Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.2, Apr. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.2.175

프로토콜 역공학 연구 동향 및 최적의 악성코드 통신 메시지 분석환경 구축 연구*

신 강 식,[1†] 정 동 재,[2] 최 민 지,[1] 조 호 묵[3‡]
[1,2,3]KAIST 사이버보안연구센터 (연구원, 선임연구원, 책임연구원)

A Study Protocol Reverse Engineering Research Trend and Construction of Optimal Environment for Malware Analysis*

Kangsik Shin,[1†] Dong-Jae Jung,[2] Min-Ji Choe,[1] Ho-Mook Cho[3‡]
[1,2,3]KAIST Cyber Security Research Center (General, Senior, Principal Researcher)

요 약

최근 복잡하고 지능화된 다양한 악성코드가 지속해서 출현하면서 악성코드와 명령을 주고받는 C&C(Command & Control) 서버도 증가하였다. 봇넷과 C&C 서버의 통신 프로토콜 분석은 봇넷을 깊이 있게 이해하고, 방어하는 데 필수적이다. 이러한 비공개 통신 프로토콜을 분석하기 위한 통신 프로토콜 역공학 분석 기법은 네트워크 기반의 분석 방법과 실행 기반의 분석 방법으로 구분할 수 있다. 본 논문에서는 각 기법의 연구 동향을 파악하고, 더욱 향상된 성능을 위해, Anti-VM 대응 환경, 가상의 C&C 서버 역할을 할 수 있는 Fakenet-ng와 Pintool 기반의 동적 분석 도구인 Protocol Tracer를 개발하여 네트워크 기반과 실행 기반의 분석 방법을 혼합한 하이브리드 분석 방식의 분석환경을 구성하였고, 실험을 통해 도출한 분석환경으로부터 기존환경의 데이터보다 양적 및 질적으로 향상된 결과를 확인하였다.

ABSTRACT

With the advent of a variety of complex and intelligent malicious code these days, the number of C&C (command and control) servers exchanging commands with malicious code is also increasing. Analysis of communication protocols between botnets and C & C servers is essential for a deeper understanding and defense of botnets. The communication protocol reverse engineering analysis method for analyzing such a closed communication protocol can be divided into a network-based analysis method and an execution-based analysis method. In this paper, we have developed a protocol tracer, a dynamic analysis tool based on Fakenet-ng and Pintool, to understand the research trends of each method and further improve its performance. It acts as an Anti-VM responsive environment and virtual. C&C server. We constructed a hybrid analysis environment that combines analysis methods and execution, and improved the results quantitatively and qualitatively from the analysis environment obtained by the experiment compared to the data of the existing environment.

Keywords: malware, command & control, dynamic analysis, reverse engineering , pintool

Received(03. 15. 2021), Accepted(04. 01. 2021)
* 본 논문은 과학기술정보통신부 글로벌사이버보안기술연구(과제 고유번호: 1711125408) 사업의 지원을 받아 수행된 연구임.

† 주저자, ksshin90@kaist.ac.kr
‡ 교신저자, chmook79@kaist.ac.kr(Corresponding author)

Ⅰ. 서 론

최근 악성코드는 과거에 비하여 매우 복잡한 매커니즘으로 구성 및 동작되며, 이를 위해 공격자는 C&C 서버를 이용하여 악성 행위를 원격으로 수행하거나 원하는 시간에 악성코드가 활성화될 수 있도록 명령을 전달한다[1, 2, 5]. 2018년 카스퍼스키 조사에 따르면 전 세계 C&C 서버 중 우리나라가 30.92%로 가장 많이 보유하고 있는 것으로 나타나고 있으며, 지속적으로 증가하는 추이를 보인다[3]. 현재 출현하는 악성코드는 복합적인 행위로 인해 명확하게 분류될 수 없지만, 행위를 기준으로 크게 9개의 타입으로 구분할 수 있다[6]. 9개의 타입은 <Table 1>과 같이 원격 코드 실행을 하는 트로이목마, 사용자 정보를 수집하는 스파이웨어, 사용자 PC를 마음대로 조작할 수 있도록 감염되게 하는 봇 등이 있으며, 대부분의 악성코드는 C&C 서버와의 통신을 통해 악의적인 행위와 공격자의 특정 명령을 수행한다.

따라서, 악성코드의 확산을 차단하고 정보 유출을 선제적으로 방지하기 위해 C&C 서버를 효과적으로 탐지 및 차단하는 것은 매우 중요하다. 이를 위해 국외에서는 통신 메시지 분석을 위한 프로토콜 역공학 연구가 활발히 진행되고 있으나, 국내 연구는 미흡한 실정이며, 급격히 증가하는 지능화된 악성코드를 실시간으로 분석하기 위해서는 악성코드와 C&C 서버 간 주고받는 통신 메시지 분석이 선행되어야 한다.

Table 1. Classification by Malware type

Type	Behavior	Network dependence
Virus	Spreading	Often
Worm	Spreading, Denying Service	Often
Ransomware	File encryption, Denying Service	Some times
Bot	C&C Command execution, Create vulnerability, Denying Service	Always
Remote Access Trojan	C&C Command execution, Deceiving the User, Stealing Information, Create vulnerability	Always
Scareware[5]	Deceiving the User	Sometimes
Spyware	Stealing Information	Often
Adware	Deceiving the User	Sometimes
Cryptominer	Stealing Resources, Deceiving the User	Often

하지만 악성코드와 통신하는 C&C 서버가 차단되거나 활동하지 않는 경우 원활한 통신이 이루어지지 않기 때문에 악의적인 행위를 수행하지 않는다. 또한, 악의적인 행위가 발생하지 않으면 효과적인 악성코드 분석이 제한되는 문제가 발생한다. 원활한 통신 메시지 분석을 위해서는 실제 악성코드가 통신할 수 있는 C&C 서버를 구축해 통신을 주고받도록 하여 메시지를 분석하는 것이 가장 바람직하다. 하지만 실제 통신을 담당하는 C&C 서버를 완벽히 재현하는 건 불가능에 가깝고, C&C 서버의 구조나 자체 프로토콜을 모르는 상태에서 통신을 자동화하는 기술은 존재하지 않는다.

