DBpia

자기 복제 성질을 이용한 웜 탐지 기법에 대한 연구

정회원 황 유 동*, 박 동 규**, 유 승 엽**, 임 황 빈***, 장 종 수****, 오 진 태****

A study of the worm detection method using self-replication

Yu-dong Hwang*, Dong-Gue Park**, Seung-Yeop Yoo**, Hwang-Bin Yim***,
Jong-Soo Jang****, Jin-Tae Oh**** ***Regular Members***

요 약

본 논문에서는 웜의 변종과 Polymorphic Worm, 그리고 알려지지 않은 공격이 보안 패치나 시그니처가 생성되기 전에 발생하는 Zero-Day 공격에 실시간으로 대응하기 위하여 Polymorphic 웜의 자기복제 성질을 이용한 탐지 기법에 대하여 연구하였고, 이를 기반으로 SSDT (System Service Dispatch Table)를 이용한 웜 탐지 시스템을 설계 및 구현하였다. 구현된 시스템은 SSDT를 커널 모드에서 액세스하여 시스템 콜을 모니터링 하는 가상의 디바이스 드라이버와 모니터링 된 데이터를 저장하고 분석하는 분석 시스템으로 구성된다. 모니터링 된 데이터는 GSR 구조에 따라 분석하였으며, 자기 복제 성질을 갖는 웜의 GSR을 이용하여 시스템의 웜 탐지 여부를 시뮬레이션 하였다.

Key Words : Polymorphic Worm, SSDT, GSR, self-replication

ABSTRACT

In this paper, we studied about detection technique by self-replication nature of Polymorphic worm to real time cope with Zero-Day attack such as worm variant and Polymorphic Worm, and unknown attack of worm those happen before security patch or signature is created. Also we designed and implemented worm detection system that use SSDT(System Service Dispatch Table). The implemented system is consist of virtual device driver that monitor system calls by access to SSDT in kernel mode and analyze system that store and analyze the monitored data. We analyzed the monitored data considering GSR(Gene of Self Replication) structure and simulate the worm detection system whether worm is detected or not.

Ⅰ. 서 론

정보기술의 발달로 누구나 네트워크를 통하여 텍스트, 이미지 등의 멀자기 복제 성질을 이용한 웜 탐지 기법에 대한 연구크 기술의 발달과 함께 네트워크를 통한 웜의 공격 형태 역시 특정 호스트나 서버를 목표로 하여 개별 시스템이나 해당 시스템이 제공하는 서비스에 대한 공격에서 광역 네트워크 자체에 대한 공격과, 공격의 부수적인 효과로 광역 네트워크의 서비스 제공을 방해하는 형태로 변화하고 있다. 또한, 네트워크 인프라 속도의 증가와 서버들의 성능 개선으로 인하여 공격 피해의 전파 속도가 단축되면서 공격 범위가 광역화 및 분산화 되면서 더욱 치명적인 형태로 진화하고 있다. 이러한 네트워크의 침해를 방지하기 위하여 침입 탐지 기술을 사용하고 있으나, 현재의 침입 탐지 및 방지

* 순천향대학교 정보보호학과(hwangyudong@gmail.com), ** 순천향대학교 정보통신공학과 (dgpark@sch.ac.kr, yoosy35@nate.com)
*** 강원도립대학 정보통신공학과(hbinyim@gw.ac.kr), **** 한국전자통신연구원(jsjang@etri.re.kr, showme@etri.re.kr)
교신저자 : 박동규
논문번호 : 09029-0601, 접수일자 : 2009년 6월 1일

www.dbpia.co.kr

기술은 오탐율이 높고, 지속적으로 출현하는 변종 웜, 지속적으로 출현하는 변종 웜, Polymorphic 웜, Zero-Day 공격 등에 대하여 신속하게 대응하지 못하는 문제점을 가지고 있으므로 이러한 공격을 실시간으로 탐지하고 대응하여 네트워크 인프라를 보호하기 위한 기술 개발이 절실히 요구되고 있다.

본 논문에서는 웜의 변종과 Polymorphic Worm, 그리고 알려지지 않은 공격이 보안 패치나 시그니처가 생성되기 전에 발생하는 Zero-Day 공격에 실시간으로 대응하기 위하여 Polymorphic 웜의 자기 복제 성질을 이용한 탐지 기법에 대하여 연구하였다.

본 논문의 구성은 다음과 같다. 2장에서는 다양한 종류의 웜 탐지 기법에 대하여 살펴보고, 3장에서는 본 논문에서 제작한 웜 탐지 시스템에 대하여 살펴보고, 4장에서는 논문의 결론에 대해 논의한다.

Ⅱ. 웜 탐지 기법 연구

네트워크의 발달로 웜 공격 또한 그 전파 속도가 빨라지면서 이에 대처하기 위한 여러 가지 웜 탐지 기법이 연구되었다. 대표적인 웜 탐지 기법으로 시그니처 기반 생성 기법, 데이터 마이닝 기법, 하이브리드 방법, GSR을 이용한 웜 탐지 기법들이 있다.

2.1 시그니처 기반 생성 기법

시그니처 기반의 공격 탐지 기술은 웜 탐지를 위하여 가장 잘 알려져 있는 대표적인 방법이라고 할 수 있다. 성공적인 시스템으로는 Snort[1]와 Bro[2]가 있으며 다형성 웜 탐지를 위한 시그니처 생성을 위한 연구로 Polygraph[3]와 Hamsa[4] 등을 들 수 있다. 이 방법은 복합적인 다형성 웜 샘플들 사이에서 불변하는 공통된 문자열을 찾아 시그니처를 생성하는 방법으로 트레이닝을 통하여 의심스러운 트래픽을 미리 선별하는 네트워크 트래픽 분류기를 필요로 한다. 이 분류기에 의해서 다형성 웜 탐지와 오탐률이 결정된다.

