한국항행학회 논문지 제16권 제1호 2012년 2월

# Rustock B형과 C형의 감염절차 분석 및 은닉파일 추출

# Analysis on the Infection Process and Abstract of the Hidden Files of Rustock B and C

이경률*, 임강빈*

Kyung-Roul Lee*, Kang-Bin Yim*

## 요 약

최근 악성코드에 의한 피해가 개인이나 기업으로부터 기관이나 국가 차원으로 진화하고 있고 악성코드가 사용하는 기술 역시 점점 고도화되고 다양한 기법들을 흡수하여 매우 지능적으로 발전하고 있다. 한편, 보안전문가들은 시그니쳐 탐색 등의 정적 분석과 역공학 등의 동적 분석으로 이에 대응하고 있지만 이는 새로이 출현하는 지능적인 악성코드에 긴급히 대처하기에는 부족함이 있다. 따라서, 악성코드들의 행위 분석에 앞서 대개의 악성코드들이 가지는 감염절차 및 파일 은닉기법에 대한 분석을 우선적으로 수행하고 이를 토대로 재빠른 초동분석이 이루어진 후 그 무력화 방법을 포함한 제반 상세분석이 이루어질 것이 요구된다. 따라서 본 논문에서는 악성코드의 초기 진압을 위하여 요구되는 감염 절차의 분석과 파일 은닉기법의 분석 방안을 연구하였으며 그 과정에서 스팸메일을 발송하는 것으로 가장 널리 알려진 Rustock을 대상으로 실험하였다. 실험 결과를 통하여, 향후 새로이 출현하는 악성코드에 대한 재빠른 대처가 가능할 것으로 판단된다.

## Abstract

The technologies used by the malicious codes have been being advanced and complicated through a merge of the existing techniques, while the damages by the malicious codes are moving from individuals and industries to organizations and countries. In this situation, the security experts are corresponding with the static analysis and the dynamic analysis such as signature searching and reverse engineering, respectively. However, they have had a hard time to respond against the obfuscated intelligent new zero day malicious codes. Therefore, it is required to prepare a process for a preliminary investigation and consequent detailed investigation on the infection sequence and the hiding mechanism to neutralize the malicious code. In this paper, we studied the formalization of the process against the infection sequence and the file hiding techniques with an empirical application to the Rustock malicious code that is most notorious as a spammer. Using the result, it is expected to promptly respond to newly released malicious codes.

Key words : Reverse engineering, Rustock, Rootkit, Malicious code analysis, Infection

## I. 서 론

최근 악성코드에 의한 DoS, DDoS, 스팸메일 등의 공격이 증가하고 있으며[1][2][3], 특히 농협 전산망

* 순천향대학교(Soonchunhyang University)
· 제1저자 (First Author) : 이경률
· 교신저자 (Corresponding Author) : 임강빈
· 투고일자 : 2011년 11월 27일
· 심사(수정)일자 : 2011년 11월 14일 (수정일자 : 2012년 2월 22일)
· 게재일자 : 2012년 2월 28일

마비 사건, 소니 플레이스테이션 해킹 사건, 7.7 DDoS 공격[4], 3.4 DDoS 공격[5][6]들의 피해규모와 그 심각성을 볼 때 악성코드에 대한 조기 대응방안이 시급한 실정이다. 이에 많은 보안전문가들이 악성코드를 탐지하기 위한 방안뿐만 아니라 감염 후의 활동에 대응하기 위한 방안들을 연구 중에 있으나 아직까지 완벽한 해결을 하기에는 현실적으로 어려움이 있다. 현재까지 가장 위협적이면서도 널리 알려진 악성코드로는 Stuxnet, Rustock 등이 있으며[7][8], 최초 발견되기까지 오랜 기간 동안 은닉되어 분포된 것으로 알려져 악성코드 탐지의 중요성에 대한 비중이 높아지고 있다. 따라서 이미 알려진 악성코드들의 전파경로에 대한 분석과 전파 후 피해자 컴퓨터 내에서 감염되는 방법에 대한 분석이 필요하며, 이를 기반으로 앞으로 발생할 악성코드로부터의 피해를 최소화하기 위한 방안을 모색해야 한다. 또한 대부분의 악성코드는 자신의 실행파일이나 디바이스 드라이버, DLL 등 실행에 필요한 파일들을 준비하고 자기방어기법이나 루트킷 기법 등을 활용하여 은닉하거나 탐지를 방해하기 때문에 그에 대한 힌트나 무력화하기 위한 방안도 강구되어야 한다[9][10].