따라서, 본 논문에서는 실제 C&C 서버가 활동하지 않아도 악성코드와 C&C 서버 간의 통신 메시지를 분석할 수 있는 최적의 환경을 실험을 통해 도출하는 것이 목적이다. 이를 위해 도출한 환경을 비교하기 위해서는 먼저 악성코드를 정상적으로 실행 후 발생한 통신 메시지를 수집하고 검증할 수 있는 최적의 분석환경이 필요하다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구와 배경 지식을 기술하고, 3장에서는 본 논문에서 최적화 분석환경에 대한 도출을 설명한다. 4장에서는 도출된 분석환경을 실험을 통해 비교하고 검증한다. 마지막으로 결론을 맺는다.

Ⅱ. 관련 연구 및 배경 지식

2.1 Background

2.1.1 Intel Pintool

Intel에서 개발한 “Pintool”은 DBI(Dynamic Binary Instrumentation)도구로 불리며, 동적으로 바이너리를 분석할 수 있다. “Pintool”은 사용자가 원하는 실행 코드를 직접 개발할 수 있는 오픈소스이다. 사용자가 개발한 “Pintool”은 실행 파일과 에뮬레이터를 통해 같은 주소 공간에서 실행되어 사용자가 만든 코드를 실행된 프로그램 내부에서 실행할 수 있다.

즉, 악성코드 실행 시 사용자가 작성한 실행 코드를 삽입하여 악성코드의 실행 흐름을 변경해 프로그램 실행 시 호출되는 API Call 및 사용 중인 메모리 주소

등을 수집할 수 있다. 대부분의 악성코드는 네트워크 패킷 분석 도구인 "WireShark"를 탐지하기 위해 NIC(Network Interface Controller)의 Promiscuous Mode를 확인해 분석을 회피하는 특징을 갖고 있다[13, 37, 38].

악성코드의 "WireShark" 탐지는 Pintool을 이용해 프로그램 실행 코드의 기본 블록(Basic Block) 단위를 기준으로 함수를 추적하기 때문에 NIC로부터 패킷 수집을 하는 "WireShark" 탐지 기술에 대응할 수 있다. 또한, "Pintool"은 에뮬레이터 기반으로 실행되기 때문에 악성코드의 Hooking 탐지도 되지 않아 Hooking 기반의 분석 도구에 대한 회피기술에 대응하여 통신 메시지를 수집하고 검증할 수 있는 도구로 활용할 수 있는 장점이 있다.

2.1.2 Fakenet-ng

Fakenet-ng는 악성코드가 정상적으로 인터넷에 연결되었다고 착각하게 만들어 실제 동작을 수행할 수 있도록 가상 환경을 구성해주는 프로그램이다. Faeknet-ng는 DNS, HTTP, HTTPS, SMTP, TCP, UDP 프로토콜을 가상 서비스를 통해 실제 서비스에 연결된 것처럼 응답을 생성한다. Fakenet-ng 실행 시 관리자 권한을 요구하며, 별도의 물리적인 서버가 없어도 운영체제 내부에서 가상서버처럼 동작한다. 가상의 서버는 C&C 서버 역할을 하고 악성코드가 인터넷이 연결되지 않거나 실제 C&C 서버가 없어도 악성코드가 C&C 서버로 연결된 상황을 구현할 수 있어 C&C 서버로 보내는 통신 메시지를 확인할 수 있다[40].

2.1.3 악성코드 분석 회피기술

〈Table 2〉는 악성코드 정적 및 동적 분석을 회피하기 위한 대표적인 회피기술이며 대표적인 정적 분석을 회피하는 기술로는 프로그램을 압축하는 패킹 기술이 있다[10, 20]. 동적 분석 회피기술은 특정 조건이 만족해야만 악성코드가 실행하도록 하여 분석 시간을 지연시키는 특징을 가지고 있다. 이번 단락에서는 본 논문에서 제안하는 최적의 분석환경을 구축하는데 필요한 동적 회피기술만을 설명한다[6, 14, 16, 21].

Table 2. Analysis avoidance technology

	Avoidance Technology
Static Analysis	Packing(Compress, Protect)
	Hide
	Code Encryption
Dynamic Analysis	Timer
	User behavior
	Operating system environment
	Hardware resource

2.1.3.1 Timer

Timer은 악성코드가 운영체제의 sleep() 함수를 이용해 실행 시간을 지연시키거나 동작 시간을 예측할 수 없도록 하는 방법이며 악성코드 분석을 회피하기 위해 자주 사용되는 기술이다. 동적 분석 시 악성코드가 정상적으로 실행되지 않아 악성 행위를 분석할 수 없는 한계가 있는데 이는 악성코드가 Timer을 통해 일정 시간 후에 악성 행위를 하기 때문이다. 또한, 특정 날짜나 정해진 시간에만 동작시켜 분석을 지연시키거나 동작이 안되는 악성코드로 인식하게 하여 분석을 회피하는 특징을 가지고 있다[42].

2.1.3.2 사용자 행위

최근 지능화된 악성코드는 사용자 행위 개입을 요구한다. 사용자의 개입이 발생할 때까지 악성코드의 실행을 대기시켜 자동화된 동적 분석을 방해하고 지연시킨다. 이러한 사용자의 행위를 통해 사용자의 입력 또는 가상 환경이 아닌지 확인하는 방법으로 아래와 같은 세 가지 방법이 있다[8, 16].

첫 번째는 사용자의 행동에 따라 악의적인 행위가 수행되는 방식이다. 예를 들어 메시지 박스 창이 실행되면 사용자가 해당 메시지 박스 창의 닫기 버튼이나 특정 영역을 클릭할 때까지 악성행위가 수행되지 않는다.

두 번째는 문서 파일과 같이 스크롤 개입이 필요한 경우, 스크롤 바가 일정 수준 이동되었을 때 악의적인 행위가 실행되는 방식이다.

세 번째는 악성코드가 사용자의 최근 파일 목록을 확인해 실제 사용 중인 PC인지 확인하거나 웹 브라우저 접속 기록을 확인하여 가상머신 여부를 확인한다.

