연구논문 정보화정책 제24권 제4호, 2017년 겨울호, pp.68~78

분석기법을 우회하는 악성코드를 분석하기 위한 프로세스 설계

이 경 률*, 이 선 영**, 임 강 빈***

요 약 악성코드는 나날이 복잡해지고 다양화되어 단순한 정보유출에서부터 시스템에 대한 심각한 피해를 유발하는 실정에 이르렀다. 이러한 악성코드를 탐지하기 위해 코드분석에 역공학을 이용하는 많은 연구가 진행되었지만, 악성코드 개발자도 분석방법을 우회하는 다양한 기법을 활용함으로써 코드분석을 어렵게 하였다. 특히, 악성코드의 감염여부조차 판단하기 어려운 루트킷 기법들이 진화하고 있고, 악성코드가 이 기법들을 흡수함으로써 그 문제의 심각성은 더욱 커지고 있다. 따라서 본 논문에서는 분석기법들을 우회하는 악성코드에 재빠르게 대응하기 위한 분석 프로세스를 설계하였다. 설계된 프로세스를 통하여 악성코드의 탐지를 더욱 효율적으로 할 수 있을 것으로 사료된다.

주제어: 역공학, 분석프로세스, 악성코드, 자기방어기법, 패킹기법

A Novel Process Design for Analyzing Malicious Codes That Bypass Analysis Techniques

Lee, Kyung-Roul, Lee, Sun-Young, Yim, Kang-Bin

Abstract Malicious codes are currently becoming more complex and diversified, causing various problems spanning from simple information exposure to financial or psychologically critical damages. Even though many researches have studied using reverse engineering to detect these malicious codes, malicious code developers also utilize bypassing techniques against the code analysis to cause obscurity in code understanding. Furthermore, rootkit techniques are evolving to utilize such bypassing techniques, making it even more difficult to detect infection. Therefore, in this paper, we design the analysis process as a more agile countermeasure to malicious codes that bypass analysis techniques. The proposed analysis process is expected to be able to detect these malicious codes more efficiently.

Keywords: reverse engineering, analysis process, malicious code, self-protection technique, packing technique

2017년 11월 9일 접수, 2017년 11월 15일 심사, 2017년 12월 8일 게재확정

* 순천향대학교 보안안전융합기술사업화센터 연구교수(carpedm@sch.ac.kr)
** 순천향대학교 정보보호학과 교수(sunlee@sch.ac.kr)
*** 순천향대학교 정보보호학과 교수(yim@sch.ac.kr)

Ⅰ. 서론

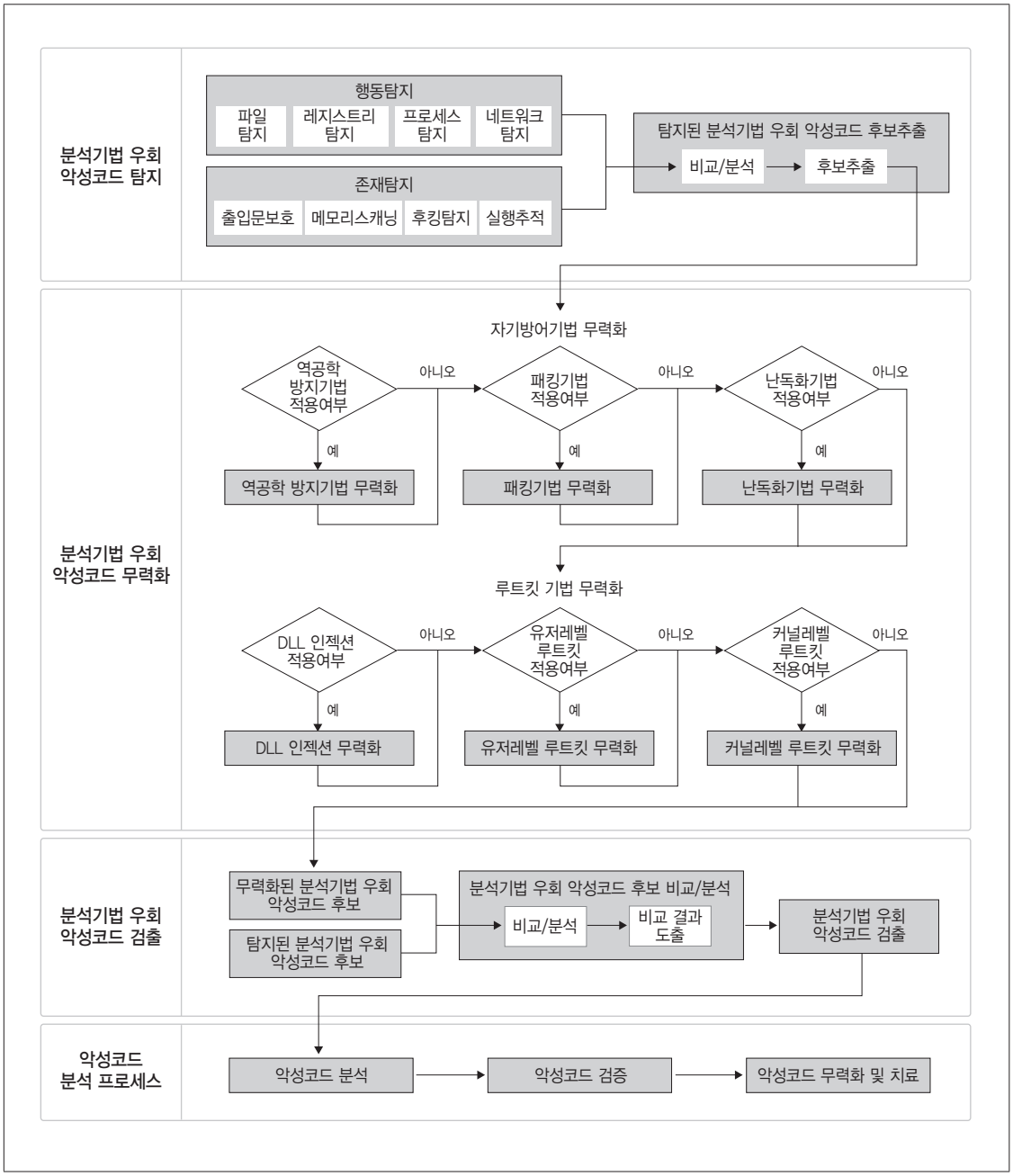
인터넷은 컴퓨터 산업을 혁신적으로 발전시켰고 인터넷의 전송속도는 나날이 빨라지고 있다. 그러나 이러한 긍정적인 측면과는 반대로 역기능 또한 증가하고 있다. 오랜 기간에 걸쳐 전파되던 과거 바이러스와 같은 악성코드들이 현재는 불과 몇 분 또는 몇 초 내로 전 세계 각지로 전파되고 있다(Bayer, et al., 2007). 과거에는 악성코드의 전파속도가 느리고 감염되는 경로가 한정적이었으며, 전파기술 또한 단순한 구조였기 때문에 이에 대응하는 방안을 연구하기에 충분한 시간적 여유가 있었다. 하지만 현재의 악성코드들은 은닉기술과 자가변형 등의 기술을 흡수하면서 매우 지능화되어, 최근 가장 문제가 되고 있는 분산서비스거부공격(DDoS)을 비롯한 스팸발송 및 개인정보 탈취 등의 문제를 야기하고 있다(김지연 외, 2009; 이경률 외, 2017; Yu, et al., 2011). 2011년 보고에 따르면 국내의 악성코드 감염 수와 유형은 지속적으로 증가하는 추세이다(안철수연구소, 2017).