2.2 데이터 마이닝 기법

데이터 마이닝 기법[5]은 단순한 NO-OP 명령어 탐지와 Neural Network를 결합시킨 것으로 이 방식에서 Neural Network는 negative와 positive 데이터 집합으로 훈련되어야 하며, 해당 데이터 집합들에 의해 탐지율이 결정된다. 이 방식은 실제로 높은 양질의 훈련 집합을 구하기도 어려우며 항상 새로운 상태로 업데이트를 유지시키기도 어려운 단점을 가지고 있다.

2.3 하이브리드 탐지 기법

다형성 웜 탐지를 위한 네트워크 레벨의 하이브리드 방법으로 static-analysis와 emulation 방법이 있다.

static-analysis는 polymorphic exploit의 특별하고 non-trivial 한 클래스를 탐지하는 방식이라고 할 수 있다. Polymorphic exploit들은 버퍼 오버플로우 취약성을 목표로 하며, 이 exploit 클래스는 제어 흐름과 데이터 흐름의 특성을 탐지 할 수 있는 프로그램 코드를 포함하게 된다. 현재까지 연구된 이 방식[9]-[11]들은 exploit 코드의 불명확한 시작 위치 때문에 self-modifying과 간접 제어 전송 명령어들과 같은 static analysis-resistant 기술을 사용하는 exploits 공격에 대처할 수 없다.

self-modifying polymorphic exploit code를 보다 효율적으로 탐지하기 위하여 명령어 emulation 방식을 사용하는 방법[12]이 제시되었다. 그러나 이 방식도 네트워크 트래픽 내에 polymorphic exploit code의 시작 위치를 확인할 수 있는 방법을 제시하지 못하는 단점을 가지고 있다. 이 방식은 모든 가능성이 있는 시작 위치에 대하여 시도된다면 너무 속도가 느려지는 문제가 있고, exploit code의 가능한 시작 위치를 줄이기 위하여 단순한 경험적인 방식을 적용한다면 일부 공격을 놓치게 되는 단점이 있게 된다.

최근에 이 두 방법을 이용하여 self-decrypting exploit code를 탐지하기 위한 새로운 방법[13]이 제시되었다. 이 방식에서는 Polymorphic exploit code의 시작 위치를 발견하고 명령어들을 확인하기 위하여 static analysis와 emulation 방식을 둘 다 사용하였다.

2.4 GSR을 이용한 웜 탐지 기법

자기 복제 성질을 이용하는 웜 탐지 기법의 최근 연구는 GSR(Gene Self-Replication)을 이용하는 방식으로 이 방식은 웜 바이러스의 알려진 시그니처에 의존하지 않고, 실행시간 동안에 자기 복제를 위한 실행 코드를 발견하는 방법[8]이다. 사용자에 의해서 실행되는 거의 모든 어플리케이션의 function calls은 사용자 모드에서 실행되고, 파일 시스템, 프로세스와 쓰레드, 그래픽 시스템 서비스, 시스템 러지스트리 등과 같은 하드웨어 리소스를 사용하는 기능은 커널모드에서 실행된다.

프로세스가 동작하는 동안의 function call은 sys-

www.dbpia.co.kr

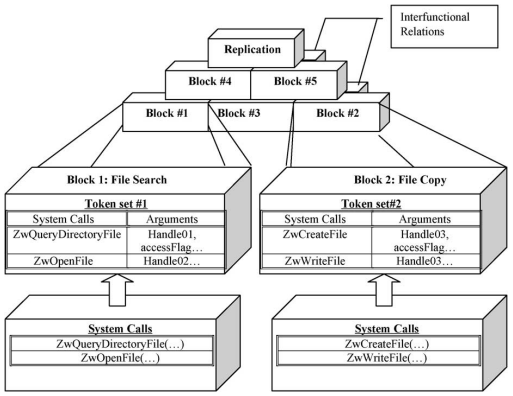

그림 1. GSR의 구조

tem call로 넘어가게 되고 system calls은 커널 레벨에서 SSD(System Service Dispatcher)에 의해 동작하게 된다. 자기 복제하는 웜들은 커널모드에서 동작하게 되므로 커널 모드에서 system calls을 모니터링하게 되면 자기 복제를 위한 웜을 탐지 할 수 있게 된다. 이 탐지 기법은 자기 복제하는 시퀀스가 존재하며 코드의 entire body를 통해서 웜이 전파 된다는 이론을 이용한다. 이것은 explicit 패턴으로 탐지가 불가능하며 여러 가지 방법으로 활동될 수 있으나 이 수는 한정되어있다. 또한 polymorphic Worms은 매번 다른 키로 암호화해서 새로운 웜을 생성한다. 따라서 복호화하기 위해 가상머신으로 보내지기 전에 드러나는 복호화 루틴이 정해져 있으며 이를 이용하는 unknown malicious software들이 존재하고 있다. 그 중 GSR의 이용은 컴퓨터 바이러스성 소프트웨어의 가장 근본적인 속성을 다루고 malware의 넓은 범위의 신뢰 있는 탐지가 가능하다. 웜은 공격으로 최대의 타격을 줄 수 있도록 컴퓨터를 감염 시킨다. 따라서 이러한 공격을 탐지하는 것은 시스템 콜의 시퀀스에서 자기 복제하기 위해서 활발하게 나타나는 시스템 콜을 찾으면 폭을 좁힐 수 있다.