2.1.3.3 운영 체제 환경 확인

가상머신은 가장 많이 사용되는 분석환경 중 하나이며, 악성코드 실행 시 운영체제가 감염되는 것을 쉽게 복구할 수 있어 리얼머신 대신하여 널리 사용한다. 하지만 가상머신은 구동에 필요한 파일을 별도로 사용하고 있어 악성코드가 관련 파일을 감지하여 가상머신 여부를 판단하기 때문에 쉽게 식별되는 문제점이 있다.

또한, 안전한 악성코드 분석을 위해 네트워크 환경을 단절하고 실행하는 경우가 많은데 이런 경우 네트워크 환경을 확인하는 악성코드는 실제 행위를 하지 않고 종료될 수가 있다. 악성코드가 네트워크 연결을 확인하는 대표적인 방법으로는 Google SMTP 서버에 Ping을 보내고 정상적으로 응답이 되는지 확인하는 것이다. 그 밖에도 프로세스 상태를 확인하거나, 실행되어있는 DLL(Dynamic-Link Library) 정보를 검사하여 가상머신 여부를 확인한다[7, 18].

2.1.3.4 하드웨어 자원 확인

가상머신은 실제 물리 NIC가 아닌 가상 네트워크에 연결되어 있다. 악성코드는 이를 인지하고 악성 행위를 실행하지 않거나 정상 행위를 수행하여 분석을 방해한다. 일반적인 가상머신은 CPU와 메모리가 Host PC보다 부족하게 설정되어있는 경우가 많다. 예를 들면 16GB 메모리를 가진 Host PC로부터 생성된 Windows 가상머신은 4GB 메모리 정도로 기본 설정이 되어있다. 악성코드는 이러한 하드웨어 자원을 확인하기 위해 악성코드는 메모리를 지속해서 할당해 메모리의 크기를 확인해 가상머신 여부를 확인할 수 있다[18, 21].

2.1.4 악성코드 분석 회피 대응 기법

악성코드가 사용하는 회피 기법들을 통해 가상머신을 우회하기 때문에 의도한 대로 프로그램이 실행되지 않아 분석의 한계가 발생한다. 다양한 회피 기법들에 대응하기 위해 연구되고 있으며, 대표적인 네 가지 방법은 다음과 같다[9, 12, 17, 18, 19].

- **VM 설정 변경** : 가상머신을 실행하는데 필요한 파일인 ".vmx" 설정 파일을 수정하여 가상 환경임을 숨길 수 있다.
- **레지스트리 변경** : 가상머신의 레지스트리 정보를 수정하여 레지스트리 정보를 수정하고, 가상머신 관련 프로세스가 실행되지 않도록 레지스트리 키를 변경하여 가상머신 회피에 대응할 수 있다.
- **사용자 행위 패턴 인식:** 사용자의 행동 및 최근 행위를 인식하여 가상머신 여부를 판단하는 경우 사용자가 직접 개입하는 것과 같이 마우스 이벤트 및 인터넷 접속 기록을 생성하여 실제 머신인 것처럼 하여 가상머신 회피에 대응할 수 있다.
- **디버거 함수 관련 무력화** : Ollydbg 프로그램을 이용하여 바이너리 실행 파일 내에 Process Environment Block(PEB) 구조체 멤버값을 수정하여 디버깅할 수 있도록 변경하거나, 디버그 관련 함수의 레지스터 값을 변경하여 디버거 관련 함수를 우회하여 대응할 수 있다[15].

2.2 관련 연구

2.2.1 네트워크 패킷 추론 분석 연구

네트워크 패킷 추론 분석은 실행 코드 없이 프로토콜 메시지의 네트워크 추적을 통해 메시지 형식을 분류하고 추론하는 연구이다[24]. 네트워크 패킷에서 프로토콜 키워드를 추출하고, 메시지 형식을 재구성하고, 프로토콜 상태를 추론하거나 네트워크 메시지 필드를 자동으로 추출하고 특성들을 활용하여 추론하였다[26]. 네트워크 패킷을 분석하는 이전 연구와 다르게 리버스 엔지니어링을 자동화하여 통신 메시지 타입 및 필드의 정보를 추출하고 분석하였다[27]. 추론 기반 연구는 대부분 악의적인 키워드와 같은 시그니처 정보를 바탕으로 추론하는 경우가 많았으며, 통계적 상관관계를 적용한다. 이러한 통계적 상관관계는 서로 다른 메시지 또는 메시지 길이, 클라이언트 또는 서버 IP와 같은 메타데이터의 관계를 중점적으로 연구하고 있다[30].

2.2.2 실행 기반 통신 메시지 연구

실행 기반 통신 메시지 연구는 악성코드를 직접 실행해 발생한 행위와 정보를 바탕으로 통신 메시지를 분석하는 연구이다. 악성코드를 반복적으로 직접 실행해 수집한 통신 메시지 데이터를 바탕으로 악성코드와 C&C 서버 간 주고받는 통신 메시지를 학습할 수 있

는 시스템을 구현하였다[25].

통신 메시지 추적 및 동적 바이너리 분석으로 통신 메시지를 수집하고 의미 있는 컨텍스트 정보를 분류해 악의적인 통신 메시지와 C&C 서버 정보를 분석할 수 있다[28, 29].

다른 관련 연구로는 악성코드를 직접 실행해 악성코드가 생성하는 통신 메시지 필드, 모든 메시지 바이트에 대한 실행 컨텍스트 정보를 수집하고 이를 클러스터링하여 프로토콜 형식을 추론할 수 있다[31]. 네트워크 추적을 기반의 역공학 프레임워크를 통해 정확하고 직관적인 통신 메시지를 수집할 수 있으며, HTTP 프로토콜의 메시지 필드 및 메시지 형식 흐름등을 메시지를 통해 분석할 수 있다[32].

2.2.3 연구 동향 및 한계점

통신 메시지의 추론 및 분석 연구는 실행 기반 통신 메시지 연구 네트워크 패킷 기반의 추론 연구와 바이너리 실행 기반의 역공학 연구로 나누어진다.

네트워크 패킷 기반 연구의 경우, 통신 메시지의 특정 키워드들을 기반으로 추론하기 때문에 적절한 키워드를 선정해야 하는 문제점이 있다. 악성코드와 C&C 서버 간 전체 통신 메시지 분석은 악의적인 행위를 파악하는데 중요한 데이터이기 때문이다.

