언어 압축 알고리즘을 이용한 컴퓨터 바이러스의 행위 패턴 추출

임영환°, 위규범
아주대학교 정보통신대학원 정보통신공학과

Extraction of Computer Virus Behavior by Using Language Compression Algorithm

Younghwan Lim, Kyubum Wee
Graduate School of Information and Communication Technology, Ajou University

요 약

컴퓨터 사용증가와 함께 컴퓨터 바이러스 또한 증가하고 있다. 바이러스 검사 프로그램은 바이러스의 특정 문자열(signature)을 찾아내는 문자열 검색도구와 프로세스의 행동을 모니터링 하는 감시 도구(general purpose monitor)의 두 가지 형태가 있으며, 각각은 미 발견 바이러스에 대한 취약성과 시스템 오버헤드를 단점으로 가지고 있다. 또한, 최근에 제안된 면역 시스템은 계산 복잡도나 시스템 구성면에서 지나친 부담을 가지고 있다. 본 논문에서는 바이러스들의 행위를 추출할 수 있도록 하기 위하여, 언어 압축 알고리즘을 이용하여 바이러스 행동 패턴을 추출하는 방법을 고안하였고, 몇 가지 바이러스를 이용하여 실험해 보았다. 그 결과 실제 학습에 이용한 바이러스가 아니더라도 유사한 동작을 하는 바이러스에 대해서는 면역성을 가질 수 있었다.

1. 서론

과거에 비해 컴퓨터의 대중화와 인터넷 등의 네트워의 일반적인 사용과 함께 컴퓨터 바이러스의 확산도 역시 더욱 고속화되고, 광역화되어져 왔다. 컴퓨터 바이러스를 막기 위해, 일반 사용자들의 바이러스에 대한 인식을 바꾸고, 각종 안티-바이러스 소프트웨어들을 배포하는 등의 정부 전산 관련 기관들과 안티-바이러스 소프트웨어 제작자들의 노력에도 불구하고 컴퓨터 바이러스는 좀처럼 줄어들 기미를 보이지 않고, 오히려 더욱 다양한 형태의 컴퓨터 바이러스가 출현하여 큰 사회적인 문제를 야기하기까지 한다. 이에 따라, 컴퓨터 바이러스 연구가들은 바이러스에 대한 빠른 대응책을 위해 노력함과 동시에 컴퓨터 바이러스에 대응하는 다양한 기술들을 자동화하려는 노력들이 수행되어지고 있다[1].

또한, 이러한 자동화 노력들과 함께 주목을 받고 있는 부분이 아직 안티-바이러스 소프트웨어 개발자들에 의해 분석되기 이전에도 바이러스로 짐작되는 프로그램을 인식할 수 있는 미 발견 바이러스(known virus)에 대한 대응책도 끊임없이 연구되고 있다. 때문에, 기존 바이러스 연구가들과 함께 대학이나 연구소 등에서도 선천적인 바이러스에 대한 감지 능력을 가지는 컴퓨터 면역시스템이라는 분야에 대해 새롭게 연구하고 있다[2,3].

본 논문에서는 이미 알려진 바이러스들의 정보로부터 알려지지 않은 바이러스를 검진할 수 있도록 하기 위해서, 컴퓨터 바이러스의 프로그램 코드로부터 언어 압축 알고리즘을 이용하여 바이러스의 일반적인 패턴을 추출하고, 바이러스를 진단 할 수 있는 방법을 제안하였다.

2. 관련 연구

2.1 기존 컴퓨터 바이러스 방지 프로그램

현재 일반적으로 바이러스에 대한 대응책으로 사용되고 있는 도구들은 여러 가지가 있으며, 다음의 몇 가지로 구분될 수 있다. 시그네쳐 검색(signature scanning) 도구는 컴퓨터 바이러스 각각이 가지는 특정 문자열과 그 문자열의 위치를 데이터베이스 화 하여 파일의 존재여부를 검색하는 도구이며, 컴퓨터 바이러스의 identification과 그에 따른 치료방법을 적용하는 데 유리하지만, 그에 반하여, 미리 분석된 컴퓨터 바이러스에 대해서만 바이러스를 검진할 수 있다는 단점을 가지고 있다. 감시 도구(general purpose monitor)는 프로세스의 행동을 감시하면서 비정상적인 동작을 감지해내는 도구이며, false positive가 높고 시스템 오버헤드가 높은 것이 문제점이다. 또 다른 도구로는 보호하고자 하는 파일의 체크 섬을 이용하여 검사하거나 접근 제어를 통해 컴퓨터 바이러스로부터의 공격을 막는 방법 등이 사용되고 있으나, 어느 도구들도 모두 장단점을 가지고 있고, 필요에 따라서 두 가지 이상의 도구를 통합하여 사용하기도 한다[4].

