파일 바이러스 복제 특성을 이용한 바이러스 탐지 및 복구[1)]

서용석[0] 이성욱 홍만표
아주대학교 정보통신 전문대학원
조시행
(주)안철수연구소
{cosm, wobbler, mphong}@ajou.ac.kr, shcho@ahnlab.com

Virus Detection and Recovery Using File Virus Self-Reproduction Characteristic

Yong-Seok Seo[0], Seung-Uck Lee, Man-Pyo Hong
Si-Haeng Cho
Graduate School Of Information and Communication, Ajou University
Ahnlab,Inc

요 약

본 논문에서는 컴퓨터 바이러스의 자기 복제 특성을 이용한 바이러스 탐지 및 복구 방안을 제안한다. 바이러스의 행동 패턴은 바이러스의 종류만큼 다양하지만 파일 바이러스의 경우, 자기 복제 행동 패턴은 대부분의 바이러스가 유사하다. 파일 바이러스가 시스템을 감염시키기 위해서는 기생할 실행파일을 열고, 자기 자신을 그 실행 파일에 복사해야 한다. 이와 같은 자기 복제 행위를 통해 바이러스가 광범위하게 전파될 때 그 피해도 커지게 된다. 바이러스의 자기 복제 특성을 감안하여 본 연구에서 제안하는 바이러스 탐지 알고리즘은 다음과 같은 특징을 가진다. 첫째, 바이러스의 자기 복제 행동 패턴을 파일 입출력 이벤트로 표현하여 바이러스의 행동 패턴으로 일반화시켰다. 둘째, 바이러스의 1차 감염 행위는 허용하고 2차 이후 감염 행위부터 탐지하고, 탐지되기 이전에 감염되었던 파일들을 복구한다. 이는 일반적인 바이러스들이 자기 복제를 지속적으로 수행한다는 점에 착안하여 false-positive 오류를 줄이기 위한 것이다. 본 고에서 제안하는 방법을 사용함으로써 특정 문자열에 의한 바이러스 탐지 및 복구 방법의 단점을 보안할 수 있을 것으로 기대된다.

1)

1. 서 론

컴퓨터의 사용이 증가할수록 컴퓨터에 대한 의존도가 높아지고 있다. 무수히 많은 일들이 컴퓨터를 통해 이루어지고 있는 현대 사회에서 컴퓨터 바이러스는 컴퓨터 시스템의 파괴뿐만 아니라 일의 진행을 마비시킨다. 컴퓨터 바이러스란 자기 자신을 스스로 복제하여 하나 혹은 여러 프로그램에 복사하는 코드를 가지고 있는 프로그램을 말한다[1]. 바이러스의 특성에 따라 숙주 프로그램 또는 시스템에 심각한 피해를 주는 것도 있다.

바이러스의 피해를 줄이고자 백신에 대한 많은 연구가 수행되었고, 이를 이용한 제품들이 제작되고 있다. 백신의 제작 기법은 크게 두 가지로 볼 수 있는데, 특정 문자열을 이용한 방법과 바이러스의 행동 패턴을 이용한 방법이 있다. 현재 대부분의 백신 제작 업체가 사용하고 있는 방법은 특정 문자열을 이용한 방법이다. 이는 새로운 바이러스가 출현할 때마다 그것을 분석하여 바이러스가 가지고 있는 특정 문자열을 추출하고 그 정보를 데이터베이스화하여 스캐닝 작업을 통해 바이러스의 존재를 탐지한다. 하지만 이 방법은 새로운 바이러스가 출현할 때마다 그 것을 분석해야 한다는 부담이 있다. 과거에 비해 새로운 바이러스가 출현하는 속도가 빠르고 바이러스 발생률(Birth rate)[2], 즉 바이러스가 다른 프로그램으로 복제하는 속도가 인터넷의 보편화로 매우 빨라졌다. 따라서 백신 제작에 있어 많은 양과 빠른 전파 속도를 고려해야 한다. 이 점에 있어 특정 문자열을 이용한 대응은 한계에 부딪힐 것이다.