하지만 오래된 악성코드의 경우 대부분 C&C 서버 간의 네트워크가 이미 단절되어있기 때문에, 악성코드의 통신 메시지 패킷을 수집하기에는 한계가 있다. 악성코드의 요청에 대해 응답할 C&C 서버가 없으므로 악성코드 통신 메시지 패킷의 양이 다소 부족할 수 있고, 네트워크가 단절된 경우 악성코드가 C&C 서버로 전송하는 초기 요청 메시지에는 단순 봇넷의 상태를 전송하는 경우가 많으므로 통신 메시지에 악의적인 명령이 포함되지 않을 수도 있다.

이러한 문제점으로 인해 악성코드를 효과적으로 분석할 수 없으므로 악의적인 명령을 포함하는 핵심 통신 메시지를 수집할 수 있도록 악성코드와 C&C 서버가 통신할 방안이 필요하다. 기존의 실행 기반 통신 메시지 분석 연구들은 특정 악성코드만 분석하고 연구했기 때문에 다양한 악성코드에 실시간으로 대응할 수 없고 분석 회피기술로 인해 악성코드 실행 시 원하는 흐름대로 추적하는 데에 한계가 있다.

급격히 증가하는 악성코드와 명령어를 주고받는 C&C 서버를 효과적으로 차단하고 대응하기 위해 다양한 악성코드의 통신 메시지를 실시간으로 수집할 방안이 필요하다. 따라서 본 논문에서는 네트워크 패킷 기반 연구의 통신 메시지 패킷 수집의 한계점을 극복하기 위해 Fakenet-ng을 사용하여 가상의 C&C 서버를 구축하고 통신 메시지를 증가시켜 악의적인 행위가 포함된 메시지를 수집한다. 또한, 악성코드가 의도한 악의적인 실행 흐름을 추적할 수 없었던 기존 실행 기반 연구의 한계점을 해결하기 위해 Pintool을 이용하여 악성코드의 실행 흐름을 추적하고 악성코드의 통신 메시지를 실시간으로 수집하고 분석할 수 있는 최적의 환경을 도출한다.

Ⅲ. 최적의 악성코드 분석환경 설계 및 구축

3.1 Pintool을 이용한 통신 메시지 분석 방법

본 논문에서는 통신 메시지를 분석하기 위해 Pintool을 사용하여 악성코드를 실행하고 통신 메시지를 실시간으로 수집한다. Pintool의 Trace 기능은 분기가 발생하지 않는 연속적인 실행 흐름을 의미하는 기본 블록 단위를 찾아 API의 흐름을 추적할 수 있는 기능이다. Trace 기능을 이용하여 API 정보뿐만 아니라 API 주소로부터 스택의 크기만큼 메모리에 접근하여 API의 Argument 정보를 가져올 수 있다.

예를 들어 〈Fig 1〉과 같이 "Function" API에 대한 메모리 주소에 접근해 함수의 Argument에 들어있는 메시지를 추출할 수 있다. 통신 메시지를 보내는 함수의 Argument를 추출하면 악성코드가 C&C 서버와 통신하는 메시지인 것을 알 수 있어 악성코드의 악의적인 행위를 파악할 수 있다.

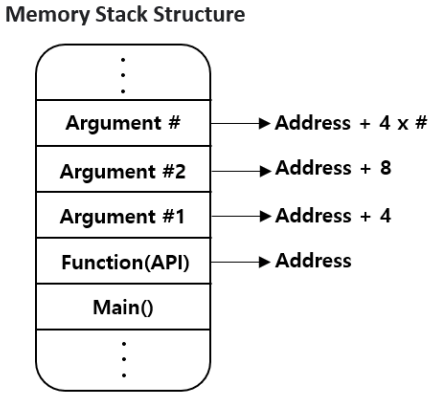


Fig. 1. Argument access concept in function

3.2 분석환경 구성

본 논문에서는 기존의 정적 분석 및 동적 분석환경에 추가적인 분석 기능을 더 하여 분석환경을 3단계로 구성하였다. 〈Fig 2〉와 같이 첫 번째 단계는 기초 정적 분석이며, 두 번째 단계는 동적 행위 분석이다. 마지막 단계는 가상 C&C 서버 역할을 하는 Fakenet-ng을 활용하고 Pintool을 이용하여 통신 메시지를 수집하고 분석한다.

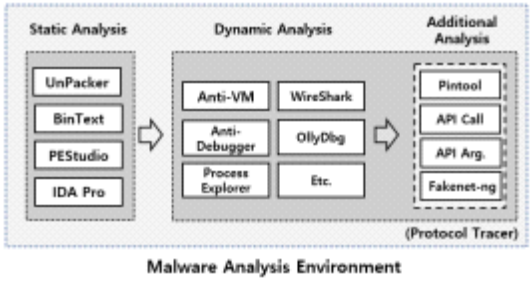


Fig. 2. Proposed optimal analysis environment

3.2.1 기초 정적 분석 단계

기초 정적 분석은 프로그램을 실행하지 않고 바이너리 파일의 구조를 분석하는 것을 의미하며, 정적 분석을 위해 "unpacker" 도구[4]로 패킹(Packing) 여부 확인 및 패킹을 해제하였다. 또한, "BinText" 도구[39]를 활용하여 악성코드가 만들어진 소스 코드로부터 사용된 문자열들을 검색하고, 도메인 주소 또는 파일을 지정하기 위한 문자열을 확인하였다. 더불어 IAT(Import Address Table) 정보 수집을 통해 악성코드가 어떤 함수를 호출하여 사용하는지 알 수 있어 어떠한 악의적인 행위를 하는지 수집하고 분석할 수 있었다. 이러한 정적 분석 결과는 4장 실험 결과로부터 도메인 주소와 통신 메시지 등을 통해 정상 동작을 검증하는 데이터로 활용했다.