이중에서 노턴 안티 바이러스나 국내의 V3등의 안티 바이러스 소프트웨어들이 주로 이용하고 있는 방법이 바로 문자열 검색 방식이며, false positive가 낮고, 시스템 오버헤드가 적기 때문에 일반으로 배포되는 소프트웨어의 방식으로는 가장 적합하다고 할 수 있다. 그러나, 앞에서도 말한 바와 같이 문자열 검색도구는 컴퓨터 바이러스를 검진하기 위해 이미 대상 바이러스의 특성 분석이 선행되고, 그 결과가 소프트웨어를 사용하는 개별 사용자에게 배포되어야 하기 때문에 지금과 같이 급속도

로 바이러스가 퍼져나가는 것을 막는데 점점 한계에 부딪히고 있다. 때문에, 이와 같은 소프트웨어에 모니터링 기능이 첨가되고 있고, 미 발견 바이러스에 대해서도 일부 검진 가능한 방법들을 사용하고는 있다. 하지만, 그다지 괄목할만한 성과는 거두고 있지 못하다.

2.2 컴퓨터 면역 시스템

컴퓨터 면역 시스템은 시스템이 컴퓨터 바이러스에 대한 면역성을 가지게 만드는 프로그램으로, 뉴멕시코 대학의 연구팀에 의해 제시된 면역 시스템과 IBM 연구소에서 개발한 디지털 면역 시스템(Digital Immune System)이 있다.

뉴멕시코 대학의 면역 시스템은 인체 면역 시스템의 이론으로서 가장 유력한 Clonal Selection의 이론을 도입하여, 보호하고자 하는 대상으로부터 랜덤하게 악성 코드를 생성하는 것을 통해 바이러스를 진단한다. 그러나, 이 시스템은 지나친 계산 복잡도를 가지고 있는 것이 단점이다[5].

IBM 연구소의 디지털 면역 시스템은 사용자 컴퓨터에 의해 바이러스로 의심이 되는 대상을 관리 PC로 전송하여 바이러스 행동을 분석하고 진단 문자열을 뽑아 내는 과정을 자동화하여, 다시 각각의 호스트로 분배하는 시스템이다. 그러나, 이 시스템은 관리 PC의 심한 시스템 부하가 생기며, 최초에 사용자 컴퓨터에서 알려지지 않은 바이러스를 검진하는 대책이 다소 부족하다[3].

3. 제안된 방법

[그림1] 바이러스 검사를 위한 3단계 과정

제안하고 있는 방법에서는 컴퓨터 바이러스의 일반적인 패턴을 얻기 위해 우리는 언어를 압축하는 알고리즘을 이용한다[6]. 이 알고리즘을 이용하여 컴퓨터 바이러스를 검진하기 위해 문자열 변환, 엔진 생성, 검사의 세 가지 단계를 거치게 된다.

3.1 문자열 변환

제안하고 있는 방식에서는 언어를 압축하는 방법을 사용하고 있다. 그러므로, 바이러스의 프로그램 코드를 조작 가능한 문자열의 형태를 가지도록 변환하는 과정이 요구된다.

문자열 변환과정에서 가장 먼저 해야 할 일은 프로그램의 코드를 의미를 가지는 명령어의 셋으로 바꾸는 일이다. 바이러스의 행동 패턴을 추출 하고자 하는 것이기 때문에, 나름대로 의미를 가지는 단위 별로 알파벳을 매핑하는 것이 유리하기 때문이다. 그 다음, 코드 내에서 패턴 분석에 별다른 의미를 가지지 않는 명령어들을 제거하고, 여러 명령어가 합쳐져서 하나의 의미를 가지는 명령어들은 하나의 그룹으로 지정하고, 최종적으로 이들을 미리 정의된 알파벳으로 변환하여, 하나의 프로그램 코드가 하나의 개별 스트링으로 변환될 수 있도록 해야 한다.

[그림2] 문자열 변환 과정

3.2 엔진 생성

엔진은 컴퓨터 바이러스 코드를 언어 압축 알고리즘을 이용해 하나의 언어로서 형성된 DFA(Deterministic Finite Automaton)를 일컫는다. 바로 이 부분에서 제안하고 있는 언어 압축알고리즘을 이용하게 된다.