악성코드에 대응하기 위해 보안전문가들은 허니넷 등을 이용하여 악성코드를 수집하고 역공학 등의 분석기술을 통해 수집된 코드를 분해/분석하고 있다(이동휘 외, 2005; Gregio, et al., 2011). 역공학을 이용하여 악성코드가 실행되는 각 단계를 추적하고 분석함으로써 발생 가능한 피해의 정도, 전파율, 제거방법과 전파를 방지하기 위한 대응방안을 마련할 수 있다(이승원, 2010a; 이승원, 2010b; 이승원, 2010c; 이태헌, 2016). 하지만 역공학은 보안 전문가만이 사용할 수 있는 기술이 아니라 악성코드 개발자도 활용할 수 있는 기술이다. 악성코드 개발자는 역공학을 통해 운영체제나 소프트웨어들의 취약점을 찾아내고 이를 이용하여 악성코드를 제작한다. 안티 디버깅, 루트킷 등의 기술들을 이용하여 보안 전문가의 역공학을 이용한 악성코드 분석을 우회함으로써 더욱 지능화되고 있다. 진화하는 악성코드에 대응하기 위한 연구가 진행되고 있으나, 유명 해킹 포럼 및 비공개 뉴스 그룹 등을 통해 악성코드 제작 도구가 유포되고 있어 악성코드 제작이 보다 쉬워지고 있다. 또한 분석회피 기술도 빠르게 진화하고 있으며, 이를 탐지하고 분석하기 어려운 문제도 존재한다. 국내에서는 '코드엔진 역공학 컨퍼런스(CodeEngn, 2017)', '해커스 드림 2010(안철수연구소, 2010)' 등을 개최하여 역공학의 활성화를 유도하고 있지만, 역공학을 통하여 악성코드를 분석하는 전문인력이 부족하여 분석기술의 발전 속도가 악성코드의 발전 속도를 앞서지 못하고 있다. 대부분의 악성코드는 자기 자신을 숨기기 위해 루트킷 기술(Woei-Jiunn, et al., 2009)을 이용하고 있어 보안전문가들이 악성코드를 탐지하기 위해서는 이들 기술의 분석이 필수적이다. 따라서 본 논문은 분석기법을 우회하는 악성코드를 분석하는 프로세스를 설계하는데 그 목적이 있다. 탐지된 악성코드의 행위를 분석하기 위해 악성코드의 검출, 분석, 검증, 무력화 과정을 통하여 시스템에 침투한, 혹은 침투하는 악성코드를 치료하는 것이 가능하다. 본 논문은 분석을 우회하는 기법이 적용된 악성코드를 분석하기 위한 프로세스를 설계하고 검증하며, 이를 위하여 악의적인 행위 자체를 분석하고 행위를 무력화하는 세부적인 과정은 배제하였다. 또한 이러한 악성코드는 역공학 방지 및 패킹, 난독화 기법을 사용하여 자신의 코드를 분석하기 못하도록 방해하므로 이러한 기법이 적용되더라도 악성코드를 분석하기 위한 체계적인 방안이 필요하며, 루트킷 기법을 이용하여 우회하기도 하므로 유저레벨 및 커널레벨, 혹은 복합적으로 구성된 루트킷을 분석하기 위한 절차가 요구된다. 따라서 상기 분석기법을 우회하는 기술들의 탐지 및 분석, 무력화는 악의적인 행위를 분석하기 위한 준비단계로서 일련의 과정으로 정의될 수 있으므로 이 과정을 분석기법을 우회하는 악성코드를 분석하기 위한 프로세스로 설계하였다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 본 논문에서 설계한 분석기법을 우회하는 악성코드의 분석 프로세스에 대해 서술하고, 제 Ⅲ장에서는 설계한 분석 프로세스의 세부 단계에 대해 기술하며, 제 Ⅳ장에 결론을 논한다.

Ⅱ. 분석기법을 우회하는 악성코드를 분석하기 위한 프로세스 설계

본 논문에서는 〈그림 1〉과 같이 분석기법을 우회하는 악성코드 분석 프로세스를 설계하였다.



〈그림 1〉 분석기법 우회 악성코드 분석 프로세스

분석기법을 우회하는 악성코드를 탐지하기 위한 단계에서는 행동탐지, 존재탐지를 통하여 악성코드 후보를 추출한다. 행동탐지에서는 파일탐지, 레지스트리탐지, 프로세스탐지, 네트워크탐지를 수행하며, 존재탐지에서는 출입문보호, 메모리 스캐닝, 후킹탐지, 실행추적을 수행한다. 이들 탐지과정을 통하여 악성코드의 존재와 악의적인 행위를 탐지하며, 탐지 결과의 비교 및 분석을 통하여 악성코드 후보들을 추출한다. 이후, 무력화단계, 검출단계를 적용하여 추출된 악성코드를 확보하고, 그 결과를 토대로 악성코드 분석 프로세스 단계를 수행하여 검출한 악성코드의 실제 악의적인 행위에 대한 세밀한 분석을 수행한다.