이와는 다르게 바이러스의 행동 패턴을 이용한 백신이 연구되고 있다. 대표적인 것이 시만텍(http://www.symantec.com)의 Bloodhound[3]이다. 이 Bloodhound 시스템은 두 단계로 이루어져 있다. 첫 번째 단계는 휴리스틱 알고리즘을 사용하여 파일을 스캐닝하고 바이러스라 의심되는 파일의 리스트를 만드는 것이다. 두 번째 단계에서는 첫 번째 단계에서 탐지된 의심되는 파일들을 가상머신에서 실행시켜 바이러스 진위 여부를 가려낸다. 이 방법은 두 단계 모두 휴리스틱 알고리즘을 사용하는데, 특히 두 번째 단계는 가상머신에서 에뮬레이션 작업을 수행하는 것으로 그 실행 속도가 상당히 느려 각 사용자의 PC에서는 동작하기 어렵다는 단점이 있다.

본 논문에서는 특정 문자열을 이용한 백신 제작의 단점을 해결하기 위해 바이러스의 행동 패턴을 이용한 방안을 연구하였다. 이를 위해 바이러스의 복제 행위로 바이러스의 행동 패턴을 일반화했고 이것을 이용하여 바이러스의 탐지 및 복구에 관한 방법을 제시한다. 또한 제안한 방법은 많은 계산을 필요로 하지 않기 때문에 일반 PC에서도 수행 가능하다.

본 고의 2장에서는 서론에서 지적한 백신제작에 대한 단점을 보안하기 위해 제안된 모델의 개략적인 설명을 한다. 3장에서는 제안한 모델의 각 모듈에 대해 세부적인 사항을 설명한다. 4장에서는 본 연구에서 다루지는 않지만, 램 상주 바이러스에 대한 해결 방안을 설명한다.

2. 제안모델

2.1 기본개념

바이러스 행위의 특징으로 크게 두 가지를 들 수 있다. 시스템 파괴와 자기 자신의 복제를 이용한 감염 행위이다.

시스템 파괴의 경우 파일 삭제 및 변조, 파일 시스템의 파괴 등을 예로 들 수 있다. 하지만 이 행위는 파괴 행위를 하는 방법이 다양하기 때문에 바이러스의 행위 패턴으로 일반화하기 어렵다.

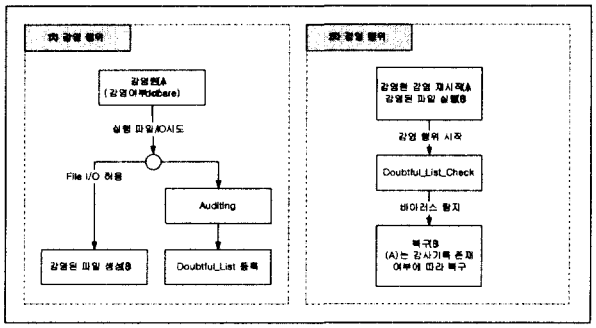
반면에 자기 자신의 복제를 통한 감염행위는 파일 입출력 레벨에서 생각한다면 그 행위의 일반화가 가능하다. 파일에 기생하여 활동하는 파일 바이러스의 경우 대부분의 바이러스가 자기 자신을 복사할 때, 실행 파일을 읽어 그 파일의 일정 영역에 자기 자신을 복사하는 행위를 한다. 이를 파일 입출력 레벨에서 생각한다면 "실행파일 읽기 - 실행파일에 쓰기"로 일반화 할 수 있다. 하지만 이 경우를 모두 바이러스라 판정하는 것은 높은 false-positive 오류를 일으킨다. 이 false-positive 오류를 줄이기 위해 다음과 같은 가정을 하였다.

1) 본 논문은 (주)안철수연구소의 연구비 지원 및 교육부 BK사업 지원에 의하여 연구되었음.