3.2.2 동적 행위 분석 단계

동적 행위 분석 단계는 악성코드를 직접 실행하여 악성코드의 행위를 분석하는 과정이다. 악성코드를 실행할 때에는 기존 PC의 안전을 위해 가상머신에서 실행하는 경우가 많다. 하지만 이런 경우 악성코드가 가상머신을 인식하는 "Anti-VM" 탐지기술로 인해 악성코드가 정상적으로 실행이 되지 않는다. 악성코드의 "Anti-VM" 탐지에 대응하기 위해 VM 관련 파일 및 레지스트리를 수정해 탐지 기술을 회피하였다. 동적 분석 도구인 "WireShark", "Ollydbg", "Process Explorer"를 통해 악성코드 실행 시 나타나는 시스템의 변화를 수집하고 분석하였다[11, 23]. 동적 분석 결과는 정적 분석 결과와 마찬가지로 4장 실험에 관한 결과가 정상적인 통신 메시지를 주고받았는지 검증하기 위한 결과로 사용된다.

3.2.3 가상 C&C 환경을 이용한 통신 메시지 분석

마지막 단계에서는 가상 C&C 환경을 구성해 악성코드의 통신 메시지를 수집할 수 있는 환경을 구성하였다. 가상의 C&C는 Fakenet-ng를 이용해 구성하고 악성코드가 C&C 및 인터넷 연결 상태가 정상적이라고 인식하게 하였다. Fakenet-ng은 악성코드가 보낸 Request 통신 메시지에 정상 Response로 응답을 주어, 악성코드가 추가 및 의도한 악의적인 행위를 할 수 있도록 도와주는 역할을 한다. 악성코드의 통신 메시지를 수집할 수 있도록 Pintool을 직접 수정하여 Protocol Tracer을 개발하였다. Pintool은 "Anti-Debugger"의 탐지에 대응할 수 있어 디버깅 탐지 및 Hooking 탐지를 우회해 악성코드가 정상적인 흐름을 수집할 수 있다.

Pintool의 Trace 기능을 직접 수정해 네트워크 통신 관련 함수와 메모리 주소로부터 함수 Argument 가지고 올 수 있도록 개발하였다. Protocol Tracer을 통해 악성코드 실행 시 호출되는 API의 Argument 값으로부터 악성코드가 통신하는 악의적인 행위 데이터 수집할 수 있으며 수집데이터를 바탕으로 통신 메시지를 분석할 수 있다. 통신 메시지의 분석 정보는 통신 메시지를 보내기 위해 사용한 API, 통신 메시지의 길이, 데이터 타입, 변수 유무 및 통신 메시지의 송신 횟수 등을 결과로 확인할 수 있다.

Ⅳ. 실험 및 결과

4.1 실험 목적 및 방법

본 논문에서는 제시한 분석환경에서 악성코드를 실행하고 Protocol Tracer을 이용해 통신 메시지를 추출하였다. 또한, 악성코드의 정적 및 동적 분석 결과와 Protocol Tracer의 수집 결과를 비교해 악의적인 행위가 수행되었는지 검증하였다. 이를 위한 실험 방법은 〈Fig 3〉과 같다.

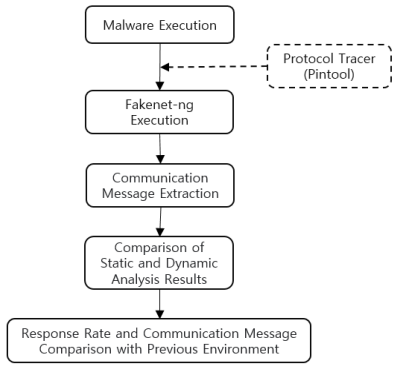


Fig. 3. Experiment method and steps

- 정적/동적 분석기반 악성 통신 행위 확인
- 악성코드 분석 회피 대응 실행 환경 구성
- Fakenet-ng 실행 후 Protocol Tracer을 통한 통신 메시지 수집
- 수집된 통신 메시지와 정적 및 동적 분석 결과 비교
- 이전 환경과 반응률 및 통신 메시지 비교

4.2 실험 환경

4.2.1 악성코드 수집 및 분석환경

〈Table 3〉은 악성코드 분석을 위한 실험 환경이다. 실험을 위해 자체 악성코드 수집 시스템인 SIMON(Suspicious Information Monitoring system in website)을 사용했다[41]. SIMON을 이용하여 약 42만 개의 악의적인 웹사이트 대한 실시간 모니터링을 수행하였고, 악성코드 파일 약 3,000여 개를 수집하였다. 또한 "VirusTotal"로부터 2017년부터 2020년 동안 제공받은 파일 307,552개 샘플 데이터로부터 실험에 사용할 파일을 선정하였다.

Table 3. Experiment environment

	Version & Detail
CPU	3.5GHz Intel Core i3
RAM	2GB
HDD	40GB
OS	Windows 7 32bit
Python	3.7.9
Visual Studio	2015 v140
Pintool	v3.13

4.2.2 악성코드 분류 및 선정

원활한 실험을 위해 Windows에서 동작하고 외부 응용프로그램이 필요하지 않은 악성코드를 대상으로 실험하고자 "SIMON"과 "VirusTotal"로부터 수집한 악성코드 중 "exe", "dll" 실행 파일 유형으로 10,203개의 악성코드를 1차 분류하였다. 2차로 "ssdeep"을 이용해 무작위로 선택한 악성코드와 유사도를 비교하여 유사도 값이 유일한 악성코드 1,000개를 분류했고, 이 중에서 네트워크 행위를 하지 않는 악성코드와 랜섬웨어 악성코드는 실험 데이터 수집 과정에서 데이터가 암호화되거나 암호화폐를 요구하는 화면이 생성돼 실험대상에서 제외하였다. 실험 및 검증은 정상 동작하고 실제 통신 행위를 하는 313개 악성코드를 대상으로 진행하였다.

4.3 실험 결과 및 분석

〈Table 4〉는 환경 구성 시 사용한 Fakenet-ng 도구를 이용한 악성코드의 통신 패킷을 "WireShark"로 수집한 결과이다[22]. Fakenet-ng 사용 전에는 "k1.rar" 하나의 GET 요청 메시지만 보내다가 연결되지 않자 TCP 연결이 종료된 후에도 반복적으로 통신 연결을 요청하는 것을 확인할 수 있었다. 하지만 Fakenet-ng 사용 후에는 하나의 GET 요청 메시지가 발생한 이전과 달리 "k1.rar" ~ "k5.rar"의 5개의 통신 메시지가 추가되는 것을 확인할 수 있었다.