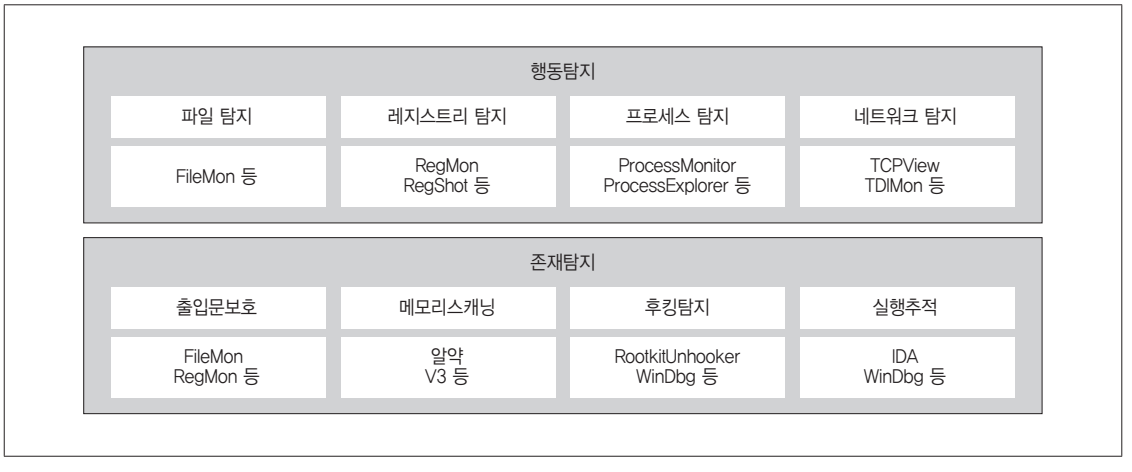
무력화단계에서는 탐지단계에서 추출한 후보 악성코드를 무력화하여 악성코드 검출을 위한 사전준비를 수행한다. 이 단계는 자기방어기법 무력화와 루트킷기법 무력화단계로 구성된다. 자기방어기법 무력화단계에서는 악성코드가 코드분석을 방해하기 위해 사용하는 역공학 방지기법 및 패킹기법, 난독화기법과 같은 자기방어기법의 적용여부를 판단한다. 자기방어기법 무력화단계가 완료되면, 유저레벨 루트킷 및 커널레벨 루트킷, DLL 인젝션으로 분류된 루트킷기법 무력화단계를 수행한다. 악성코드의 대부분이 상기의 기술로 분석기법을 우회하므로 무력화를 통하여 추출된 후보 악성코드의 검출이 가능하다.

검출단계에서는 상기 단계, 즉, 탐지단계와 무력화단계의 결과들을 비교하여 최종적으로 악성코드를 검출한다. 검출된 악성코드는 실제 악의적인 기능들의 분석을 위해 활용되며, 악성코드 분석단계, 검증단계, 무력화 및 치료단계를 거쳐 최종적으로 악성코드 분석 프로세스를 완료한다.

Ⅲ. 제안한 분석 프로세스 세부과정

1. 분석기법 우회 악성코드 탐지단계

악성코드를 탐지하고 분석하기 위해서는 상황에 맞는 환경적 요소 구축이 필수적이다. PC가 악성코드에 의해 감염된 상태라면, 감염 후 발생하는 악의적인 행위를 수행하지 못하도록 네트워크 및 시스템의 환경을 구성하여야 하며, 이미 추출된 악성코드일 경우에는 효과적으로 분석할 수 있는 환경을 구축하여야 한다. 이러한 환경으로는 가상머신, 허니팟 등이 있다. 특히 가상머신의 경우, 환경을 구성하고 시스템을 감시하기가 용이하며, 실제 시스템으로부터 논리적으로 분리되어



〈그림 2〉 분석기법 우회 악성코드 탐지 단계

시스템에 직접적으로 영향을 미치지 않아 감염 전/후의 비교 및 복구를 빠르게 수행하는 장점이 있다. 그림 2와 같이 행동탐지 과정에서는 파일 및 레지스트리, 프로세스, 네트워크로부터 악성코드의 행동을 탐지하기 위하여 각 도구를 이용하며, 존재탐지 과정에서는 출입문보호 및 메모리 스캐닝, 후킹탐지, 실행추적을 위한 수동적 분석 및 가능한 도구를 활용한다.

1) 행동탐지 단계

행동탐지는 행위를 기반으로 악성코드를 탐지하는 것을 의미하며, 운영체제를 속이는 것을 찾아내는데 그 목적이 있다. 예를 들어, 의도적으로 올바르지 않은 값을 반환하는 API와 같은 호출이 존재한다면, 이는 악성코드의 존재를 확인할 수 있을 뿐 아니라 숨기고자 하는 목적도 파악할 수 있다.

⑴ 파일탐지

악성코드는 시스템을 감염시키기 위해 특정한 패턴을 가지는 이름 또는 전혀 다른 이름의 파일을 생성하며, 생성한 파일을 이용하여 자신을 위장하거나 은닉한다. 또한 유저레벨에서의 후킹이나 커널레벨에서의 후킹, 디바이스 드라이버를 악용하여 은닉이 가능하다. 일반적인 경우에는 파일을 은닉하는 과정은 필요하지 않기 때문에 은닉행위 자체가 악성코드의 후보라는 것을 의미하므로 파일의 생성, 수정 및 삭제동작을 감시하여 악성코드 후보를 추출한다. 파일탐지를 위한 도구로는 FileMon, Explorer 등이 있다(한국인터넷진흥원, 2009 ; 한국인터넷진흥원, 2010).