본 논문은 Rustock B형과 C형의 루트킷 기법에 대해 분석하고 분석된 결과를 토대로 은닉된 악성코드의 추출방안에 대해 고찰하고자 한다. 추출된 악성코드는 전파 시 빠르게 대응할 수 있는 방안을 마련하기 위해 반드시 필요하므로 본 논문의 분석 과정 및 결과는 앞으로 새로이 출현할 악성코드의 초동조사에 효과적으로 활용될 수 있을 것으로 판단된다.

## Ⅱ. 관련연구

Win32/Rustock은 백도어 트로이목마의 일종으로 스팸메일을 발송하기 위한 목적으로 제작된 악성코드이며 2006년 초에 처음으로 발견되었다. 이후 2008년부터 중요하게 나타나기 시작하여 2010년에 가장 일반적이고 널리 퍼진 악성코드 중의 하나가 되었으며 현재까지 A형, B형, C형이 발견되었다. 그림 1은 지난 4년간 Rustock의 탐지결과에 대한 변화를 보이고 있다.

### 2-1 Rustock의 구성요소와 감염과정

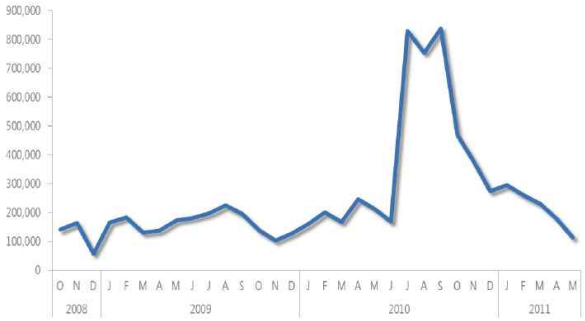



그림 1. Microsoft 안티소프트웨어에 의해 발견된 Rustock 탐지결과[8]
Fig. 1. Detection result of the Rustock by the Microsoft's anti-software[8]

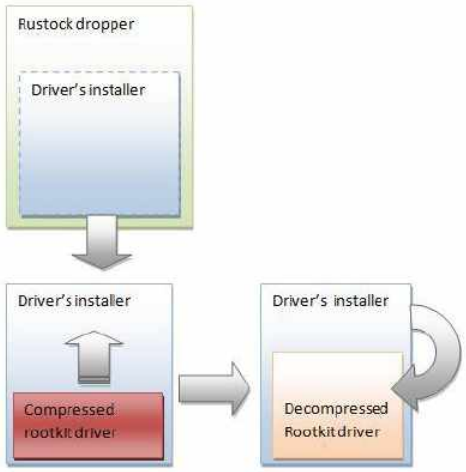



그림 2. Rustock 감염과정[8]
Fig. 2. Infection process of the Rustock[8]

Rustock은 정적분석에서 자신이 발견되는 것을 최소화하기 위하여 코드 압축, aPLib와 UPX 같은 난독화, RC4 암호 알고리즘 등을 활용하였으며, 그 감염과정을 살펴보면 그림 2 및 표 1과 같다.