수집한 악성코드 중 대표적인 4개를 비교한 결과 〈Table 4〉와 같이 Fakenet-ng 사용 전에는 0~1번의 유일한 메시지의 통신만 확인되었지만, 사용 후에는 통신이 2~7번까지 증가하는 것을 확인할 수 있었다.

Table 4. Fakenet-ng Before/After Compare

Before	After
GET /cj//k1.rar HTTP/1.1 799 → 49218 [ACK] Seq=1 Ack 799 → 49218 [RST, ACK] Seq= 49219 → 799 [SYN] Seq=0 Win [TCP Retransmission] 49219 799 → 49219 [SYN, ACK] Seq= 49219 → 799 [ACK] Seq=1 Ack GET /cj//k1.rar HTTP/1.1 799 → 49219 [ACK] Seq=1 Ack 799 → 49219 [RST, ACK] Seq= 49220 → 799 [SYN] Seq=0 Win 799 → 49220 [SYN, ACK] Seq= 49220 → 799 [ACK] Seq=1 Ack GET /cj//k1.rar HTTP/1.1 799 → 49220 [ACK] Seq=1 Ack 799 → 49220 [RST, ACK] Seq= 49221 → 799 [SYN] Seq=0 Win 799 → 49221 [SYN, ACK] Seq=	GET /cj//k1.rar HTTP/1.1 GET /cj//k1.rar HTTP/1.1 GET /cj//k1.rar HTTP/1.1 HTTP/1.0 200 OK (text/html) HTTP/1.0 200 OK (text/html) HTTP/1.0 200 OK (text/html) GET /cj//k2.rar HTTP/1.1 GET /cj//k2.rar HTTP/1.1 GET /cj//k2.rar HTTP/1.1 HTTP/1.0 200 OK (text/html) HTTP/1.0 200 OK (text/html) HTTP/1.0 200 OK (text/html) GET /cj//k3.rar HTTP/1.1 GET /cj//k3.rar HTTP/1.1 GET /cj//k3.rar HTTP/1.1 HTTP/1.0 200 OK (text/html) HTTP/1.0 200 OK (text/html) HTTP/1.0 200 OK (text/html) GET /cj//k4.rar HTTP/1.1 GET /cj//k4.rar HTTP/1.1 GET /cj//k4.rar HTTP/1.1 HTTP/1.0 200 OK (text/html) Continuation Continuation GET /cj//k5.rar HTTP/1.1 GET /cj//k5.rar HTTP/1.1

추가로 "Pintool" 도구를 통해 〈Fig 4〉와 같이 악성코드와 C&C 간의 통신 메시지와 사용한 API와 횟수까지 확인할 수 있었다.

악성코드와 C&C 서버와의 주고받는 메시지 중 "Send" 관련 API를 기준으로 분석하였다. 〈Table 5〉는 분류한 313개의 악성코드를 실험한 결과이다. 〈Table 5〉의 반응률은 "Send" 관련 API가 사용되어 유일한 통신 메시지가 추가로 생성돼 Fakenet-ng에 반응한 악성코드의 비율을 말한다.

〈Table 5〉에서 "①"은 가상머신만 구축한 기본 환경이며, "②"는 "①" 환경에서 Anti-VM 회피 대응 및 Anti-Debugging 회피 대응을 구축한 환경을 말한다. "③"은 "①"과 "②" 대응 환경에 더해 Fakenet-ng을 사용한 환경이다. "③" 분석환경은 Fakenet-ng을 기반으로 가상의 C&C 서버를 구성해 악성코드를 실행하고 기존 환경과 다르게 "WireShark"가 아닌 개발한 "Pintool"을 이용해 통신 메시지를 수집했다.

〈Table 6〉는 313개의 악성코드 중 대표적인 4개의 악성코드를 분석한 결과이다. 메시지 개수는 중복이 없는 유일한 "Send" 관련 API 통신 메시지를 의미한다. 변수는 중복이 제거된 유일한 메시지로부터 단순

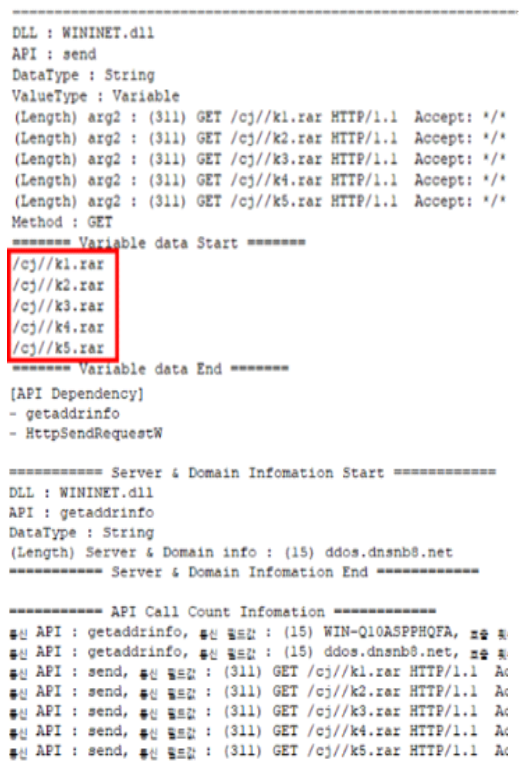


Fig. 4. Malware D's communication message result

Table 5. Malware reaction result according to environment

Environment Configuration	Response rate
①. Default Environment	12%
②. ①+Anti-VM, Anti-debugging	17%
③. ①+②+Fakenet-ng+Pintool	24%

Table 6. Comparison of the derived optimal analysis environment

	② Analysis Environment		③ Analysis Environment	
	Number of messages	Variable	Number of messages	Variable
Malware'A'[33]	1	X	2	O
Malware'B'[34]	1	X	2	O
Malware'C'[35]	1	X	7	O
Malware'D'[36]	0	X	5	O

히 양적으로 생성된 메시지가 아닌 통신 메시지에 파라미터 정보가 변경되어 악성코드와 C&C 간 통신 메시지가 생성된 것을 의미한다. 〈Fig 4〉에 표시된 구역처럼 "Send" 메시지로부터 변하는 데이터가 있는지에 따라 "O"와 "X"로 비교하였다.