⑵ 레지스트리탐지

윈도우즈 운영체제에서는 시스템의 설정을 저장하고 관리하는데 레지스트리를 이용한다. 윈도우즈가 부팅되면서 사용자 로그인, 응용 프로그램의 실행과 관리 등의 모든 작업들이 레지스트리를 통하여 이루어진다. 따라서 악성코드에 감염된 시스템은 악의적인 행위를 수행하기 위하여 레지스트리의 추가 및 수정, 삭제 행위를 수행한다. 악성코드의 경우에는 자신이 반드시 실행되어야 하므로 부팅 시 자동으로 실행되도록 설정하는 경우가 많으며, 이러한 설정은 안티 바이러스 프로그램의 탐지를 우회하기 위해 사용되기도 한다. 이러한 이유로 악성코드가 악의적인 행위를 위하여 주로 활용하는 레지스트리를 조사하고 분석하여 악성코드를 탐지하고 추출할 수 있으며, 이를 위한 도구로는 RegMon, RegShot, Regedit 등이 있다. 또한 RegOpenKeyEx나 RegQueryValueEx와 같은 API를 이용하여 획득한 레지스트리 키와 데이터, 레지스트리 정보를 수집한 레지스트리 하이브파일을 분석하여 얻은 결과와의 비교를 통해서도 악성코드 탐지가 가능하다(한국인터넷진흥원, 2009; 한국인터넷진흥원, 2010).

⑶ 프로세스탐지

윈도우즈 운영체제는 프로세스를 생성하여 응용 프로그램을 실행하고 관리한다. 이에 공격자는 새로운 프로세스를 생성하여 시스템을 감염시킬 수 있기 때문에 이를 감시함으로써 악성코드를 탐지한다. 악성코드의 분석기법 우회 기술 중에는 프로세스와 파일을 숨기는 기법이 존재하며, 숨겨진 정보를 확인하지 못하므로 위협적이다. 숨겨진 프로세스를 탐지하는 방법에는 Swap Context 후킹방법이 있으며, ntoskrnl 모듈의 Swap Context 함수는 컨텍스트가 교체될 때 호출되는 특징이 있다. 이러한 이유로 SwapContext 함수를 후킹하여 다음 스레드가 프로세스 리스트에 존재하는지 확인함으로써 악성코드를 탐지한다. 따라서 Swap Context 함수를 후킹하여 수집한 프로세스 리스트와 NtQuery SystemInformation과 같은 API로 수집한 프로세스 리스트를 비교한다면, 숨겨진 프로세스를 탐지하는 것이 가능하다(한국인터넷진흥원, 2009; 한국인터넷진흥원, 2010; Hoglund & Butler, 2005; Russinovich, et al., 2009).

⑷ 네트워크탐지

악성코드는 악의적인 행위에 대한 명령 전송, 코드

변형 과정, 탈취한 정보의 전송, 공격자와의 연결유지 등의 기능을 수행하기 위해 네트워크에 연결된다. 시스템을 장악한 악성코드는 데이터 유출, 백도어 등의 기능을 수행하기 위해 특정 네트워크로 접속하므로 이를 탐지하기 위해서는 시스템에서 네트워크를 감시한 후, 그 결과를 분석함으로써 악성코드의 특징을 파악한다. 네트워크 감시는 TCPView, TDIMon 등의 도구를 통하여 감시한다.

2) 존재탐지 단계

존재탐지는 악성코드의 존재 여부를 탐지하는 방식이다. 과거에는 Tripwire와 같은 도구로 악성코드 이미지를 검색하는 기술이 사용되었지만, 이 기술은 악성코드가 파일시스템을 사용한다는 전제 하에 사용 가능하며, 파일을 은닉하기 위한 목적으로 하드웨어의 특정 부분, 혹은 메모리에만 존재하도록 구성하거나 필터 드라이버로 은닉한다면 탐지가 불가능한 한계가 존재한다. 이로 인해 악성코드의 존재여부를 판단하기 위한 여러 가지 방법들이 제안되었으며, 출입문보호, 메모리 스캐닝, 후킹탐지, 실행추적 등의 방법이 있다.

(1) 출입문보호

악성코드의 존재를 탐지하는 방법은 메모리에서 악성코드를 탐지하는 방법과 존재증거를 탐지하는 방법이 있다. 메모리에서 악성코드를 탐지하는 방법은 악성코드가 실행을 위하여 자신의 코드를 메모리로 로드하는 것을 탐지하는 방법과 메모리를 주기적으로 스캔함으로써 탐지하는 방법으로 나누어진다. 메모리로 로드하는 것을 탐지하는 방법은 악성코드가 실행을 위하여 반드시 메모리에 존재하여야만 하므로 메모리로 로드되는 진입점을 감시함으로써 악성코드를 탐지한다. 하지만 이와 같은 경우에는 그 결과가 복합적으로 드러나기 때문에 FileMon, RegMon, WinDbg 등과 같은 다양한 도구를 활용하여 파악해야 하는 단점이 존재한다(한국인터넷진흥원, 2010; Hoglund & Butler, 2005).

(2) 메모리 스캐닝

메모리 스캐닝은 메모리에서 악성코드를 검색하기 위한 기술이다. 상기 출입문보호 기술은 악성코드가 메모리로 로드되는 것을 지속적으로 검사해야만 하는 단점이 존재하므로, 더욱 효율적으로 탐지하기 위하여 주기적으로 메모리를 스캔하여 악성코드를 탐지한다. 하지만 악성코드가 메모리로 로드되는 것은 방지하지 못하며, 메모리에 로드된 악성코드만 탐지할 수 있는 단점이 존재한다. 메모리 스캐닝에 사용되는 도구로는 안티 바이러스 프로그램들이 있으며, 대표적으로 알약, V3 등이 있다(한국인터넷진흥원, 2010; Hoglund & Butler, 2005).