GSR을 이용한 웜 탐지 기법은 더 없이 일반적이다. 따라서 어느 컴퓨터 시스템이든 적용이 가능하다. 실행되는 모든 프로세스는 System calls에서 시작되며 웜이 아닌 일반적인 system call은 오브젝트 핸들, 포인터, 메모리 옵셋에 의해서 수행된다. 그렇지 않은 system call은 malicious라 고려되며 위 그림 1과 같이 GSR 블록 형태로 나타나게 된다

이 탐지 기법은 컴퓨터 바이러스성 소프트웨어의 가장 근본적인 속성을 다루고 malware의 넓은 범위의 신뢰 있는 탐지 기법으로 시스템 콜 인수를 고려하여 변종 웜 탐지 에러를 발생 시키지 않는다. 그러나 malicious와 normal의 행동차가 작음으로 자기 복제 인자를 정확히 정의하여야 한다는 단점이 있다.

Ⅲ. 자기 복제 성질을 이용한 웜 탐지 시스템 구현

3.1 시스템 구성

자기복제 성질을 이용한 웜 탐지 시스템은 SSDT를 이용하여 시스템 콜을 수집하기 위한 윈도우 디바이스 드라이버와 수집된 시스템 콜 정보를 분석하여 웜 인지 정상적인 프로세스인지를 판별하기 위한 소프트웨어로 구성된다. 다음 그림 2는 웜의 자기복제 특성 탐지를 위한 윈도우 시스템 콜 로그 저장 및 패턴 분석 도구의 구성도이다. 점선 사각형 부분은 프로세스로부터 호출되는 시스템 콜을 가상의 디바이스 드라이버로 모니터링하여 붉은 색의 유저모드에서 동작하는 분석 시스템으로 데이터를 전달하고 데이터 분석 시스템은 전달받은 데이터를 이용하여 시스템 콜을 한 프로세스가 정상적인 프로세스 인지 웜 인지 판별하게 된다.

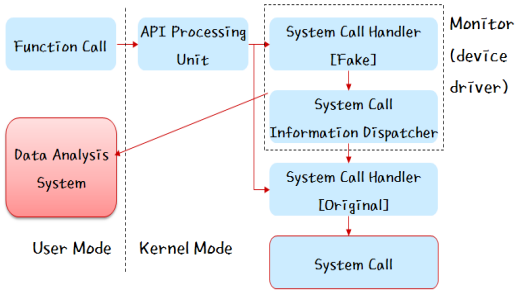

그림 2. 시스템 구성도

3.2 자기 복제 성질을 이용한 웜 탐지 시스템 알고리즘

자기복제 성질을 이용한 웜 탐지 시스템은 SSDT를 이용하여 시스템 콜을 수집하기 위한 윈도우 디바이스 드라이버[1],[2],[1]와 수집된 시스템 콜을 로그로 저장하고, 저장된 로그 정보를 분석하여 웜 인지 정상적인 프로세스 인지를 판별하기 위한 소프트웨어로 구성된다.

3.2.1 로그 분석 알고리즘.

자가 복제를 탐지하기 위해서는 탐지되는 시스템 콜의 모든 입력과 출력 파라미터들이 고려되어야 한다. 어떤 종류의 시스템 콜들은 입력과 출력 파라

www.dbpia.co.kr

미터들의 가능한 조합들로 그 조합이 복제의 위협으로 받아들여 질 수 없는 반면, 특정 파라미터들로 호출된 특정 시스템 콜은 자가 복제의 명확한 패턴을 나타낸다.

GSR 탐지 프로세스가 동작 중에, 모니터링된 모든 시스템 콜은 바로 탐지와 필터 메커니즘이라고 불리는Replication Detector로 들어오게 된다. GSR은 여러 블록으로 구성되며, 탐지 메커니즘은 호출된 시스템 콜이 GSR 피라미드 구조의 꼭대기에 도달하기 전까지 모든 블록들과 관련된 판단을 하고 그 판단된 결과를 감시한다 그리고 시스템 콜이 탐지된 현재의 알람 상태를 알리게 된다. 다음 그림 3은 간단한 블록 탐지 알고리즘 다이어그램이다. 그림 3의 블록 탐지 알고리즘[8]은 다음의 순서로 동작한다.

1) Detector가 시스템 콜을 탐지한다.

2) 탐지된 시스템 콜을 History Tracer가 GSP데이터베이스의 lower level block에 있는 시스템 콜과 결합이 가능한지를 판단한다.

- 가능성이 없다면 어떠한 동작도 하지 않는다
- GSR 데이터베이스에 어떤 시스템 콜도 들어가 있지 않다면 가장 먼저 들어갈 자리에 특정 시스템 콜(예. FileOpenBlock)을 집어 넣는다.

3) 결합이 가능한 두 개의 시스템 콜(최근 탐지된 시스템 콜과 GSP데이터베이스에서 결합 가능한 콜)을 가져와서 Combiner가 두 콜을 결합하여 upper level block을 형성한다.

4) 형성한 Upper Block의 입력 값이 최근에 탐지된 콜에 입력 갑과 동일하고, GSP 데이터 베이스에서 가져온 콜에 출력 값과 Upper Block의 출력 값으로 상속되는지 검사한다.

5) 만들어진 Upper Block을 GSP 데이터베이스에 업데이트 한다.