하나의 파일 혹은 한 시스템에 감염된 바이러스가 번식 행위를 하지 않고, 현재 숙주 시스템만을 파괴한다면 피해 범위는 크지 않을 것이다. 즉 번식 행위가 왕성한 바이러스의 경우 여러 파일 또는 여러 시스템을 감염시키므로 피해 범위가 커질 것이다. 전자의 경우는 거시적인 관점에서 볼 때 바이러스의 수는 점점 줄어들어 어느 순간 번식되지 않기 때문에 바이러스의 탐지 및 치료에 고려 대상이 되지 않는다. 주목해야할 것은 번식 행위를 하는 바이러스인 것이다. 번식 행위를 수행한다는 것은 자기 자신 복제 행위를 수행한다는 의미와 같다. 따라서 위에서 일반화한 바이러스의 행위 패턴인 "실행파일 읽기 - 실행파일에 쓰기"가 여러 번 발생하게 될 것이다.

본 연구를 통해 조사한 바에 의하면, 일반적인 PC 시스템에서 정상적인 하나의 프로세스가 실행파일을 읽고, 실행파일에 쓰는 행위를 하지 않았다. 따라서 위의 일반화에서 우려되던 false-positive 오류는 상당히 낮은 수치로 떨어진다.

본 연구에서 제안하는 모델의 개념을 그림 1에 나타내었다.

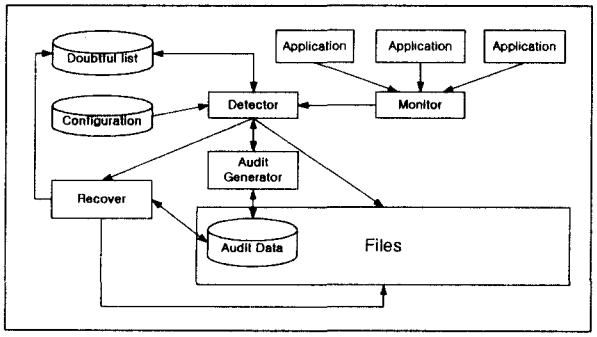


[그림 1] 모델 개념도

위의 그림처럼 바이러스의 1차 감염 행위는 허용을 하며 감사 데이터만 기록한다. 또한 바이러스라 의심되는 행위를 수행하였기 때문에 Doubtful_List에 추가하여, 감염원이나 피 감염원이 2차 감염 행위를 수행하는지 감시한다. 감염원이나 피 감염원이 2차 감염 행위를 수행할 경우, 실행파일의 입출력 패턴이 1차 때와 동일할 것이다. 이 경우를 바이러스의 감염 행위로 간주할 수 있다. 이 감염행위의 회수를 임계값으로 사용할 경우, false-positive 오류를 줄일 수 있다.

2.2 시스템 구조

제안하는 시스템의 구조는 그림 2와 같으며, 탐지기, 모니터, 복구기, 감사기록 생성기로 구성되어 있다. 사용되는 데이터베이스는 Doubtful_list, Configuration, Audit data가 있다. 각 모듈 및 데이터베이스에 대한 상세한 설명은 3.시스템 모듈에서 다룰 것이다.



[그림 2] 시스템 구조

2.3 알고리즘

2.3.1 바이러스 탐지 알고리즘

바이러스 탐지는 한 프로세스가 바이러스라 의심되는 행위를 여러 번 수행하였을 때 탐지한다. 즉 사용자가 정의한 임계 회수를 초과할 경우 바이러스라 확정한다.

바이러스 탐지에 대한 알고리즘을 그림 3에 나타내었다.

```
/* File I/O Event가 Monitor로부터 hooking 되어     */
/* 이 모듈로 전달된다                              */
...
if ( Event가 Configuration에 위배 ) {
    if ( Event를 발생시킨 프로세스가 Doubtful_list에 존재 ) {
        if ( Configuration 위배 회수 >= 복제 회수 임계값 ) {
            Detect Virus;
        }
        else {
            Configuration 위배 회수 ++;
        }
    }
    else {
        Doubtful_list에 추가;
        Configuration 위배 회수 = 1;
    }
    Record Audit data;
}
...
```