이 알고리즘은 MDL(Minimum Description Length)이라는 개념을 이용한다. DL(Description Length)은 데이터 인코딩 길이(data-encoding length, Δ)와 그래마 인코딩 길이(grammar encoding length, Γ)의 두 값의 합으로 결정되며, m이 인코딩되는 문장의 수, $|S_i|$는 i번째 문자열 S_i의 길이, Z_{ij}는 문장 S_i의 j번째 심볼에 도달한 상태를 떠나는 방법의 수, $|\delta|$는 δ에 속하는 triple의 수, |Q|는 상태의 수, $|\Sigma|$는 알파벳의 크기, 그리고, |F|가 최종 상태의 수일 때, 두 값 Δ와 Γ는 다음과 같이 정의한다[6].

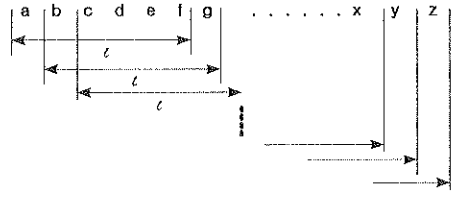
$$\Delta = \sum_{i=1}^{m}\sum_{j=1}^{|S_i|}\log_2 z_{ij}$$

$$\Gamma = |\delta|[2(\log_2|Q|)+\log_2|\Sigma|]+|F|[\log_2|Q|]$$

주어진 문자열의 집합으로부터 MDL을 가지는 DFA를 구성하는 알고리즘은 다음과 같다[6].

1단계. 입력 문장을 생성한다.
2단계. 입력 문장의 집합을 인식하는 DFA를 생성한다.
3단계. DFA에서 2개의 상태를 합쳐서 1개의 상태로 만들더라도 NFA가 되지 않고 DFA로 남아있도록 하는 것 중에서 DL을 최소로 하는 2개의 상태를 합쳐서 1개의 상태로 만든다.
4단계. 3단계를 반복한다. 3 단계의 조건을 만족하는 2개의 상태가 존재하지 않으면 종료한다. (hill-climbing)

처음에 알파벳으로 변환된 바이러스의 시퀀스를 [그림 3]과 같이 일정 길이 ℓ로 겹쳐진 형태로 분할하고, 분할된 각각의 문자열을 최초 입력 데이터로 사용한다.



[그림3] 문자열의 분할

3.3 검사

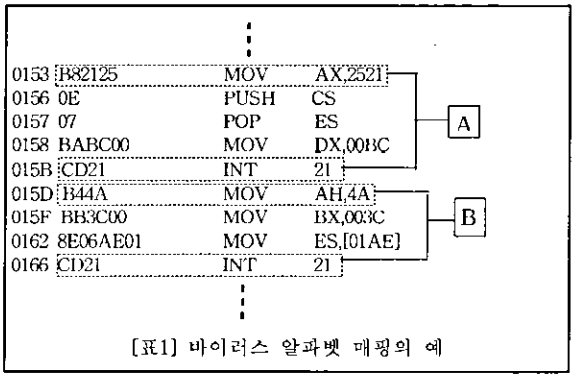
검사는 엔진으로 만들어진 DFA에 입력으로 들어오는 코드를 대입함으로써 이루어진다. 입력 코드는 엔진생성 전에 행해진 문자열 변환과 비슷한 형태의 알파벳 매핑 과정을 거치게 되지만, 문자열 변환과정에서는 전문가에 의해 분석된 결과를 통해 어느 정도 정확하게 컴퓨터 바이러스의 행동을 추출한 길과인 반면, 검사 과정에서는 심한 오버헤드를 막기 위해 단순한 형태의 필터링과 매핑 과정만을 거치게 된다.

이렇게 코드에서 변환된 입력 문자열을 입력으로 주었을 때, DFA의 최종 상태까지 인식하게되면, 이 코드를 컴퓨터 바이러스로 간주하게 된다.

이 과정에서 컴퓨터 바이러스는 실제 프로그램보다 먼저 실행 되야 하므로 프로그램에서 처음 시작위치를 알아내어 그 위치로부터 일정길이 만큼만 검사하는 것을 통해 파일 전체를 검사하는 부담을 줄일 수 있다.