Rustock은 dropper를 이용하여 다른 컴퓨터에 감염시키며, 감염 후 악의적인 행위를 위한 커널 모드 드라이버를 설치한 후 자기 자신의 은닉을 위한 후킹 및 기타의 기능들을 수행한다. 마이크로소프트의 DCU(Digital Crimes Unit)에서는 Rustock의 행위와 관련하여 2011년 1월 22일부터 2011년 2월 4일까지 1,300,000 개의 고유의 IP 주소들이 C&C 서버에 접속

을 시도하는 것을 탐지하였으며, 감염 후 활동에 대해 분석한 결과 인터넷의 다양한 DNS A 호스트들을 위해 1,406개의 고유의 색인을 하였고, 메일 서버를 위한 DNS MX 기록을 위해 2,238 개의 고유의 색인을 하였다. 또한 1,376 개의 이메일 서버에 스팸 메일 발송을 시도하였으며, C&C 서버나 다른 서버에 22개의 접속을 시도하였다. 이에 대한 활동을 그림 3에 나타내었으며, 이는 24분 동안 Rustock에 감염된 단일 컴퓨터에서 시도한 결과이다[8].

표 1. Rustock 감염과정의 단계별 설명[8]
Table 1. Step-by-step description on the Rustock infecion process[8]

| 단계 | 설명 |
|---|---|
| Step 1 (Dropper) | · 유저모드에서 실행<br>· 복호화 기능<br>· 루트킷 드라이버 설치<br>· 이전에 감염되었는지 확인하기 위해 레지스트리를 점검<br>· RC4 알고리즘으로 코드 암호화<br>· aPLib로 패킹<br>· 갱신 가능한 요소를 점검하기 위해 C&C(Command and Control) 서버에 접속 |
| Step 2 (Driver) | · 커널 모드에서 실행<br>· 윈도우 시스템 드라이버로 위장<br>· beep.sys나 null.sys와 같은 드라이버로 대체해 은닉 |
| Step 3 (Concealment) | · INT 2Eh 인터럽트를 이용하여 유저모드 봇 클라이언트와 통신<br>· 자가복호기능을 통해 스스로 암호화된 코드를 복호화<br>· 존재를 숨기기 위해 SSDT를 후킹<br>· 코드 분석을 방지하기 위해 디버거의 존재를 점검 |

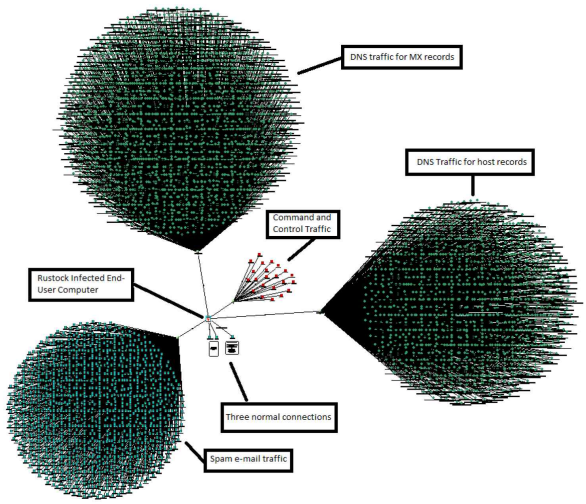



그림 3. Rustock에 감염된 단일 컴퓨터에서 24분 동안 발생한 움직임[8]
Fig. 3. Visual representation of the activity of a single Rustock-infected computer within a 24-minute timeframe[8]

## 2-2 스팸

Rustock이 사용자 컴퓨터에 감염되면 C&C 서버에 접속하여 통신을 위한 준비를 하고 감염된 컴퓨터의 정보를 서버에 보내며 사용자와의 어떠한 관계도 가지지 않은 채 스팸 메시지를 발송한다. 초기 버전의 Rustock은 "botdll.dll"이라 불리는 주문 제작한 SMTP 클라이언트 엔진이었으며 2008년에 C&C 서버에 의해 Windows Live Hotmail을 이용하도록 수정되었다. 또한 전통적인 이메일 메시지가 평문으로 전송되기 때문에 Rustock은 DHTTPS를 이용하여 메시지를 암호화하여 발송하고 Win32/Lethic과는 다르게 한 번에 한 사람에게 메일을 전송한다. 그림 4는 Rustock과 Win32/Lethic에서의 메일 전송 방법을 비교하고 있다.