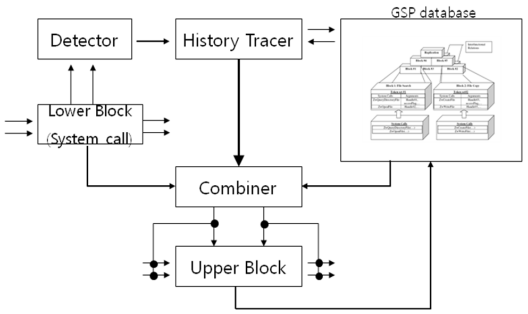

그림 3. 로그 분석 알고리즘

위의 알고리즘은 GSR 피라미드 구조에 꼭대기 블록이 만들어 지기 전까지 반복된다.

3.2.2 시스템 콜 수집 드라이버

B.Skormin 자기복제 웜을 탐색하기 위하여 그림 3.1과 같은 피라미드 구조를 이루는 블록으로 구성된 GSR을 사용한다.

GSR에서 웜은 Host Search - File Access - Networking - Memory Injection / infection의 과정을 거쳐 자기복제를 하게 된다.

다음 그림 4는 File Access Block의 예를 보여주고, 그림 5는 Host Search Block, 그림 6은 Memory Mapping Block, 그림 7은 Code Injection Block의 예를 보여준다.

본 연구에서 제작한 디바이스 드라이버는 그림 4, 5, 6, 7에서 보여주는 시스템 콜을 모니터링 할 수 있도록 제작하였다. 위 그림 4, 5, 6, 7은 자기복제 웜이 복제를 할 때 웜의 탐색 예를 보여준다.

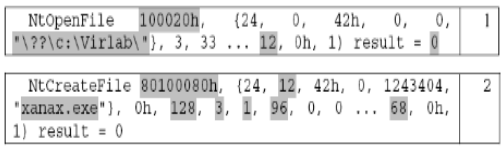

그림 4. File Access Block

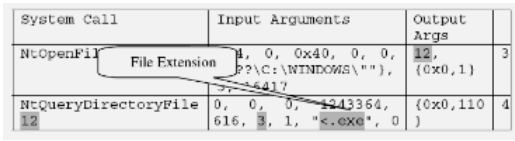

그림 5. Host Search Block

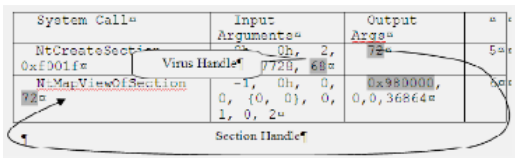

그림 6. Memory Mapping Block

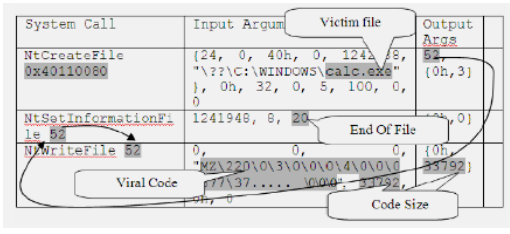

그림 7. Code Injection Block

www.dbpia.co.kr

자기 복제 웜이 동작되었을 때, 웜은 윈도우 디렉터리에서 실행 파일을 찾고 파일들의 엔트리 포인트를 변경하면서 파일들로 복제를 하게 된다. 이 웜은 복제 알고리즘을 따른다. 모니터를 통해 "Virlab" 디렉터리를 열어 자기 스스로 접근하는 행동을 따라 웜 행동과 관련된 복제가 관찰 되었다.

위 그림 4, 5, 6, 7의 각 블록의 설명은 다음과 같다.

-. File Access Block

그림 4에서 NtOpenFile 시스템 콜이 성공적으로 실행 완료되면서 새로운 디렉터리 핸들을 알려주게 된다. 이후에 이 핸들은 "Virlab" 내부 파일들에 접근할 때 사용하게 된다.

많고 복잡한 계산 사이클을 완료 후에 읽기 모드로 자기 스스로 자신의 파일을 열려고 시도할 때 바이러스들은 의심스러운 명령어들을 불러내게 된다. GSR 구조의 정의에 따르면, 그림 5.3의 두 시스템 콜은 File Access Block를 표현하며 더 큰 구조를 형성하기 위해서도 결할 될 수 있다. 이 시스템 콜들은 디렉터리 핸들과 입력 플래그와 같은 몇 가지 다른 중요 매개 변수들로 결합된다. 한 번의 결합으로 만들어진 새로운 구조는 첫 번째 시스템 콜로부터 입력 매개 변수를 받아 두 번째 시스템 콜로 블록의 출력 매개변수를 상속시킨다.

-. Host Search Block

목적 호스트 파일의 위치를 알아낸 이후에, 바이러스는 실행 코드를 통한 제어가 바이러스 코드를 통해서 이루어 질 수 있도록 바이러스 몸체를 호스트 파일에 추가하고 호스트 파일을 열 것이라고 예상된다.

바이러스를 동작시키는 실험에서,매우 높은 전염 가능성을 고려하여 윈도우 디렉터리, 매우 흔한 바이러스 타겟들의 위치를 알아 낼 수 있었다.[8]

전염시키기 위한 호스트 파일을 찾는 동안, 바이러스는 실행 파일의 위치를 알아내는 다른 시스템 콜 쌍을 불러낸다. NtOpenFile과 NtQueryDirectoryFile 이 시스템 콜 쌍은 그림 5의 Host Search Block이라고 불리는 복제 블록을 형성한다.