정적 및 동적 분석 결과 대부분의 악성코드는 네트워크 환경에 연결 여부를 확인하기 위해 구글 SMTP 서버에 Ping을 보내 확인하고 C&C 서버와 통신한다. 실험 결과 "①" "Default 환경"과 "②" "Anti-VM 대응 환경"보다 "③" 환경일 때 악성코드가 "Send" 메시지 더 많이 호출해 C&C 서버와 통신 메시지를 주고받는 것을 〈Table 6〉 실험 결과로 검증할 수 있었다.

위의 실험은 단순히 악성코드의 통신 양이 증가한 것이 아니라, 특정 통신 메시지가 다수 발생하여 각각 다른 통신 메시지가 C&C 서버로 전송되는 것 확인하였다. 기존환경 대비 통신 메시지에서 유일한 "Send"가 추가로 생성되어 반응률이 높아졌다.

이러한 통신 변화는 악성코드가 네트워크 환경 및 C&C 서버와의 통신 가능 여부를 확인하고 실행되기 때문으로 분석된다. 그 결과 기본 환경보다 통신 메시지 수집할 수 있는 최적의 환경의 구성을 통해 정확한 통신

메시지 정보를 수집하고 분석할 수 있음을 검증하였다.

Ⅴ. 결론 및 향후 연구

악성코드의 통신 및 내부 동작에 대한 원리를 분석하기 위해서는 악성코드를 만든 해커의 의도를 파악하고 실행의 목적 및 조건을 파악하는 것이 무엇보다 중요하다. 그러나 악성코드마다 실행 조건 알아내는 것은 한계가 있는데, 이는 실행 또는 분석환경에 따라 악성코드의 동작이 다르기 때문이다. 따라서 본 논문에서는 Pintool 기반의 Protocol Tracer와 Fakenet-ng을 활용해 악성코드의 통신 메시지를 분석하기 위한 최적의 환경을 도출하였다.

본 연구 결과는 향후 프로토콜 역공학 및 실시간 C&C 서버 차단을 위한 통신 메시지 분석환경과 가상화 C&C 서버 자동 구성연구로 활용할 수 있을 것으로 판단하며, 향후 본 논문에서 제안한 분석환경을 바탕으로 악성코드 자동실행 환경 및 통신 메시지 데이터를 활용해 기계학습 기반의 C&C 가상화 서버 분석 시스템을 개발하는 연구를 진행할 예정이다.

References

[1] CheckPoint, "Cyber Security Report 2020", CheckPoint, 5 Ha'Solelim Street, Tel Aviv 67897, Israel, 80, 2020.

[2] Or-Meir, Ori, et al. "Dynamic malware analysis in the modern era—A state of the art survey," ACM Computing Surveys (CSUR) 52.5, vol.52, no.5, pp.1-48, Sep. 2019.

[3] securelist, "DDoS attacks" https://securelist.com/ddos-report-in-q1-2018/85373/, May. 2021

[4] github, "unipacker" https://github.com/unipacker/unipacker, May. 2021.

[5] Shahzad, Raja Khurram, and Niklas Lavesson. "Detecting scareware by mining variable length instruction sequences," IEEE, pp.1-8, Aug. 2011.

[6] A. Moser, C. Krügel, and E. Kirda, "Exploring multiple execution paths for malware analysis," IEEE Security and Privacy, pp.231-245, May. 2007.

[7] CHEN, Xu, et al., "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), IEEE, pp.177-186, Jun, 2008.

[8] Moser, Andreas, Christopher Kruegel, and Engin Kirda. "Exploring multiple execution paths for malware analysis," 2007 IEEE Symposium on Security and Privacy (SP'07). IEEE, pp.231-245, May, 2007.

[9] Jiang, Xuxian, Xinyuan Wang, and Dongyan Xu. "Stealthy malware detection and monitoring through VMM-based "out-of-the-box" semantic view reconstruction," ACM Transactions on Information and System Security (TISSEC), vol.13, no.2, pp1-28, Mar, 2010.

[10] Sun-Kyun Kim, Hajin Kim And Mi-Jung Choi. "Design and Implementation of Malware Automatic Unpacking System in Anti-VM/Debugging Environment," The Journal of Korean Institute of Communications and Information Sciences 43(11), pp.1929-1940, Nov, 2018.

[11] Talukder, Sajedul, and Zahidur Talukder. "A survey on malware detection and analysis tools," International Journal of Network Security & Its Applications vol. 12, no.2, pp.37-57, Mar, 2020.

[12] Choi, Suk-June, Deuk-Hun Kim, and Jin Kwak. "A study on the Prevention of Malware Anti-VM Technique," Proceedings of the Korea Information Processing Society Conference. Korea Information Processing Society, pp.246-249, Apr. 2017.

[13] Park, Juhyun, et al. "Automatic Detection and Bypassing of Anti-Debugging Techniques for Microsoft Windows Environments," Advances in Electrica