(3) 후킹탐지

후킹탐지는 운영체제나 프로세스에 대해 탐지한다. 이는 상기 메모리 스캐닝 방법과는 달리 후킹검색을 수행할 때 악성코드 탐지를 위한 알려진 패턴이 요구되지 않는 장점이 존재한다. 후킹탐지 알고리즘은 프로세스에서 허용된 범위 외부로의 실행분기를 찾아내는 것이다. 다시 말하면, 프로세스는 자신의 코드를 실행하기 위한 허용된 범위를 가지며, 악의적인 목적의 코드를 실행하기 위해서는 해당 범위를 벗어난 메모리에 코드를 위치시켜야 하므로 범위를 벗어난 코드로 분기하거나 참조하는 경우에는 후킹된 것으로 판단이 가능하다. 악성코드가 주로 활용하는 후킹기술로는 IAT 후킹, SSDI 후킹, IDT 후킹, 인라인 후킹이 있으며, Rootkit Unhooker, IceSword, WinDbg 등의 도구를 통하여 후킹을 탐지하는 것이 가능하다(한국인터넷진흥원, 2010; Hoglund & Butler, 2005).

(4) 실행추적

후킹탐지와는 다르게 후킹을 검사하는 방법으로 함수의 호출흐름을 추적하는 방법이 있다. 일단 후킹이 시작되면 기존 코드보다 더 많은 명령어를 실행한다. 따라서 후킹되지 않은 상태의 함수들에 대한 정상적인 수치를 수집한 후, 대상 함수를 주기적으로 호출하여

해당 함수가 실행되면서 추가적으로 실행되는 코드가 존재하는지 산정한 기준치와 비교함으로써 후킹여부를 판단한다. 하지만 이 방법은 무결한 기준치가 필요하며, 후킹되지 않은 상태에서도 실행되는 코드의 개수가 일정하지 않은 단점이 있다.

3) 탐지한 분석기법 우회 악성코드 후보 추출 단계

악성코드 후보를 추출하기 위해서는 행동탐지 단계를 통해 탐지한 악성코드와 존재탐지 단계를 통해 탐지한 악성코드의 비교를 통하여 실제 악성코드인지 정상적인 응용 프로그램인지 판단해야 한다. 즉, 행동탐지와 존재탐지 결과를 비교함으로써 분석기법을 우회하는 악성코드 후보들을 추출한다.

2. 분석기법 우회 악성코드 무력화 단계

무력화 단계는 상기의 탐지단계에서 추출한 후보 악성코드를 검출하기 위해 자기방어기법, 루트킷 기법과 같은 악성코드가 사용하는 분석기법 우회방법을 무력화하여 검출 단계에 필요한 사전준비를 한다. 이 단계는 이전 단계에서 추출한 후보 악성코드만을 대상으로 하기 때문에 효과적으로 분석하고 무력화할 수 있지만 많은 인력과 시간이 요구되는 단점도 있다.

1) 자기방어 기법 무력화 단계

악성코드는 자신의 코드가 역공학을 통하여 분석되는 것을 방해하기 위하여 자기방어기법을 이용한다. 이러한 자기방어기법으로는 역공학 방지, 패킹, 난독화 기법이 있으며, 단독이나 조합을 통하여 분석이 어렵도록 방해한다. 따라서 각 기법들을 무력화해야 하며, 그 과정을 〈그림 3〉에 나타내었다.

(1) 역공학 방지기법

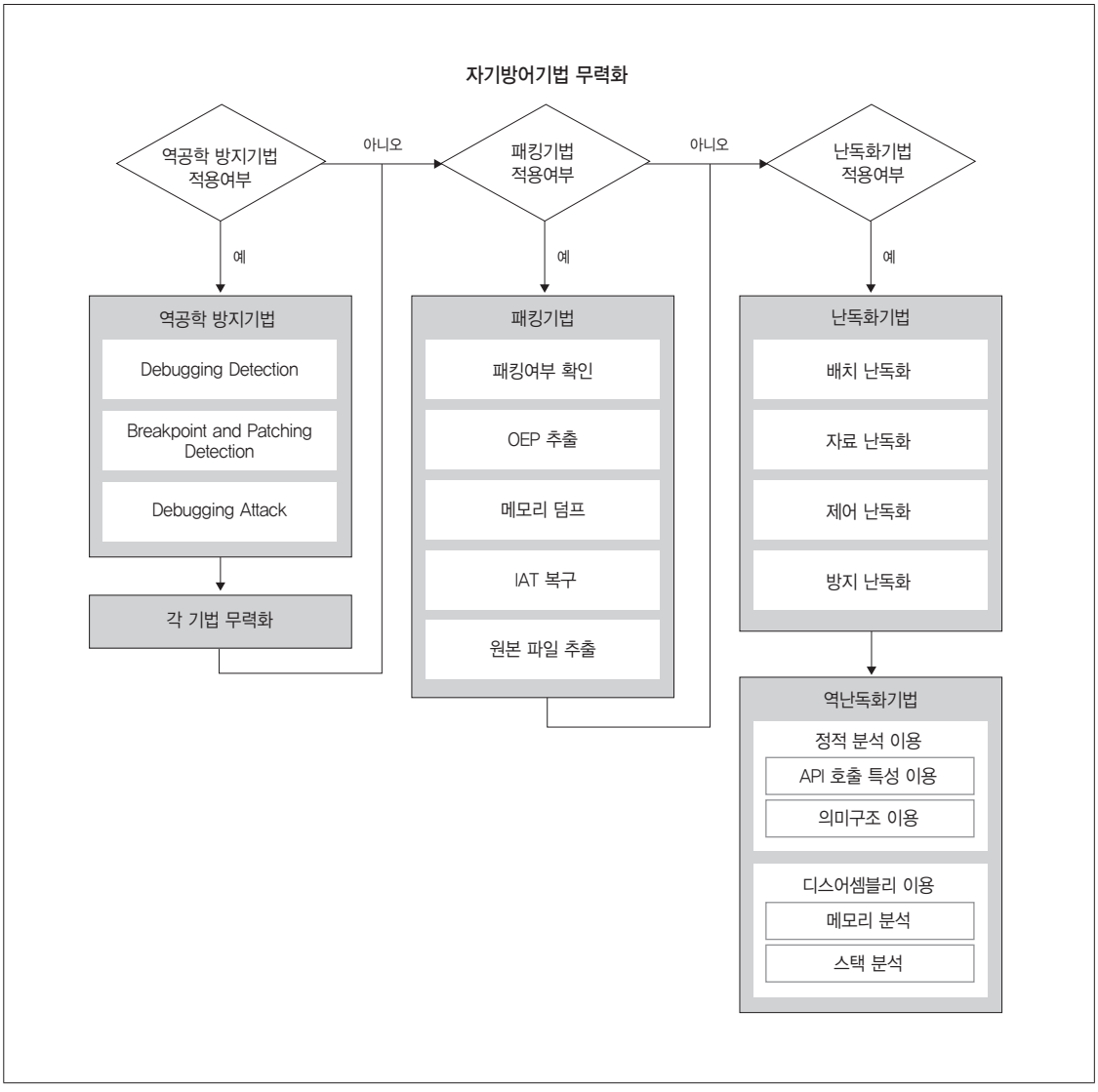
공격자는 분석기법을 우회하기 위하여 내부에서 디버깅 수행여부를 검사한 후, 만약 디버깅되는 경우에는 악의적인 행위를 수행하지 않도록 구성하거나 다양한 기법을 적용하여 디버깅을 방해한다. 이와 같은 경우, 악의적인 행위가 존재하는지 판단하기 이전에 코드분석을 방해하기 때문에 무력화 단계가 반드시 필요하다. 역공학 방지기법은 디버깅 탐지, 중지점 및 패칭 탐지, 디버거 공격으로 분류되며, 각 기법들의 적용여부를 판단하여 무력화해야 한다(Yason, 2007; 이찬희 외, 2013; 이경률 · 임강빈, 2012; 홍수화, 2016).