-. Memory Mapping Block

메모리 매핑 루틴 쌍(시스템 콜 두 개의 결합)은 그림 6의 Memory Mapping Block이라는 이름에 다른 복제 빌딩 블록을 정의한다. 이 블록은 이미 만들어진 File Access Block에 의해 제공되는 입력 매개 변수로써 파일 핸들을 필요로 한다. 이 두 블록(File Access, Memory Mapping Blocks)은 File-in-Memory Block이라는 이름을 가진 새로운 상위 레벨로 결합된다. 이 상위 레벨 블록은 2개의 부모 구조들의 입력과 출력들을 상속하게 된다.

-. Code Injection Block

마지막으로, 바이러스가 메모리에 있고 희생 파일의 실체가 확인 되었을 때, 복제가 성공적으로 완성되기 위해 다른 시스템 콜들의 집합이 필요하다. 그러기 위해서 바이러스 몸체를 호스트 파일에 추가하고 바이러스 코드가 먼저 정식 파일 실행을 허용하는 원래의 호스트 파일에 따라 실행되는 방법으로 코드 엔트리 포인터들을 변경시키게 된다. 그러므로, 이 문제의 바이러스는 호스트 파일을 열고, 정확한 바이러스 코드 삽입에 섹션의 위치를 확인하고 그리고 마지막으로 자신의 코드를 NtWriteFile 시스템 콜을 실행 시킴으로써 바이러스 코드를 추가하게 된다.

복제 과정에서 마지막 연속과정이 되는 NtCreateFile, NtSetInformationFile,, NtWriteFile콜들의 집합은 그림 7의 Code Injection Block라고 불리는 GSR 피라미드의 마지막 블록을 형성한다. NtWriteFile의 출력 값들이 이 블록(Code Injection Block)에 출력 파라미터들이 되는 반면, Code Injection Block은 첫 번째 시스템 콜 NtCreateFile 으로부터 입력 매개변수들을 상속받는다. 이런 방법으로 복제 패턴을 모두 거치고 난 이후에 성공적인 복제 결과를 얻을 수 있다.

결과적으로 웜의 자기복제 특성 탐지를 위한 윈도우 시스템 콜 로그 저장 및 패턴 분석 도구는 위 그림 4, 5, 6, 7에서 보여주는 7개의 시스템 콜을 가상 디바이스 드라이버에서 모니터링하며 실시간으로 호출된 시스템 콜을 로그 저장 및 패턴 분석을 위한 데이터 분석 시스템에 전송한 후, 로그 저장 및 분석도구는 전송된 데이터를 로그로 저장하고 웜의 탐지를 위하여 패턴을 분석한다.

웜이 자기복제를 하는 동안 위에서 언급된 시스템 콜들은 순서대로 호출되고, 호출되는 시스템 콜의 순서와 호출되는 시스템 콜의 파라미터의 데이터를 이용하여 웜의 자기복제를 판별한다.

www.dbpia.co.kr

Ⅳ. 자기 복제 성질을 이용한 웜 탐지 시스템 구현 예

4.1 로그 저장 및 분석 도구

자기복제 성질을 이용한 웜 탐지 시스템은 시스템 콜을 모니터링 하기 위한 가상의 디바이스 드라이버와 모니터링된 데이터를 분석하기 위한 분석 시스템으로 이루어졌다. 디바이스 드라이버는 분석 시스템에서 로드와 동시에 SSDT를 후킹하게 되며 드라이버에서 전달되는 시스템 콜의 데이터를 분석 시스템에서 로그로 저장 및 화면 출력하여 보여주고, 내부적으로 웜의 자가복제 동작 여부를 판별하는 패턴 분석을 하게 된다. 다음 그림 8은 정상적으로 로드 된 시스템 콜 모니터링 드라이버의 메시지 출력 화면이다.

위에서 언급한 자가 복제의 행동으로 의심될 수 있는 시스템 콜들의 데이터들이 실시간으로 커널모드에 있는 가상 디바이스 드라이버로부터 오는 것을 볼 수 있다. 커널모드에 있는 가상 디바이스 드라이버로부터 받은 데이터를 탐지 메커니즘에서 사용하기 용이하도록 분석 및 분류하여 로그로 저장하면 실시간 후킹 되는 모든 시스템 콜들의 로그 저장을 완료하게 된다. 제한된 Windows 2000운영체제 환경에서 자가 복제하는 프로세스와 정상적인 동작을 수행하는 프로세스가 각기 따로 실행 될 때 호출되는 모든 시스템 콜들을 수집하여 분석한 결과 자가 복제하는 프로세스가 실행 되었을 때 추출된 로그에서 자가 복제의 시도를 볼 수 있었다.

다음 그림 9는 자가 복제하는 프로세스가 실행되었을 때 발생하는 시스템 콜을 GSR블록 레벨에 따라 구분하여 그래프로 표시하였다. 그림 9의 y축은 GSR의 피라미드를 구성하는 블록들을 의미한다. 예를 들어 y축 레벨 0은 일반적인 정당한 프로세스