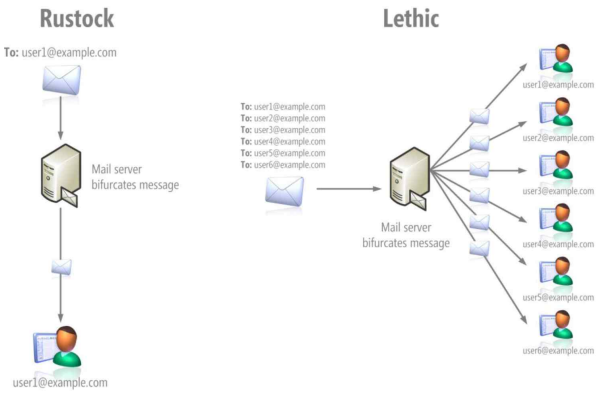



그림 4. Rustock과 Lethic의 전파 모델 비교[8]
Fig. 4. Comparison of the Rustock and Lethic spam distribution models[8]

Rustock은 세계에서 가장 큰 스팸봇 중 하나로 발표되었으며 하루에 300억 개의 스팸 메일을 발송한다. 마이크로소프트의 DCU가 관찰한 결과 Rustock에 감염된 단일 컴퓨터가 45분 만에 7500개의 스팸 메일을 발송하며, 하루에는 24만개의 스팸 메일을 발송하는 것으로 밝혀졌다. Rustock에 의해 보내진 스팸 메시지를 그림 5에 나타내었다.

관련연구에서 확인할 수 있듯이 이러한 악성행위를 수행하는 악성코드를 초기에 탐지하여 분석하지 못하면 매우 심각한 피해를 초래할 수 있기 때문에 감염 절차 및 은닉된 악성코드를 추출하기 위한 방안이 제시되어야 한다.


그림 5. Rustock에 의해 발송된 스팸 메일의 예[8]
Fig. 5. An example of spam message sent through the Rustock botnet[8]

## Ⅲ. Rustock 감염 절차 및 파일 은닉 방법의 분석

본 논문에서는 감염 절차 및 파일 은닉 방법을 분석함에 있어서 이미 널리 알려진 도구들을 활용하였으며, 파일 탐지, 레지스트리 탐지, 후킹 탐지 등의 결과를 토대로 은닉된 파일을 판단하고, 감염 후 발생한 행위에 대하여 분석하였다[7][10].

### 3-1 Rustock B형 분석

#### 3-1-1 초동 분석

초동 분석은 파일 탐지, 레지스트리 탐지, 후킹 탐지로 이루어지며, 이러한 세 단계를 통하여 의심스러운 행위를 하는 파일 및 프로세스를 탐지하고 이를 추출하여 세부적인 코드 분석 단계에 활용한다.

· 파일 탐지

Rustock B형에 감염된 컴퓨터에서 숨겨진 파일을 탐지한 결과, 커널 모드 디바이스 드라이버 파일을 은닉하고 있었으며 그 결과를 그림 6에 나타내었다.

| Suspect File | Status |
|---|---|
| C:₩WINDOWS₩system32:lzx32.sys:$DATA | Hidden |

그림 6. Rustock B형 은닉 파일 탐지 결과
Fig. 6. Detection result of hidden file of the Rustock type B

악성코드가 응용 프로그램 형태로 배포되는 상황에서 은닉된 드라이버 파일이 존재하는 것으로 볼 때, 일차적으로 응용 프로그램이 드라이버 파일을 생성하고 이를 은닉하는 것이라 판단된다.