(2) 패킹기법

과거 네트워크로의 전송에서 파일용량을 최소화하는데 이용되었던 패킹기법들이 암호화, 코드 재배치, 역공학 방지기법 등의 기술들과 결합되면서 코드분석을 더욱 어렵게 하도록 진화되었다. 하지만 악성코드도 자신의 코드를 분석하기 어렵도록 이러한 기법들을 활용하기 시작하였다. 따라서 패킹된 악성코드의 경우에는 무력화 방안인 언패킹을 수행하여 원본 파일을 추출할 수 있으며, 추출된 파일이 난독화된 경우에는 역난독화를 수행한다. 이를 통하여 확보한 파일은 루트킷기법 무력화단계에서 사용 가능하다(Lee, et al., 2010). 하지만 상기 언패킹에서도 한계점이 존재한다. 일부 악성코드에서는 암호화 등의 패킹기술을 사용하기도 하며, 이러한 패킹기술은 정적 분석에서 패킹에 활용된 코드를 디스어셈블하거나 분석하는 것이 어려우며, 이로 인하여 주로 동적 분석으로 해당 코드를 분석한다. 하지만 동적분석에서는 코드 적용범위 (Code Coverage)의 한계가 존재하여 분석이 어려운 경우도 존재한다.

(3) 난독화기법

소프트웨어 저작권을 위한 목적으로 이용되었던 난독화 기법은, 적용위치에 따라 소스레벨과 바이너리레벨 난독화, 적용방법에 따라 배치, 자료, 제어, 방지 난독화로 분류된다. 난독화는 코드 분석을 어렵게 하기 위해 코드를 변형하는 것을 말한다. 따라서 난독화가 적용된 악성코드를 효과적으로 분석하기 위한 무력화 방안이 필요하며, 이를 역난독화라 부른다. 역난독화는 정적분석 및 디스어셈블리를 이용한 방법으로 분류



〈그림 3〉 분석기법 우회 악성코드 무력화단계 – 자기방어기법 무력화단계

되며, 정적분석을 이용한 방법은 API 호출특성, 의미구조를 이용한 방법으로 나누어진다. 디스어셈블리를 이용한 방법으로는 메모리 분석, 스택분석을 이용한 방법이 있다. 이러한 방법들을 이용하여 역난독화를 수행하면 더욱 효과적으로 코드를 분석할 수 있다(국가보안기술연구소, 2005; 김정일 · 이은주, 2011; Balakrishnan & Schulze, 2005; You & Yim, 2010; 정진혁 · 이정현, 2013; 석재혁 외, 2013; 이재휘 외, 2017). 하지만 난독화된 악성코드의 경우에는 분석이 어렵도록 방해하는 기술이 적용될 수 있으며, 이러한 악성코드는 역난독화를 하더라도 동적으로 실행되지 않는 코드가 존재하여 분석에 한계가 있을 수 있다.

2) 루트킷기법 무력화 단계

선행 조사의 결과를 기반으로 악성코드가 은닉을 위하여 활용하는 루트킷기법을 무력화하는 단계를 수행한다. 유저레벨과 커널레벨로 나누어진 이 단계는 각 단계를 순차적, 혹은 역으로 분석하거나 그 반대, 또는 두 가지 모두를 분석한다. 분석을 위한 탐지방법은 DLL 인젝션 탐지, 유저레벨 루트킷탐지, 커널레벨 루트킷탐지 방법으로 분류된다.