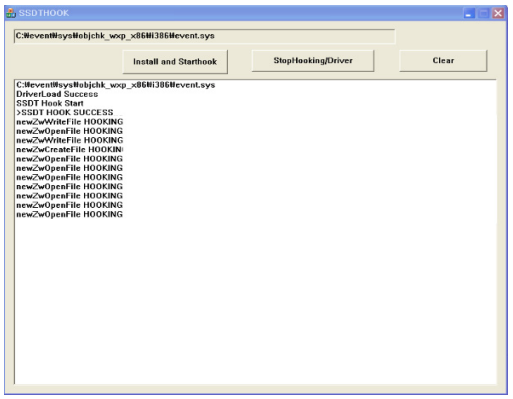

그림 8. 시스템 콜 모니터링 후 정상적인 서비스 호출 예

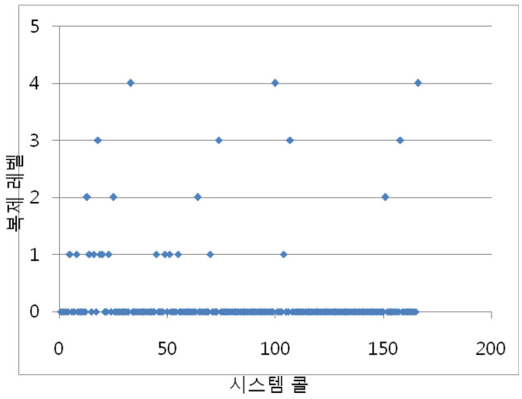

그림 9. 자가 복제 프로세스 실행 시 발생하는 시스템 콜과 GSR 블록 레벨의 연관 그래프

들의 시스템 콜 호출이라고 볼 수 있으며 "1"은 File Access Block, "2"는 Host Search Block, "3"은 Memory Mapping Block 그리고 마지막으로 "4"는 Code Injection Block을 의미하며 마지막 블록 이후에 복제가 이루어 진다.

위 그림 9에서 GSR 블록 레벨에 따라 세 번의 자가 복제가 탐지 되었음을 알 수 있다.

아래 그림 10은 자가 복제 프로세스가 실행되지 않은 상태에서 발생한 시스템 콜의 GSR 블록 레벨 분석 그래프이다. 자가 복제 프로세스가 실행되지 않은 정상적인 동작을 수행하는 일반적인 프로세스들의 행동들을 감시한 결과, 복제의 시도로 오해가 가능할 정도로 많은 시스템 콜에서 복제 레벨 1~3까지 폭 넓게 분포하는 것을 볼 수 있다. 하지만 복제 시도로 판단할 수 있는 GSR블록 레벨 4에 해당하는 시스템 콜의 조합은 없었다. 위 그림 10에서 운영체제와 여러 프로세스가 동작 될 때 웜 탐지를

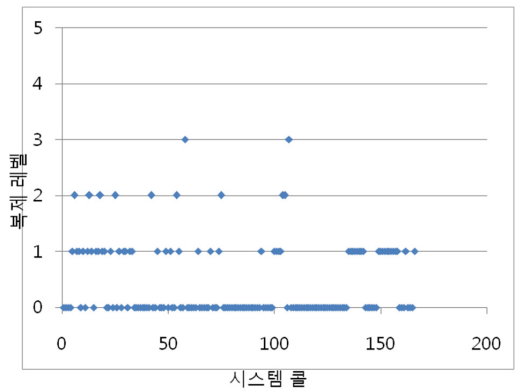

그림 10. 자가 복제를 하지 않는 일반 프로세스 실행 시 발생하는 시스템 콜과 GSR 블록 레벨의 연관 그래프

www.dbpia.co.kr

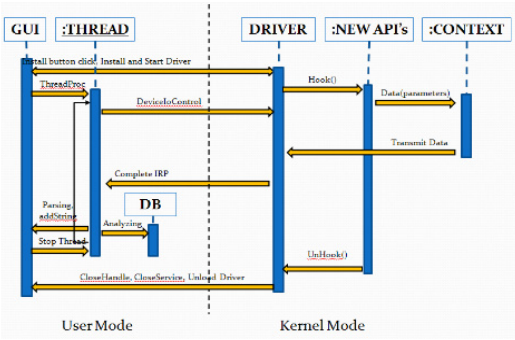

그림 11. 웜의 자기복제 특성 탐지를 위한 윈도우 System Calls 로그 저장 및 패턴 분석 도구의 시퀀스 다이어그램

위한 GSR 블록에 해당하는 시스템 콜이 반복 동작되어 실행됨을 알 수 있다. 따라서, GSR 블록 레벨 4에 해당하는 Code Injection Block까지 이어지는 시스템 콜의 정확한 탐지가 없으면 오탐율이 높을 수 있음을 알 수 있다.

자가 복제 프로세스의 정확한 탐지를 위해서 디바이스 드라이버에서 받은 데이터를 정밀하게 분석하여 복제 레벨의 단계들이 되는 피라미드의 한 블록 한 블록을 구성하는 시스템 콜 쌍이 결합할 때 입출력으로 들어오는 매개 변수와, 상위 하위 블록 간에 상속 데이터, 희생 파일의 위치 등등을 사전 검증하며 이전 기록들을 유지하여 복제 행위의 연속된 과정과 블록들 간에 상호 연관을 고려하는 프로그램을 추가하였다.

다음 그림 11은 웜의 자기복제 특성 탐지를 위한 윈도우 시스템 콜 로그 저장 및 패턴 분석 도구의 시퀀스 다이어그램이다.

분석도구를 실행하게 되면 먼저 디바이스 드라이버를 로드하게 되고, 로드 된 디바이스 드라이버는 SSDT를 Hook하여 웜의 자기복제에 사용되는 native API와 파라미터에 대한 정보를 로그 저장 및 분석 도구로 전송한다. 로그 저장 및 분석도구는 드라이버로부터 전송된 데이터를 데이터베이스에 저장을 하고 저장된 데이터베이스의 데이터를 분석하여 자기복제 중인 웜이 있는지 없는지 여부를 분석하게 된다.

4.2 시스템 콜 수집 드라이버

본 연구에서 제작한 디바이스 드라이버는 다음과 같은 네 부분으로 이루어져 있다.