· 레지스트리 탐지

탐지한 드라이버 이름으로 레지스트리를 확인한 결과 서비스를 시작하기 위하여 드라이버가 레지스트리에 등록된 것을 확인할 수 있으며, 이를 그림 7에 나타내었다.

· 후킹 탐지

후킹 탐지를 위해 Rootkit Unhooker를 이용하였으며 그에 대한 결과를 그림 8에 나타내었다.

후킹 탐지 결과를 살펴보면 IAT 후킹, 인라인 후킹, 시스템 콜 후킹을 이용하였으며 네트워크 드라이버인 tcpip.sys, ndis.sys, wanarp.sys 내의 특정 함수들을 후킹하였다. 상기 세 단계의 결과로 볼 때 일반적으로 정상적인 프로그램이 가지는 행위가 아닌 악성코드로 의심할 만한 후킹 행위들이 존재하므로 은닉된 파일을 삭제하는 것만으로 초기 대처가 가능하다.

#### 3-1-2 상세 분석

초동 분석이 제대로 이루어졌는지 확인하기 위해 실제 Rustock B형을 실행하여 감염 절차 및 파일 은닉 등 악의적인 행위에 대해 상세히 분석하였다.

· IAT 함수

Rustock B형이 임포트하여 사용하는 라이브러리 함수는 표 2와 같으며, IDA 디스어셈블러를 이용하여 결과를 도출하였다.

응용 프로그램으로 배포됨을 유추하여 볼 때 CreateFileW를 통해 드라이버 파일을 생성하고, CopyFileA를 이용하여 내부 코드를 복사하는 것으로 판단되며, 사용자 컴퓨터의 프로세스에 접근하기 위해 OpenProcess를 활용할 것으로 판단된다.

· 파일 은닉 분석

CereateFileW는 파일을 생성하는 함수로서 내부에 중지점(Breakpoint)를 설정하여 살펴보면 "C:\Windows\system32:lzx32.sys"가 인자로 전달되는 것을 확인할 수 있으며, 이를 그림 9에 나타내었다.

표 2. Rustock B형의 IAT 함수 목록
Table 2. IAT function list of the Rustock type B

| 라이브러리명 | 함수명 |
|---|---|
| kernel32.dll | HeapReAlloc |
| | DeviceIoControl |
| | FindClose |
| | GetDriveTypeA |
| | FindResourceA |
| | CreateFileW |
| | GetMailslotInfo |
| | FindFirstFileA |
| | HeapAlloc |
| | OpenProcess |
| | SetSystemTime |
| | Sleep |
| | CreateMailslotA |
| | SetFilePointer |
| | DeleteAtom |
| | CopyFileA |
| | AddAtomA |

이러한 결과는 초동분석에서 확인한 것과 같은 결과를 가지는 것을 확인할 수 있으며 NTFS ADS(Alternate Data Streams)에 파일을 생성하는 것 또한 확인할 수 있다.

· DLL 인젝션

또한, Rustock B형은 OpenProcess 함수를 이용하여 explorer.exe에 인젝션을 시도한다. explorer.exe의 핸들을


그림 9. Rustock B형의 CreateFileW 함수 호출 시의 인자
Fig. 9. Parameters to CreateFileW function in the Rustock type B in a function call


그림 7. Rustock B형 레지스트리 검색 결과
Fig. 7. Registry search result for the Rustock type B

