사용자와 객체의 관계를 고려한 악성객체 방지 기법

이희대[0], 임영환, 예홍진
아주대학교 정보통신전문대학원
eehd0602@shinbiro.com[0], yhlim@netsgo.com, hjyeh@ajou.ac.kr

Malicious Object Prevention Technique Through Relation Between User And Object

HeeDae Lee[0] YoungHwan Lim HongJin Yeh
Graduate School of information and communication, Ajou University

요 약

악성 코드를 포함하고 있는 악성 오브젝트(또는 프로그램)들을 막기 위한 방법으로서 오브젝트의 정상행위를 정의해 놓고 정상행위에서 벗어나면 악성으로 판단하는 Sandbox, MAPBox등이 있다. 이러한 방법들 모두 오브젝트의 정상행위가 무엇인지를 정의하고, 정상행위에 맞는 자원을 할당하는데 초점을 맞추고 있다. 하지만, 정상행위로 정의된 행위의 범위 안에서도 악성행위를 할 수 있다. 본 논문에서는 정상행위의 범위 안에서의 발생할 수 있는 악성행위를 막기 위한 방법으로서 사용자와 오브젝트와의 관계를 고려한 강화된 악성 객체 방지 기법에 대하여 제시하였다.

1. 서 론

네트워크가 점자 거대해지면서 많은 사용자들이 인터넷을 통하여 많은 데이터들을 주고받고, 마이크로소프트의 익스플로러나 네스케이프의 내비게이터, 또는 ftp등과 같은 어플리케이션을 통하여 인터넷으로부터 실행 가능한 오브젝트(또는 프로그램)를 다운로드 한다.

사용자들은 이렇게 다운로드한 실행 가능한 오브젝트를 실행하거나, 시스템에 설치하여 사용하게 된다. 하지만, 이러한 실행 가능한 오브젝트들은 보안적 관점에서는 반드시 안전한 것이라고 할 수 없다. 인터넷에는 많은 실행 가능한 오브젝트들이 있다. 그리고 이러한 실행 가능한 오브젝트들 중에는 악의를 가진 누군가에 의해 만들어진, 바이러스, 백도어, Trojan horse와 같은 악성 코드를 포함한 것들도 있을 수 있기 때문에 반드시 안전하다고는 볼 수 없다.

악성코드를 포함한 악성 오브젝트를 막기 위한 방법으로서 다음과 같은 몇 가지 방법이 제안되고 있다.

첫 번째로, 가장 많은 연구가 이루어지고 있는 것으로서, 정의된 정상행위를 가지고 프로그램을 실행하다 권한 이상의 것을 수행하려고 하면, 이것을 악성이라고 판단하는 샌드박스(Sandbox)가 있다. [1]

두 번째, 기존의 단일 샌드박스 시스템이 가지는 단점인 낮은 적용 성을 보완한 것으로서, 프로그램을 에디터, 컴파일러, 게임등의 클래스로 나누고 각 클래스는 해당 정상행위만을 가지고, 정상 행위에 벗어나면 악성이라고 판단하는 맵박스(MAPbox)가 있다.[2,3]

마지막으로, 각 오브젝트 마다 제한된 권한을 가진 sub-user ID를 할당하고, 이렇게 할당된 sub-user ID가 가진 권한에 따라서 오브젝트가 자원을 접근할 수 있는 Secure Web Browser가 있다.[4,5]

이러한 방법들 모두 오브젝트의 권한을 제한하는 방법이다. 즉, 사용자가 가지고 있는 권한을 모두 오브젝트가 가질 필요는 없다는 것이다. 각 오브젝트의 특성에 따라서 제한된 권한을 줌으로서, 혹시라도 발생 할 수 있을 문제를 막겠다는 것이다.

하지만, 앞에서 말한 방법들은 각 오브젝트의 권한을 제한하는 것에만 중점을 두었을 뿐이고, 사용자와 오브젝트의 관계에 대해서는 고려를 하지 않았다. 본 논문에서는 사용자와 오브젝트와의 관계를 고려한 강화된 악성객체 방지 기법에 대하여 논하고자 한다. 2장에서는 본 논문의 기본 아이디어에 대해서 설명한다. 3장에서는 사용자와 오브젝트의 관계에 대하여 설명한다. 4장에서는 시스템의 구조에 대해서 설명하고, 마지막으로 5장에서는 결론 및 향후 연구 방향에 대하여 설명한다.