l and Computer Engineering, 19(2), pp23-29, May, 2019.
[14] Chakkaravarthy, S. Sibi, D. Sangeetha, and V. Vaidehi. "A survey on malware analysis and mitigation techniques," Computer Science Review, vol.32, pp.1-23, May, 2019.
[15] Raffetseder, Thomas, Christopher Kruegel, and Engin Kirda. "Detecting system emulators," International Conference on Information Security. Springer, Berlin, Heidelberg, pp.1-18, Oct, 2007.
[16] Graziano, Mariano, et al., "Needles in a haystack: mining information from public dynamic analysis sandboxes for malware intelligence," Proc of the 24th USENIX Conference on Security Symposium, USENIX Association, pp.1057-1072, Aug, 2015.
[17] Lin, Jie, et al. "VMRe: A Reverse Framework of Virtual Machine Protection Packed Binaries," 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC). IEEE, pp.528-535, Jun, 2019.
[18] Sun, Li, Tim Ebringer, and Serdar Boztas. "An automatic anti-anti-VMware technique applicable for multi-stage packed malware," 2008 3rd International Conference on Malicious and Unwanted Software (MALWARE). IEEE, pp17-23, Oct, 2008.
[19] J. W. Kim, J. W. Bang and M. J. Choi. "Anti-Anti-Debugging Study to Understand and Disable Anti-Debugging for Malware Analysis," The Journal of Korean Institute of Communications and Information Sciences 45(1), pp.105-116, Jan. 2020.
[20] J. W. Kim, J. W. Bang, and M. J. Choi, "A study on automatic disabling of anti-debugging in manual unpacking," KNOM 2019 Conference, pp. 58-61, May. 2019.
[21] Homook Cho, et al. "Automatic Binary Execution Environment based on Real-machines for Intelligent Malware Analysis," KIISE Transactions on Computing Practices, 22(3), pp. 139-144, Mar. 2016.
[22] ASLAN, Ömer; SAMET, Refik. "Investigation of possibilities to detect malware using existing tools," IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA). IEEE, pp. 1277-1284, Oct. 2017.
[23] Afianian, Amir, et al. "Malware dynamic analysis evasion techniques: A survey," ACM Computing Surveys (CSUR), Vol. 52, No. 6, Nov. 2019.
[24] Wang, Yipeng, et al. "A semantics aware approach to automated reverse engineering unknown protocols," 20th IEEE International Conference on Network Protocols (ICNP), pp. 1-10, Oct. 2012.
[25] Graziano, Mariano, Corrado Leita, and Davide Balzarotti. "Towards network containment in malware analysis systems," Proceedings of the 28th Annual Computer Security Applications Conference, pp. 339-348, Dec. 2012.
[26] Luo, Jian-Zhen, and Shun-Zheng Yu. "Position-based automatic reverse engineering of network protocols," Journal of Network and Computer Applications, Vol. 36, No. 3, pp. 1070-1077, May. 2013.
[27] Caballero, Juan, and Dawn Song. "Automatic protocol reverse-engineering: Message format extraction and field semantics inference," Computer Networks, Vol. 57, No. 2, pp. 451-474, Feb. 2013.
[28] Bossert, Georges, Frédéric Guihéry, and Guillaume Hiet. "Towards automated protocol reverse engineering using semantic information," Proceedings of the 9th ACM symposium on Information, computer and communications security, pp. 51-62, Jun. 2014.
[29] Xu, Zhaoyan, et al. "Autoprobe: Toward

s automatic active malicious server probing using dynamic binary analysis," Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 179-190, Nov. 2014.

[30] Bermudez, Ignacio, et al. "Automatic protocol field inference for deeper protocol understanding," 2015 IFIP Networking Conference (IFIP Networking), pp. 1-9, May. 2015.

[31] Lin, Zhiqiang, et al. "Automatic Protocol Format Reverse Engineering through Context-Aware Monitored Execution," NDSS, Vol. 8, Feb. 2008.

[32] Goo, Young-Hoon, et al. "Framework for precise protocol reverse engineering based on network traces," NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1-4, Apr. 2018.

[33] virustotal, "90b309d0616391af7732ef3eb70ad4a39c61dd9163774a17f7df69094e95745e" https://www.virustotal.com/gui/file/, May. 2021.

[34] virustotal, "8924332e99cdc1cea5fb5a1a61c1633dc4fa7d40765072f2177ee8235093b8ef" https://www.virustotal.com/gui/file/, May. 2021.

[35] virustotal, "9002a6aa9685d0d41a98142c4d0699a6d6df827553bf750e73ae5875e8bc88b4" https://www.virustotal.com/gui/file/, May. 2021.

[36] virustotal, "3a8735434cfa5b86bde96f88d7594976e8d5cef4e553c282079a5cbc54831029" https://www.virustotal.com/gui/file/, May. 2021.

[37] Qadeer, Mohammed Abdul, et al. "Network traffic analysis and intrusion detection using packet sniffer," 2010 Second International Conference on Communication Software and Networks. IEEE, pp. 313-317, Feb. 2010.

[38] Wikipedia, "Promiscuous mode" https://en.wikipedia.org/wiki/Promiscuous_mode, May. 2021.

[39] Google, "Bintext" https://www.aldeid.com/wiki/BinText, May. 2021.

[40] Gitgub, "Fakenet" https://github.com/fireeye/flare-fakenet-ng/releases, May. 2021.

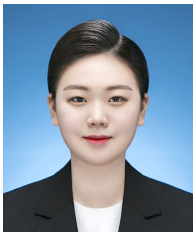
[41] Ho-Mook Cho, Kyeong-Seok Lee, Yong-Min Kim, "Intelligent Malware Distributing Web Page Detection based on Real Web Browser," The Korean Institute of Information Scientists and Engineers, pp. 1075-1077, Jun. 2017.

[42] Crandall, Jedidiah R., et al. "Temporal search: Detecting hidden malware timebombs with virtual machines," ACM SIGOPS Operating Systems Review, Vol. 40, No. 5, pp. 25-36, Oct. 2006.

〈저자소개〉

신 강 식 (Kangsik Shin) 정회원
2016년 2월: 충남대학교 컴퓨터공학과 학사
2018년 2월: 충남대학교 컴퓨터공학과 석사
2020년~현재: 한국과학기술원 사이버보안연구센터 연구원
〈관심분야〉 악성코드 분석, 사이버보안, 딥러닝 보안

정 동 재 (Dong-Jae Jung) 정회원
2011년: 아주대학교 정보 및 컴퓨터공학부
2013년: 한국과학기술원 정보보호대학원 석사
2020년: 한국과학기술원 정보보호대학원 박사
2020년~현재: 한국과학기술원 사이버보안연구센터 선임연구원
〈관심분야〉 악성코드 분석, 시스템보안, 흐름 분석



최 민 지 (Min-Ji Choe) 정회원
2020년: 순천향대학교 정보보호학과 학사
2020년~현재: 한국과학기술원 사이버보안연구센터 연구원
〈관심분야〉 악성코드 분석, 시스템/사이버 보안

조 호 묵 (Ho-Mook Cho) 정회원
2006년: 아주대학교 정보통신공학과 정보보호학(공학석사)
2018년: 전남대학교 정보보안협동과정(이학박사)
2014년~현재 한국과학기술원 사이버보안연구센터 책임연구원
〈관심분야〉 악성코드 분석, XAI