[그림 3] 바이러스 탐지 알고리즘

2.3.2 바이러스 복구 알고리즘

바이러스 복구 알고리즘은 감염원에 의해 감염된 모든 피 감염원을 복구하기 위해 재귀함수를 이용한다. 복구에 관한 알고리즘을 그림 4에 나타내었다.

```
/* 실시간 혹은 배치작업으로 복구를 원할 경우 이 모듈을 실행 */
/* 파라미터로 감염을 시키는 프로세스 이름과 감염을 당한     */
/* 프로세스 이름이 넘어온다.                                 */
Recover( 감염원, 피 감염원)
{
    A <- 감염원;
    B <- 피 감염원;
    while ( Audit data에 감염원 A 존재 ) {
        Recover( B, Audit data에 기록된 피 감염원);

        Audit data에 기록되어 있는 피 감염원 검색;
        Audit data에 기록되어 있는 WRITE action을 시간의
        역순으로 피 감염원 파일에 WRITE하여 피 감염원 복구;
        Delete this audit data;
        if ( -- (A의 Configuration 위배 회수) == 0)
            A를 Doubtful_list에서 삭제
    }
    return;
}
```

[그림 4] 바이러스 복구 알고리즘

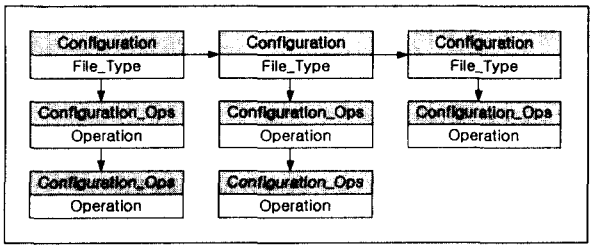
3. 시스템 모듈

3.1 Configuration

Configuration은 Doubtful_list를 생성하는 기준을 제시한다. 파일 바이러스의 복제 행위 패턴을 일반화한 것으로써 false-positive 오류가 상당히 클 것으로 예상되나 현재까지 조사한 바에 의하면 일반 PC에서 정상적인 프로세스가 이와 같은 행위를 수행하는 것은 없었다. 이 구조를 나타내면 그림 5와 같다.

각 Configuration은 파일 타입과 그 파일 타입에 바이러스가 보낼 가능성이 있는 파일 입출력 이벤트를 가지고 있다. 예를 들면, File_Type이 'EXE' 이고, Operation이 'WRITE'라고 하면, 'EXE' 확장자를 가진 파일에 'WRITE' 이벤트를 보내면 바이러스라 의심하여 추

후 대응을 하겠다는 것을 의미한다.



[그림 5] Configuration Data Structure

3.2 Doubtful list

Doubtful list는 3.1 Configuration에 해당되는 이벤트가 발생했을 경우, 이를 바이러스라 의심하고 그 리스트를 나타내는데 사용한다. 이 리스트의 각 노드는 {Process Name, State, Count, pNext}의 데이터를 가지고 있다.

Process Name은 이벤트를 발생시킨 프로세스의 이름을 기록한다.

State는 Process Name에 해당하는 프로세스의 상태를 나타내는데, 값은 '바이러스라 의심' 또는 '바이러스로 확정'으로 나타낸다.

Count는 Process Name에 해당하는 프로세스가 바이러스라 의심되는 행위를 수행한 회수를 나타낸다. 이 값은 바이러스로 확정하기 위해 설정한 임계값과 비교를 위해 사용하며, 이 값이 임계 회수를 초과하면 바이러스라 확정한다.

pNext는 다음 노드를 가리킨다.

3.3 Detector

이 모듈은 시스템의 중심이 되는 것으로서, 바이러스인지 판단하고, 필요한 경우 audit data를 기록하고, 복구를 수행하는 역할을 한다. 이것의 알고리즘은 그림 3과 같다.

3.4 Monitor

Monitor는 한 시스템에서 파일 시스템으로 보내는 모든 파일 입출력 이벤트를 가로채는 역할을 한다. 이 곳에서 가로채는 이벤트를 이용하여 바이러스를 탐지하거나 감사자료를 기록하는데 사용해야 하기 때문에 최소한 다음의 데이터를 얻어야 한다.

```
struct Monitor_data {
    char m_szTime[128];      // 이벤트 발생 시간
    char m_szProcess[128];   // 이벤트 발생시킨 프로세스 이름
    char m_szRequest[128];   // 이벤트 종류
    char m_szFileName[128];  // I/O가 수행될 파일 이름
    char m_szFilePath[512];  // I/O가 수행될 파일 경로
    int  m_nOffset;          // File I/O offset
    int  m_nLength;          // File I/O length
};
```

[그림 6] Monitoring Data

3.5 Recovery

복구는 실시간으로 진행하거나 배치업무로 진행할 수 있다. 필요한 시점에서 복구를 호출하여 복구 작업을 수행하며, 그림 4에 나타낸 알고리즘에 의해 수행된다.

