부에서 지원하는 공통핵심기술개발사업(과제번호: 10006464)으로 수행되었음.

참고문헌

[1] 최주영, "인터넷 서버용 병렬처리 안티바이러스 엔진 설계," 서울여자대학교 대학원 학위논문, 2003

[2] http://clamav.elektrapro.com/

[3] http://www.jabook.org/

[4] http://home.ahnlab.com/

[5] http://www.hauri.co.kr/

[6] http://www.trendmicro.com /en/home/global/enterprise.htm

[7] http://blue.skhu.ac.kr/%7Emckim/Lecture/DS /dna/index.html

[8] 황종선, 정영식, "C 언어로 설명한 알고리즘," 정익사, 2000

[9] Kai Hwang, *"Advanced computer architecture," Parallelism, Scalability, Programmability"* ,1993.

[10] Jeffrey O. Kephart and William C. Arnold, "Automatic Extraction of Computer Virus Signatures", Proc. 4th Int'l Virus Bulletin Conf., Virus Bulletin Ltd., Abingdon, England, 1994