2. 기본 아이디어

1장에서 언급한 샌드박스, 맵박스, Sub-Operating system등과 같은 방법은 각 오브젝트의 권한을 미리 정의한다. 그리고 오브젝트가 실행을 하다가 정상행위 이상의 권한을 요구하게 될 경우 악성이라고 판단하게 하는 방법을 사용하고 있다.

그러나 이러한 방법 모두 오브젝트와 사용자와의 관계에 대해서는 고려를 하고 있지 않다. 각 오브젝트들이 정상행위로 정의된 권한 안에서도 충분히 악성의 행위를 할 수 있다. 맵박스에서 정의한 에디터 클래스의 경우 파일을 읽고 쓸 수 있다. 이 경우에 사용자와 오브젝트와의 관계를 고려하지 않는다면 정보의 유출이 있을 수

있다. 관리자만이 접근이 허용되는 파일의 정보가 다른 사용자에게로 유출이 될 수 있다.

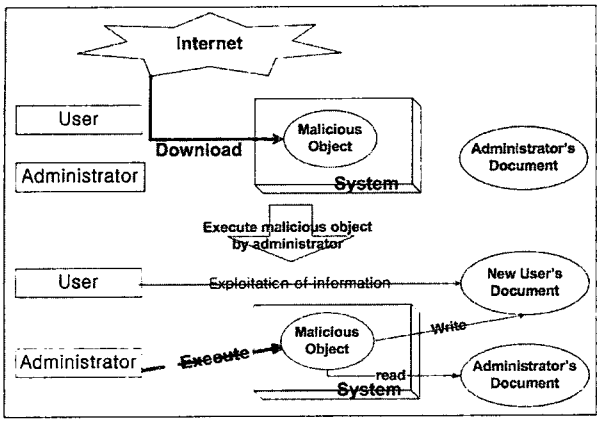

그림 1 일반사용자에 의해 다운로드 된 악성 오브젝트를 관리자가 실행 하였을 때, 발생 할 수 있는 정보의 유출.

예를 들어, 관리자의 권한이 없는 일반 사용자가 정보의 유출을 의도 하기위하여, 인터넷으로부터 에디터 클래스에 속하는 악성 행위를 하는 오브젝트를 다운받는다. 그리고 이렇게 저장된 악성 오브젝트를 관리자가 실행 시킬 경우, 악성 오브젝트는 관리자가 접근 할 수 있는 파일의 접근이 허용된다. 맵박스에서 정의한 에디터 클래스의 경우 파일의 읽기/쓰기가 허용되기 때문에 관리자만이 접근할 수 있는 파일을 읽어서, 악의를 가진 사용자가 접근할 수 있는 새로운 파일을 만들 수 있다. 이 경우 정보의 유출이 일어날 수 있다. 또한 맵박스의 단점인 정적인 클래스의 정의 문제를 해결한기 위한 동적인 클래스의 정의를 다룬 경우, 에디터 클래스에서도 네트워크의 접근이 허용된다.[3] 이런 경우 네트워크를 통한 정보의 유출도 가능하다. 바이러스의 경우도 마찬가지 이다. 윈도우 시스템의 경우 MS-Office계열의 제품군에서 작동하는 매크로 바이러스가 많이 존재한다. 악성 오브젝트가 매크로 바이러스를 파일에 감염 시킬 수 도 있다.

위의 예제의 경우 오브젝트의 권한만을 제한시키는 것으로는 막을 수 없는 일들이 발생한다. 즉 정상행위로 정의해 놓은 범위 안에서도 악성행위를 할 수 있다. 따라서 시스템에 새로운 오브젝트가 들어오는 경우, 이 오브젝트를 다운로드한 사용자를 고려하여야 한다. 즉, 사용자와 오브젝트와의 관계에 대해서도 고려를 해야 한다. 예를 들어, 현재 대부분의 시스템은 관리자의 경우 모든 파일에 대한 접근 권한을 가지고 있다. 하지만, 관리자의 경우라 할지라도 검증되지 않은 오브젝트에 대해서는 접근을 허용하지 않아야한다.