4. 실험 및 분석

위의 이론을 실험하기 위해, 17종의 골도스용 바이러스를 사용하였다. 스트링 변환은 [표1]과 같이 바이러스에서 주로 사용하는 도스 펑션 콜의 종류를 각각의 알파벳으로 변환하여 실험하였다[7].



[표1] 바이러스 알파벳 매핑의 예

위에 제시된 17종의 바이러스 중, 10종의 바이러스를 학습에 이용하였고, 4종은 학습에 이용된 바이러스와 유사한 바이러스이며, 나머지 3종은 학습에 사용된 바이러스와는 상이하게 다른 형태의 바이러스이다.

바이러스 코드를 분할 할 때의 길이 ι을 정하는 특별한 알고리즘이 존재하지 않으므로, 학습에 이용된 바이러스들로부터 변환된 문자열들 중에서 가장 짧은 문자열의 길이를 ι로 설정하였다.

이때 학습에 이용된 바이러스에 대해서는 100% 진단율을 보였고, 유사 바이러스 중 3종은 학습에 이용된 바이러스와 마찬가지로 진단이 되었으나 나머지 하나의 유사 바이러스와 상이하게 다른 바이러스는 진단하지 못했다.

유사 바이러스 중 1종이 진단되지 못한 것은 비록 학습에 이용된 바이러스와 비슷한 패턴을 가지며 행동 양식이 비슷하기는 하지만, JMP 명령어 등에 의해 실질적으로 코드에 배열된 순서와 실행시의 순서와는 다르기 때문이다. 실제 분석에서 그 바이러스가 행동 패턴이 유사하다라고 분석되었다고 하더라도 코드로부터 얻은 정보는 실제 패턴과는 차이를 보일 수 있다.

또한, 실험 편의를 위해 단순히 도스 펑션 콜만으로 알파벳을 정의하였고, 충분한 학습 데이터가 주어진 것이 아니기 때문에, 다양한 바이러스의 행동 패턴을 일반화하는 데는 다소 부족하다.

5. 결론 및 향후계획

언어 압축 알고리즘은 바이러스 코드의 패턴을 정의하기 위해 어느 정도의 효과를 보여주었다. 다만, 패턴이 상이하게 다른 경우 이 바이러스를 검진하지 못하였고, 유사 패턴이더라도 순서나 길이 등을 어느 정도 이상의 변형이 가해진 경우에도 검진하는 데 어려움이 있었다. 더구나, 암호화 바이러스 등의 본래 코드를 엄폐하는 경우에는 바이러스를 진단하는 것이 불가능하다.

이와 같은 문제를 극복하기 위해 바이러스 패턴을 정의하는 언어압축 알고리즘에 휴리스틱 정보를 첨가하여 변형 종에 대한 대응책을 강화하고, 단순히 파일 형태의 프로그램 코드로부터 정보를 추출하기보다는 프로그램이 실행되면서 행해지는 시스템 콜을 모니터링 한 결과로부터 바이러스의 패턴을 정의하고 진단하면 바이러스가 자신을 엄폐하더라도 이를 검진할 수 있게 하며, 또한, 디스크 전체를 스캐닝 하는 것에 대한 부담을 없앨 수 있을 것이다.

또한, 제안된 방식을 발전시켜 바이러스에만 국한시키지 않고 트로이 목마나, 해킹 등의 악성 코드와 공격 전체에 대한 패턴을 정의할 수 있도록 하여 IDS(Intrusion Detection System)에 응용될 수도 있을 것이다.

6. 참고문헌

[1] S. R. White, "Open Problems in Computer Virus Research," Virus Bulletin Conference, Munich, Germany, Oct. 1998.

[2] S. Forrest, "Self-Nonself Discrimination in a Computer," Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, pp. 202-212, 1994.

[3] J. O. Kephart, "Blueprint for a Computer Immune System," Virus Bulletin International Conference, San Francisco, Oct. 1997.

[4] W. T. Polk, "A Guide to the Selection of Anti-Virus Tools and Techniques," pp. 11-16, National Institute of Standards and Technology Computer Security Division, Dec. 1992.

[5] S. Forrest, "An Immunological Approach to Change Detection Algorithms, Analysis and Implications," Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy, pp. 110-119, 1996.

[6] T. K. Teal and C. E. Taylor, "Effect of Compression on Language Evolution," Artificial Life 6(2) pp. 129-144, 2000.

[7] R. Irvine, "Assembly Language for Intel-Based Computers," pp. 151-175, Prentice Hall, 2000.