첫 번째, 디바이스 드라이버가 로드 될 때 SSDT를 액세스하여 저장된 시스템 콜의 주소를 변경하여, 시스템 콜이 호출될 때 호출된 시스템 콜의 파라미터를 모니터링 할 수 있는 변형된 시스템 콜이 저장된 주소로 변경하는 루틴.

두 번째, 실시간으로 실행되는 시스템 콜들의 입력과 출력 파라미터를 IOControl Dispatch Routine을 통해 시스템 콜을 분석, 저장하는 분석 도구로 전송하는 루틴.

세 번째, 시스템 콜을 분석/판별할 수 있도록 전송한 다음 운영체제에서 정상적인 서비스를 제공할 수 있도록 변경해 두었던 시스템 콜을 호출해 주는 루틴.

네 번째, 디바이스 드라이버가 로드 될 때 변경했던 시스템 콜의 주소를 원래의 상태로 되돌리는 루틴.

4.3 시스템 콜 로그 분석

다음 그림 12는 본 연구에서 구현한 디바이스 드라이버를 통하여 수집된 시스템 콜을 로그 저장 및 분석 도구가 저장한 시스템 콜 로그의 예이다.

다음 그림 12의 로그는 자기 복제를 하지 않는 정상적인 프로세스에 의해 호출된 시스템 콜들과 자기 복제를 하는 프로세스에 의해 호출된 시스템 콜이 혼재하며, 두 가지 프로세스 모두 같은 시스템 콜들을 호출함을 알 수 있다. 따라서 정상적인 프로세스에 의해 호출된 시스템 콜들도 자기 복제의 시도로 여겨지는 경우도 있을 수 있다. 그러나 다음 그림 12의 로그에서와 같이 자기 복제 특성 탐지를 위한 GSR 정의에 따른 피라미드 구조의 블록을 적용하면, 그림 안에 표시한 블록들을 순서대로 추출할 수 있으므로, 자기 복제 특성을 탐지함을 알 수 있다.

V. 결 론

본 논문에서는 네트워크를 통한 웜 공격에 대처하기 위하여 기존 연구 방법들을 살펴보고 그 중 웜의 가장 근본적인 속성인 자기 복제 성질을 이용한 웜 탐지 기법에 대하여 살펴보았고, 웜을 통한 공격으로부터 네트워크 인프라를 보호하기 위해서 SSDT (System Service Dispatch Table)를 이용한 웜 탐지 시스템을 설계 및 구현하였다. 구현된 시스템은 SSDT를 커널 모드에서 액세스하여 시스템 콜을 모니터링 하는 가상의 디바이스 드라이버와 모니터링된 데이터를 저장하고 분석하는 분석 시스템으로 구성된다. 모니터링 된 데이터는 GSR구조에 따라 분석하였으며, 자기 복제 성질을 갖는 웜의

www.dbpia.co.kr

```
ZwMapViewOfSection,3800
ZwCreateSection,3800,0
ZwMapViewOfSection,3800
ZwOpenFile,2912,0x100020,₩,0,0
ZwCreateSection,1976,2912
ZwMapViewOfSection,1976
ZwOpenFile,1976,0x100020,₩,0,0
ZwCreateSection,2912,1976
ZwMapViewOfSection,2912
ZwCreateFile,4920,0x40100080,0,₩,8320,5,3,96,0
ZwSetInformationFile,4920,14
ZwWriteFile,4920,136,136
ZwCreateFile,4920,0x40100080,0,₩,8320,5,3,96,0
ZwSetInformationFile,4920,14
ZwOpenFile,4428,0x100020,₩,0,0
ZwOpenFile,4920,0x100020,₩,0,0
ZwCreateSection,4428,4920
ZwMapViewOfSection,4428
ZwCreateFile,1976,0x40100080,0,₩,8320,5,3,96,0
ZwSetInformationFile,1976,14
ZwWriteFile,1976,136,136
ZwCreateFile,1976,0x40100080,0,₩,8320,5,3,96,0
ZwSetInformationFile,1976,14
ZwWriteFile,1976,220,220
ZwCreateFile,1976,0x40100080,0,₩,8320,5,3,96,0
ZwSetInformationFile,1976,14
ZwWriteFile,1976,166,166
ZwOpenFile,244,0x100001,₩,0,0
ZwQueryDirectoryFile,244,3,t
ZwCreateFile,256,0x80100080,0,₩,0,1,1,2097252,
```

File Access Block

Host Search Block

```
ZwCreateSection,4920,4428
ZwMapViewOfSection,4920
ZwQueryDirectoryFile,136,3,t
ZwCreateSection,2516,0
ZwMapViewOfSection,2516
ZwCreateSection,2516,0
ZwMapViewOfSection,2516
ZwCreateSection,2516,0
ZwMapViewOfSection,2516
ZwCreateFile,220,0x40110080,0,₩,32,0,5,100,0
ZwSetInformationFile,220,20
ZwCreateSection,2516,0
ZwMapViewOfSection,2516
ZwWriteFile,220,34304,34304
ZwOpenFile,220,0x100001,₩,0,0
ZwQueryDirectoryFile,220,3,t
ZwCreateFile,4856,0x80100080,0,₩,0,0,1,96,0
ZwCreateSection,4180,4856
ZwMapViewOfSection,4180
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
ZwCreateFile,4180,0x20088,0,₩,0,7,1,0,0
```

Memory Mapping Block

Code Injection Block

그림 12. 드라이버에서 전송되어 저장된 시스템 콜 로그 예

GSR을 이용하여 시스템의 웜 탐지 여부를 시뮬레이션 하였다.