3. 사용자와 오브젝트와의 관계

이 논문에서는 오브젝트가 악성 코드를 포함한 것인지에 대한 판단을 하는 것이 아니다. 기존의 Sandbox, MAPbox, Secure Web Browser에서 고려되지 않은 사용자와 오브젝트의 관계를 고려하기 위함이다.

앞에서 말했던 바와 같이 실행 가능한 오브젝트에 대해서는 오브젝트가 정상행위로 정의해 놓은 클래스 안에서도 악성행위를 할 수 있기 때문에 관리자의 권한을 가졌을 지라도 무조건 실행을 하게 해서는 안 된다. 일반적으로 인터넷을 통해 다운로드 되는 실행 가능한 오브젝트의 유형에 대해 다음과 같이 분류 하였다. 설치 없이 실행을 할 수 있는 파일과 실행을 하기 위해서는 설치가 필요한 파일로 분류 하였다. 그리고 사용자의 그룹은 현재 나와 있는 시스템들 중 사용자의 그룹이 가장 잘 분류되어져 있는 윈도우2000 시스템의 그룹을 사용하여 오브젝트와의 관계를 정리 하였다.

사용자 그룹	허용된 권한
Administrator	모든 object에 대하여 읽기, 실행, 설치 가능하다. 단, 검증이 안 된 object에 대하여 무조건 읽기, 실행, 설치 할 수 없다.
Backup Operators	모든 object에 대하여 백업을 위한 읽기만 가능하다. 어떤 object도 실행, 설치, 다운로드 할 수 없다.
Guests	어떤 object도 읽기, 실행, 설치, 다운로드 할 수 없다.
Power Users	자신이 다운로드한 object만 읽기, 실행을 할 수 있고, 관리자가 인증한 object를 실행, 설치 할 수 있다.
Users	자신이 다운로드한 object만 읽기, 실행을 할 수 있고, 관리자가 인증한 object를 실행 할 수 있다.

표 1. 사용자와 오브젝트의 관계

기본적으로 실행 가능한 오브젝트에 쓰는 행위를 하는 경우는 바이러스를 제외하고 없기 때문에 쓰기권한은 모두 금지된다. 각 사용자는 자신이 들여온 오브젝트에 대하여 읽기, 실행의 권한을 가지고 있다. 단 Backup Operator와 Guest 그룹의 경우는 어떤 오브젝트에 대해서도 읽기, 실행 그리고 설치를 할 수 없다. 나머지 그룹들은 관리자에 의해서 검증이 된 오브젝트에 대해서는 읽기 그리고 실행을 할 수 있다. 오브젝트에 대하여 처음에는 관리자도 접근을 하지 못하게 하였으나, 관리자의 판단에 의해서 접근 권한을 바꿀 수 있을 것이다. 안

전하지 않은 오브젝트에 대해서는 접근을 못하게 하는 것이 좋으나, 꼭 필요한 경우 관리자에 의해서 접근 권한을 바꿀 수 있기 때문에 관리자의 역할이 중요하다.

표1에 정리되어 있는 오브젝트와 사용자와의 관계는 오브젝트와 사용자 그룹과의 관계를 정리한 것이지만, 오브젝트와 사용자의 관계는 그룹이 아닌 각 사용자와의 관계를 설정하는 것도 가능할 것이다.

4. 시스템 구성

현재 구현된 시스템은 그림2와 같이 Monitor, Filter, Object ID Generator, Permission Checker등의 모듈로 구성되어 있다.

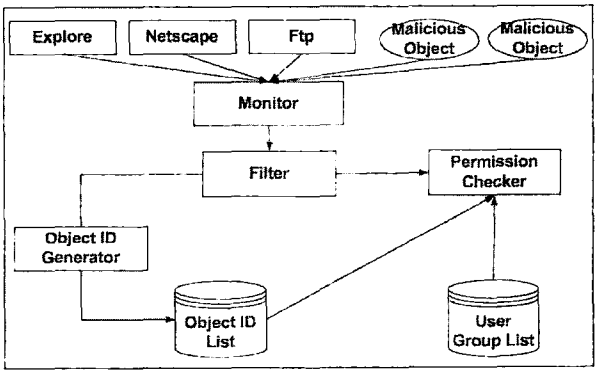

그림 2. 시스템 구성 모듈

Monitor에서의 기능은 시스템에서 일어나는 모든 파일의 입출력 이벤트를 가로채는 역할을 한다. 또한 Monitor에서는 이벤트를 발생시킨 프로세스 이름, 이벤트 종류, I/O가 수행될 파일 이름, I/O가 수행 될 파일 경로, 프로세스의 pid등을 생성한다.