(1) DLL 인젝션

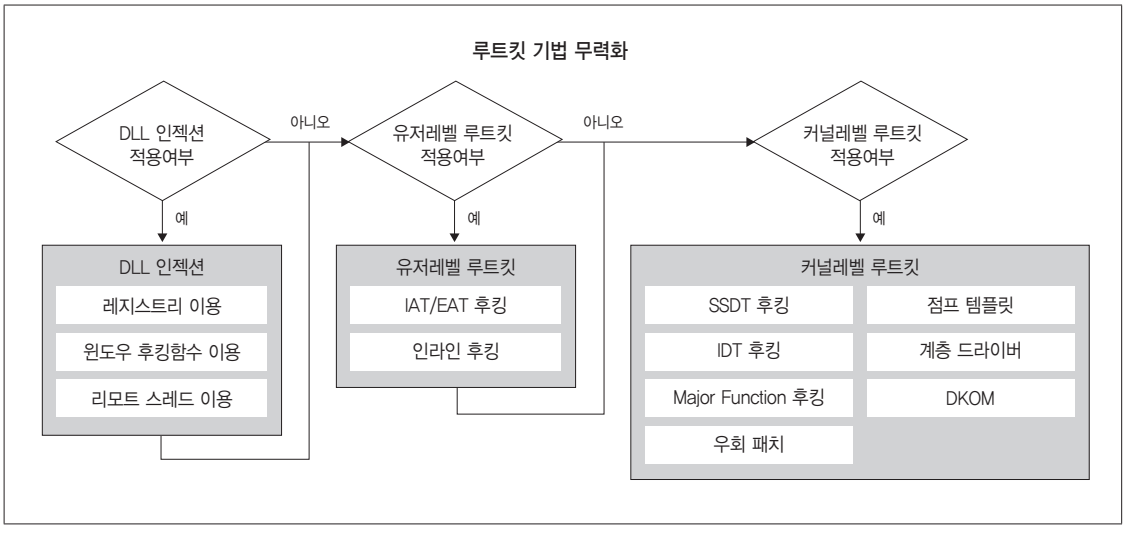
악성코드가 유저레벨에서 자신을 은닉하기 위하여 후킹을 수행하며, 이는 주로 DLL 인젝션을 통하여 이루어진다. 이를 탐지하기 위하여 현재 시스템의 DLL 상태를 분석하여 무력화해야 한다. 또한 가능하다면 추가적인 인젝션 코드를 분석함으로써 차후 악성코드 분석 프로세스를 수행할 때 필요한 정보로 활용할 수 있다 (이승원, 2010a; 이승원, 2010c; Hoglund & Butler, 2005).

(2) 유저레벨 루트킷

유저레벨 루트킷기법은 주로 API 후킹으로 그 목적을 달성하며, API 후킹은 IAT/EAT 후킹, 인라인 후킹으로 분류된다. 이들 기법은 별도로 실행 중인 악성코드 내부에서 사용되거나 상기의 DLL 인젝션으로 삽입된 후 실행되기도 한다. 이러한 이유로 DLL 인젝션을 탐지하여 무력화한 후, 확보된 코드를 기반으로 유저제벨 루트킷 기법인 IAT/EAT 후킹과 인라인 후킹을 분석하고 무력화해야 한다 (이승원, 2010b; Hoglund & Butler, 2005).

(3) 커널레벨 루트킷

유저레벨보다 낮은 레벨인 커널레벨에서는 유저레벨에서 확보할 수 있는 정보보다 많은 정보를 확보하므로 커널레벨에서 유저레벨 루트킷을 더욱 쉽게 탐지한다. 따라서 악성코드 제작자는 이러한 한계점을 극복하기 위해 커널레벨에서 동작하는 악성코드를 제작하기 시작하였다. 그 결과, 동일한 레벨에서 다양한 기능을 활용한 공격으로 그 대응이 어려운 문제점이 존재한다. 이러한 공격기법으로는 SSDT 후킹, IDT 후킹, Major function 후킹, 우회 패치, 점프 템플릿, 계층 드라이버, DKOM이 있다 (Hoglund & Butler, 2005; Woei-Jiunn, et al., 2009; Kim, et al., 2012).



〈그림 4〉 분석기법 우회 악성코드 무력화단계 – 루트킷기법 무력화단계

3. 분석기법 우회 악성코드 검출 단계

검출단계에서는 상기 탐지단계에서 추출한 후보 악성코드와 무력화단계에서 확보한 결과를 비교함으로써 악성코드를 검출한다. 악성코드가 사용하는 기법은 악성코드만이 사용하는 것은 아니므로 비교/분석을 통해 악성코드인지 정상적인 프로그램인지를 판단할 필요성이 있다. 비교/분석 결과가 악성코드일 경우에는 탐지 및 무력화단계에서 분석된 내용을 기반으로 이를 검출하여 악성코드 분석 프로세스에 적용한다. 또한 탐지단계와 무력화단계에서 조사한 분석기법 우회기술을 정리하고 이를 분류하여 분석기법 우회기술의 난이도나 기준을 설정하여 악성코드 분석 프로세스의 자료로 활용한다.

Ⅳ. 결론

본 논문은 분석기법을 우회하는 악성코드에 보다 재빠르게 대응하기 위한 분석 프로세스를 설계하였다. 본 설계에서는 탐지단계, 무력화단계, 검출단계를 거쳐 악의적인 행위의 분석을 위한 악성코드를 확보한다. 탐지단계는 행동탐지, 존재탐지 과정을 통해 악성코드 후보들을 추출하며, 무력화단계에서는 역공학 방지기법, 패킹기법, 난독화기법으로 이루어진 자기방어기법을 무력화하고, DLL 인젝션, 유저/커널레벨 루트킷으로 이루어진 루트킷기법을 무력화하여 악성코드를 추출한다. 검출단계에서는 탐지단계와 무력화단계를 통해 추출된 후보 악성코드들의 분석결과를 비교하여 최종적인 판단을 함으로써 악성코드를 검출한다. 검출한 악성코드는 별도의 악성코드 분석 프로세스를 통해 세부적인 동작을 분석하여 방지 및 치료를 위해 활용하며, 본 논문에서 제안한 분석 프로세스를 통해 분석기법을 우회하는 악성코드에 대한 대응을 민첩하게 할 수 있을 것으로 판단된다.

그럼에도 불구하고 본 본문에서 설계한 프로세스를 실제 시스템에 적용할 경우에는 그 환경적인 영향으로 인하여 문제점이 발생할 가능성이 존재한다. 그 이유는 설계한 프로세스는 분석기법을 우회하는 악성코드를 확보하기 위하여 기술적인 절차를 토대로 설계하였기 때문에 실제 시스템에 존재하는 다양한 환경적인 요소에 대응하지 못하는 부분이 있을 수 있다. 그 일례로 운영체제의 종류에 따라 DLL 인젝션과 같은 기술이 무력화되지 못하도록 방지하는 보안기술이 적용된 경우가 있을 수 있으며, 자기방어기법 및 루트킷 기법을 무력화하기 위하여 활용되는 코드가 정상적이지 않은 방법으로 접근할 경우에는 백신 프로그램이 악성코드로 탐지하거나 충돌이 발생하여 무력화하지 못하는 경우도 있을 수 있다. 이러한 문제점들은 이미 경험한 전문가로부터 자문을 구하여 극복할 수 있으며, 백신 프로그램의 경우에는 분석가들의 목적이 악성코드를 확보하는 것이기 때문에 프로그램을 동작시키지 않도록 하거나 종료시키는 방법으로 일시적으로 우회하여 해결할 수 있을 것으로 사료된다.