또한, 자기 복제 성질을 갖는 시뮬레이션용 어플리케이션을 작성하여 본 논문에서 구현된 웜 탐지 시스템을 이용하여 어플리케이션의 자기 복제 특성을 탐지하였다. 따라서 실제 시스템에 적용해도 자기 복제 특성의 탐지 가능함을 예측할 수 있다.

향후, 연구를 계속하여 자기복제 웜이 있는 다양한 환경에서 구현된 시스템을 시험하고, 자기 복제뿐만 아니라, 다양한 종류의 웜을 탐지할 수 있는 패턴 및 알고리즘에 대한 연구를 통하여 넓은 범위의 신뢰성 있는 웜 탐지가 가능하리라 사료된다.

참 고 문 헌

〔1〕 Snort: an open source network intrusion prevention and detection system, 2005. http://www.snort.org

〔2〕 Bro Intrusion Detection System, 2003. http://www.bro-ids.org

〔3〕 J.Newsome, B. Karp, and D.Song. Polygraph : Automatically Generating Signatures for Polymorphic Worms.InProceedingsof2005 IEEESymposiumonSecurityandPrivacy,pages226-241, May2005.

〔4〕 Z. Li, M.Sanghi, Y. Chen, M. Kao, and B. Chavez. Hamsa : Fast signature generation for zero-day polymorphic worms with provable attack resilience. In Proceedings of 2006 IEEE Symposium on Security and Privacy, pages 32-47, May 2006.

〔5〕 U.Payer, M.Lamberger, and P.Teufl. Hybrid engine for polymorphic code detection. In Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment, pages 19-31

www.dbpia.co.kr

〔6〕 S. Forrest, S. A. Hofmeyr, A. Somayaji, and T.A.Longstaff. A Sense of Self for Unix Processes, In Proceedings of 1996 IEEE Symposium on Computer Security and Privacy, 1996

〔7〕 Jiang N., Hua K., and Sheu S. Considering Both Intra-pattern and Inter-pattern Anomalies in Intrusion Detection. Proc. Intel. Conf. Data Mining, 2002

〔8〕 V.Skormin, A.Volynkin, D.Summerville and J.Moronski. Prevention of information attacks by run-time detection of self-replication in computer codes. Journal of Computer Security 15, pages 273-301, 2007

〔9〕 R.Chinchani and E. Berg. A Fast Static Analysis Approach To Detect Exploit Code Inside Network Flows. In Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection, pages 284-308, September 2005.

〔10〕 P.Akritidis, E.Markators, M.Polychronakis, and K.Anagnostakis. STRIDE: Polymorphic Sled Detection through Instruction Sequence Analysis. In Proceedings of the 20th IFIP International Information Security Conference, Pages 375-392, June 2005.

〔11〕 X.Wang, C.Pan, P.Liu, and S.Zhu. SigFree : A Signature-free buffer Overflow Attack Blocker. In Proceedings of the 15th USENIX security Symposium pages 225-240, July 2006.

〔12〕 M.Polychronakis, K.Anagnostack, and E. Markatos. Network-Level Polymorphic Shellcode Detection Using Emulation. In Proceedings of the Conference on Detetion of Intrusions and Malware and Vulnerability Assessment, July 2006.

〔13〕 Qinghua Zhang, Douglas S. Reeves, Peng Ning, S.Purushothaman Iyer. Analyzing Network Traffic To detect self-decrypting exploit code. In Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, pages 4-12, March 2007.

〔14〕 D. Bruschi, L. Martignoni, and M. Monga. Using code normalization for fighting self-mutating malware. In Proceedings of the International Symposium on Secure Software Engineering, pages 37-44. IEEE CS Press, 2006.

황 유 동 (Yu-dong Hwang) 정회원

1998년 2월 순천향대학교 제어계측 공학과 공학사

2000년 8월 순천향대학교 전기전자공학과 석사

2003년 ~ 현재 순천향대학교 전기전자공학과 정보보호전공 박사과정

<관심분야> 네트워크 보안, 시스템 보안, 접근제어

박 동 규 (Dong-Gue Park) 정회원

1992년 한양대학교 대학원 전자공학과 공학박사

1999 ~ 2003년 순천향대학교 정보기술공학부 부교수

2004년 ~ 현재 순천향대학교 정보통신공학과 교수

<관심분야> 네트워크 보안, 유비쿼터스 컴퓨팅 보안

유 승 엽 (Seung-Yeop Yoo) 정회원

2008년 9월 순천향대학교 정보통신공학과 학사

2008년 9월 ~ 현재 순천향대학교 정보통신공학과 석사과정

<관심분야> 네트워크 보안, 시스템 보안

www.dbpia.co.kr

임 황 빈 (Hwang-Bin Yim) 정회원

2003년 순천향대학교 전기전자 공학과 (공학박사)
2003년~현재 강원도립대학 정보통신과 조교수
<관심분야> 통신응용시스템, 광통신 , 정보 보호

장 종 수 (Jong-Soo Jang) 정회원

1984년 경북대학교 전자공학과 학사 졸업
1986년 경북대학교 전자공학과 석사 졸업
2000년 충북대학교 박사졸업
1989년~현재 한국전자통신연구원 네트워크보안 그룹 그룹장/책임연구원
<관심분야> 네트워크보안, 정책기반 보안관리기술

오 진 태 (Jin-Tae Oh) 정회원

1990년 경북대학교 전자공학과 학사 졸업
1992년 경북대학교 전자공학과 석사 졸업
1992년~1998 한국전자통신 연구원 선임연구원
2003년~현재 한국전자통신연구원 네트워크보안 연구팀 팀장/선임 연구원
<관심분야> 네트워크보안, 비정상행위 탐지기술

www.dbpia.co.kr