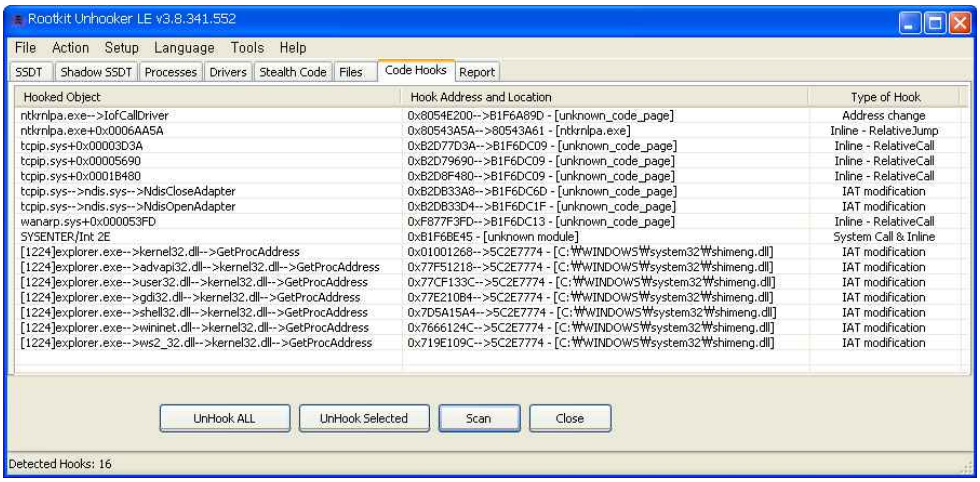



그림 8. Rustock B형 후킹 탐지 결과
Fig. 8. Hooking detection result of the Rustock type B

구한 뒤 프로세스 내에 메모리를 할당하며 할당된 메모리에 WriteProcessMemory로 악의적인 코드를 삽입한 후, 삽입한 코드를 실행하기 위해 CreateRemoteThread를 호출한다. 인젝션에서 제일 처음 이루어지는 과정인 OpenProcess를 호출할 때 중지점을 설정한 결과를 그림 10에 나타내었다.


그림 10. Rustock B형의 OpenProcess 호출 시의 중지점 설정 결과

Fig. 10. Breakpoint set result of OpenProcess function in the Rustock type B when function calls

OpenProcess 함수의 인자 중 ProcessID에 해당하는 값이 0x4C8(1224)이며, 작업 관리자를 통해 explorer.exe 프로세스의 PID가 1224임을 확인하였다. 이 결과의 의미는 explorer.exe 프로세스를 통해 악의적인 작업을 시도한다는 것으로서, 실제 explorer.exe에 새로이 할당된 메모리가 존재함을 확인할 수 있었으며 이를 그림 11에 나타내었다.

OpenProcess를 이용하여 구한 핸들을 활용하여 explorer.exe 내의 새로 할당한 메모리에 특정 코드를 두 번 삽입하는 것을 확인하였으며 이를 그림 12에 나타내었다.

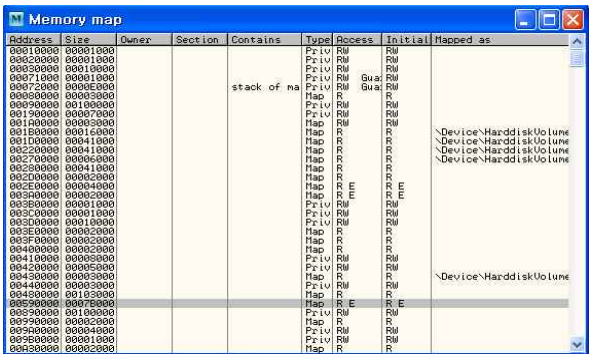



그림 11. Rustock B형의 explorer.exe 프로세스에 할당한 메모리 영역

Fig. 11. Allocated memory area of Rustock type B on the explorer.exe process

처음 호출할 때의 파라미터는 EBX에 저장되어 있고, 두 번째 호출할 때의 파라미터가 EBP에 저장되어 있음을 토대로 살펴볼 때 처음 복사하는 코드는 Rustock B형 자체의 응용 프로그램이며, 두 번째 복사하는 코드는 Rustock 내의 0x401B82번지임을 확인할 수 있다. 이후, 삽입한 코드를 실행하기 위해 CreateRemoteThread 함수를 호출하며 이를 그림 13에 나타내었다.