■ 참고문헌

국가보안기술연구소 (2005). 「코드 난독화를 이용한 악성코드 분석 기법에 관한 연구」. 대전: 국가보안기술연구소.

김정일 · 이은주 (2011). “제어 흐름 난독화를 효과적으로 수행하기 위한 전략.” 「한국컴퓨터정보학회 논문지」, 16(6): 41-50.

김지연 · 이주리 · 박은지 · 장은영 · 김형종 (2009). “DDoS 공격 피해 규모 및 대응기법 비용분석을 위한 모델링 및 시뮬레이션 기술 연구.”, 「한국시뮬레이션학회 논문지」, 18(4): 39-47.

석재혁 · 김성훈 · 이동훈 (2013). “가상화 난독화 기법이 적용된 실행 파일 분석 및 자동화 분석 도구 구현.”, 「한국정보보호학회논문지」, 23(4): 709-720.

안철수연구소 (2010). “해커스 드림 2010.” http://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?cmd=print&seq=17027&menu_dist=1 (Retrieved on November 9, 2017).

안철수연구소 (2017). “월간보안보고서.” http://www.ahn

lab.com/kr/site/securityinfo/asec/asecReportView.do?groupCode=VNI001 (Retrieved on November 9, 2017).
이경률 · 임강빈 (2012). "코드은닉을 이용한 역공학 방지 악성코드 분석방법 연구." 「한국항행학회논문지」, 16(3): 488-494.
이경률 · 이선영 · 임강빈 (2017). "인터넷 뱅킹 서비스에서의 보안위협 분류 및 분석." 「정보화정책 저널」, 24(2): 20-42.
이동휘 · 이상호 · 김귀남 (2005). "허니넷 기반의 사이버위협 조기탐지기법 연구." 「한국사이버테러정보전학회 정보보증논문지」, 5(4): 67-72.
이승원 (2010a). "리버싱 관점에서 애플리케이션의 기능 추가 방법 : DLL Loading by PE Patch." 「마이크로소프트웨어」, 319: 204-211.
이승원 (2010b). "리버싱 관점에서 애플리케이션의 기능 추가 방법, 2 : Inline Code Patch." 「마이크로소프트웨어」, 320: 200-205.
이승원 (2010c). "리버싱 관점에서 애플리케이션의 기능 추가 방법, 3 : DLL Injection." 「마이크로소프트웨어」, 321: 200-207.
이재휘 · 한재혁 · 이민욱 · 최재문 · 백현우 · 이상진 (2017). "Themida의 API 난독화 분석과 복구방안 연구." 「한국정보보호학회논문지」, 27(1): 67-77.
이찬희 · 정윤식 · 조성제 (2013). "안드로이드 애플리케이션에 대한 역공학 방지 기법." 「보안공학연구논문지」, 10(1): 41-50.
이태헌 (2016). "텍스트 마이닝을 이용한 정보보호인식 분석 및 강화 방안 모색." 「정보화정책」, 23(4): 76- 94.
정진혁 · 이정현 (2013). "프로가드 난독화 도구 구조 및 기능 분석." 「한국통신학회논문지」, 38B(8): 654-662.
한국인터넷진흥원 (2009). 「악성코드 유형에 따른 자동화 분석 방법론 연구」. 서울: 한국인터넷진흥원.
한국인터넷진흥원 (2010). 「분석기법 우회 악성코드 분석방법 연구」. 서울: 한국인터넷진흥원.
홍수화 (2016). "Pin을 이용한 안티디버깅 우회 설계 및 구현" 「인터넷정보학회논문지」, 17(5): 33-42.
Bayer, U., Habibi, I., Balzarotti, D., Kirda, E. & Kruegel, C. (2009). "A View on Current Malware Behaviors." *Usenix Workshop on Large-scale Exploits and Emergent Threats(LEET)*, http://static.usenix.org/event/leet09/tech/full_papers/bayer/bayer_html/ (Retrieved on November 9, 2017).
Balakrishnan, A. & Schulze C. (2005). "Code Obfuscation Literature Survey," http://pages.cs.wisc.edu/~arinib/writeup.pdf (Retrieved on November 9, 2017).
CodeEngn (2017). "CodeEngn Conference.", http://www.codeengn.com (Retrieved on November 9, 2017).
Gregio, A.R.A., Filho, D.S.F., Afonso, V.M., Santos, R.D.C., Jino, M. & de Geus, P.L. (2011). "Behavioral analysis of malicious code through network traffic and system call monitoring." *Proceedings of the SPIE*.
Hoglund, G. & Butler, J. (2005). *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional.
Kim, S., Park, J., Lee, K., You, I. & Yim, K. (2012). "A Brief Survey on Rootkit Techniques in Malicious Codes." *The journal of Internet Services and Information Security*, 2(3/4): 134-147.
Lee, K., You, I. & Yim, K. (2010). "A Hint to the Analysis of the Packed Malicious Codes," *Proceedings of the WISA*.
Russinovich, M. & Solomon, D. (2009). *Windows Internals*. Microsoft.
Woei-Jiunn T., Yuh-Chen C. & Being-Yu T. (2009). "A New Windows Driver-Hidden Rootkit Based on Direct Kernel Object Manipulation." *The journal of Lecture notes in computer science*, 5574: 202-213.
Yason, M. (2007). "The Art of Unpacking," https://www.blackhat.com/presentations/bh-usa-07/Yason/Whitepaper/bh-usa-07-yason-WP.pdf (Retrieved on November 9, 2017).
You, I. & Yim, K. (2010). "Malware Obfuscation Techniques: A Brief Survey," *International Conference on Broadband Wireless Computing, Communication and Applications*, 297-300.
Yu, S., Zhou, W., Doss, R. & Jia, W. (2011). "Traceback of DDoS Attacks Using Entropy Variations." *IEEE transactions on parallel and distributed systems; a publication of the IEEE Computer Society*, 22(3): 412-425.