3.6 Audit Generator

3.3 Detector에서 현재 실행중인 프로세스가 바이러스라 의심되는 경우 감사자료를 기록하며, 이 때 실행되는 모듈이다. 프로세스가 파일 입출력을 시도할 때, 이 보다 먼저 백업 작업을 수행한다. 예를 들면, 프로세스가 한 파일의 특정 오프셋 위치에, 길이 n의 데이터를 쓰려고 할 때, 이 작업보다 먼저 Audit Generator가 그 위치에 있는 데이터를 읽어 보관한다. 기록하는 데이터는 다음과 같다. (1)이벤트 발생 시간 (2)이벤트를 발생시킨 프로세스 (3)이벤트 종류 (4)I/O가 수행될 파일 경로 (5)I/O가 수행될 파일 이름 (6)파일 오프셋 (7)데이터 길이 (8)원본 데이터.

4. 램 상주 바이러스

램 상주된 바이러스는 감염원을 알 수 없기 때문에 감사자료를 기록하는 것이 불가능하게 된다. 본 연구에서 다루지는 않았지만, 이 것에 관한 대응은 다음과 같이 할 수 있다.

```
Real Mode 시작할 때 메모리 크기 기록;
Doubtful_File <- NULL;
while (Real Mode) {
    Application 시작;
    Application 실행 {
        Detector에 의해 바이러스라 의심되는 경우 탐지되면,
        감염원을 Doubtful_File로 기록
    }
    Application 종료 {
        if (메모리 크기 감소 | 인터럽트 벡터 테이블 변화) {
            Doubtful_File <- Application Name;
        }
    }
}
```

[그림 7] 램 상주 바이러스 처리 방안

램 상주는 리얼모드(Real mode)에서 인터럽트 벡터 테이블을 변형하는 것이다. 리얼모드에서 보호모드(Protect mode)로 전환되면 상주되어 있던 것은 없어진다. 보호모드에서 리얼모드로 변환될 때, 남아있는 메모리의 양을 기록한다. 그 후 리얼모드에서 일반 어플리케이션 프로그램이 종료될 때, 메모리의 변화나 인터럽트 벡터 테이블의 변화 유무를 검사하여 어느 프로세스가 램 상주를 일으켰는지 체크할 수 있다.

5. 결론

본 연구는 특정 문자열을 이용한 바이러스 탐지 및 복구 방법의 단점을 보완하기 위해 바이러스의 자기 복제 행위를 파일 입출력 이벤트로 표현하여 바이러스의 행위 패턴으로 삼아 바이러스를 탐지한다.

도스용 파일 바이러스의 경우 다른 파일을 감염시키기 위해 실행파일(.EXE, .COM)을 열어 자기 자신을 실행파일에 복제한다. 따라서 제안한 방법을 이용하면 바이러스의 복제행위를 쉽게 탐지할 수 있으며, 현재 특정 문자열을 이용한 방법의 최대 단점인 바이러스를 분석해야 한다는 점이 개선될 것이다.

본 논문에서는 파일 입출력 이벤트를 감시하여 바이러스의 1차 감염을 허용하고, 2차 이후의 감염 행위부터는 불허한다는 방법론 사용했다. 이를 이용한다면, 현재 바이러스의 큰 비중을 차지하고 있는 전자우편을 통한 인터넷 웜의 탐지 및 치료 방안에도 이용될 수 있을 것이다.

6. 참고문헌

[1] Eugene H. Spafford, Computer Viruses as Artificial Life, Journal of Artificial Life, 1994.

[2] Jeffrey O. Kephart, David M. Chess, Steve R. White, Computers and Epidemiology, IEEE SPECTRUM, 1993 .

[3] Understanding Heuristics: Symantec's Bloodhound Technology, Symantec White Paper Series Volume XXXIV.

[4] http://www.sysinternals.com/