Filter에서는 시스템에 새로운 오브젝트가 들어오는 것을 확인하기 위하여, 웹 브라우저나 FTP에 의해서 발생되는 이벤트를 필터링한다. 그리고 Object ID가 할당된 오브젝트의 실행 여부를 판단하기 위하여, Object ID List에 속한 오브젝트에 대한 이벤트를 필터링한다.

Object ID Generator는 시스템에 들어오는 실행 가능한 각각의 오브젝트에 대하여 Object ID를 할당하여 Object ID List에 저장한다. Object ID List에는 오브젝트를 시스템으로 가져온 사용자와, 오브젝트에 대한 접근 권한을 나타내는 Permission bit가 저장된다. 접근 권한은 표1의 사용자와 오브젝트와의 관계에 의하여 지정된다. 개별적인 사용자와 오브젝트와의 관계를 고려할 수 도 있지만, 본 논문에서는 사용자가 속한 사용자 그룹과 오브젝트의 관계에 대하여 고려하였다.

Permission checker에서는 Object ID가 할당된 오브젝트의 읽기, 쓰기 그리고 실행에 대한 접근권한을 판단한다. Object ID가 할당된 오브젝트의 접근이 시도되었을 경우 현재 사용자 그룹과 오브젝트를 시스템으로 가져온 사용자 그룹(Object ID List에 저장된 사용자 그룹)과 Permission bit를 고려하여 접근을 허용 여부에 대하여 판단한다.

5. 결 론

기존의 샌드박스, 맵박스, Secure Web Browser에서는 각 오브젝트가 가져야 할 정상행위를 정의해 놓고, 이 행위에서 벗어난 행동을 하면 악성으로 판단하였다. 그러나 이러한 방법들은 오브젝트와 사용자의 관계에 대해서는 고려를 하지 않았기 때문에 정상행위로 허용되는 범주 안에서도 악성 행위를 할 수 있다.

본 논문에서는 사용자와 실행 가능한 오브젝트와의 관계를 고려하여 정상행위에서 발생할 수 있는 악성행위에 대하여 보완을 하였다. 하지만 아직까지는 실행 가능한 오브젝트와 사용자의 관계에 대해서만 고려를 하였기 때문에, 샌드박스, 맵박스, Secure Web Browser등에 적용을 하기에는 낮은 효율성을 가지고 있다. 현재는 악성 오브젝트로 의심되는 것에 대해서는 접근을 금지하고 다. 그러나 맵박스에서 사용자와 오브젝트와의 관계를 고려하기 위해서는 오브젝트가 속한 클래스의 특성에 따라 사용자와의 관계를 고려해야 할 것이다.

향후 과제로서 맵박스에서 정의해 놓은 각 클래스에 대하여 사용자와 클래스와의 관계를 연구해 보고자 한다.

6. 참고문헌

[1] Vassilis Prevelakis, Diomidis Spinellis. "Sandboxing Applications"

[2] Anurag Acharya, Mandar Raje. "MAPbox: Using Parameterized Behavior Classes to Confine Applications"

[3] 김철민. "샌드박스의 동적 클래스 계층구조를 통한 악성코드 탐지 기법"

[4] Sotiris Ioannidis and Steven M. Bellovin. "Building a Secure Web Browser." 2001.

[5] Sotiris Ioannidis and Steven M. Bellovin. "Sub-Operating Systems: A New Approach to Application Security." 2000.

[6] Paul C. Clark. "Policy-Enhanced Linux."

[7] Mastering Windows2000 Server

[8] Jonathan D. Moffett. "Specification Of Management Policies and Discretionary Access Control"