· 기타 행위

Rustock B형은 시스템 서비스 함수를 제어하기 위한 목적으로 INT 2E를 후킹하였으며 도구를 이용한 탐지 결과를 그림 14에 나타내었다.

그리고 네트워크 통신을 은닉하기 위해 윈도우즈 네트워크 드라이버(wanarp.sys, tcpip.sys)의 내부 함수를 후킹하였으며 도구를 이용한 탐지 결과를 그림 15에 나타내었고, 드라이버 은닉 결과를 그림 16에 나타내었다.

마지막으로 Rustock B형은 IofCallDriver를 후킹하고 있음을 확인하였으며, 이를 그림 17에 나타내었다. 이와 같은 후킹의 경우 IRP 요청을 필터링하기 위한 용도로 활용될 수 있기 때문에 악의적인 행위로 판단이 가능하다.

### 3-1-3 은닉 파일 추출

Rustock B형의 은닉된 드라이버를 추출하기 위한 방안은 감염되지 않은 컴퓨터와 감염된 경우로 나뉜다.

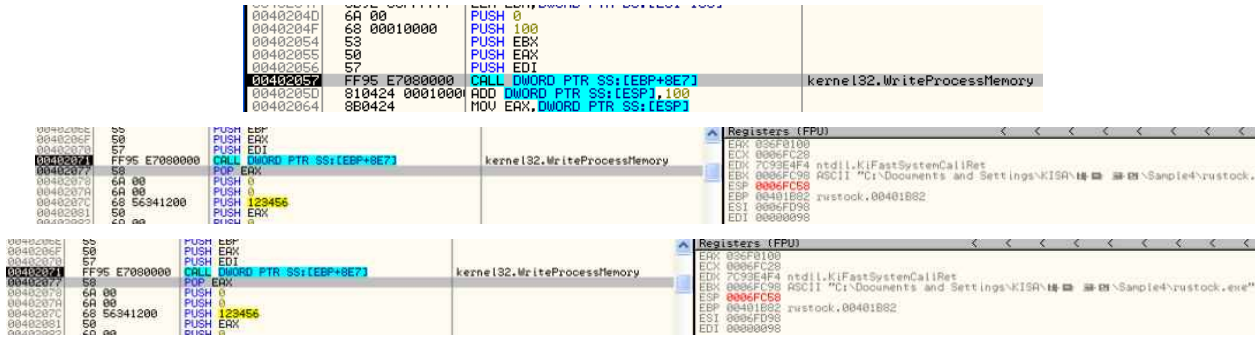



그림 12. Rustock B형의 WriteProcessMemory 파라미터
Fig. 12. Parameters of WriteProcessMemory function of Rustock type B

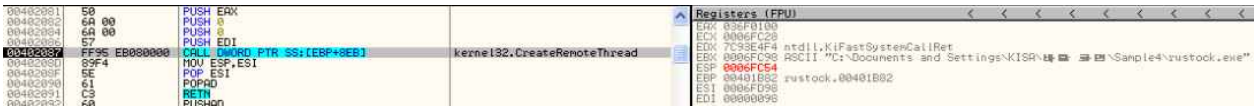



그림 13. Rustock B형의 CreateRemoteThread 호출 시 중지점 설정 결과
Fig. 13. Breakpoint set result of CreateRemoteThread function in the Rustock type B when function calls

| | | |
|---|---|---|
| SYSENTER/Int 2E | 0xB1F6BE45 - [unknown module] | System Call & Inline |

그림 14. Rustock B형 시스템 서비스 콜 후킹 탐지 결과
Fig. 14. Hooking detection result of system calls of the Rustock type B

| | | |
|---|---|---|
| tcpip.sys+0x00003D3A | 0xB2D77D3A-->B1F6DC09 - [unknown_code_page] | Inline - RelativeCall |
| tcpip.sys+0x00005690 | 0xB2D79690-->B1F6DC09 - [unknown_code_page] | Inline - RelativeCall |
| tcpip.sys+0x0001B480 | 0xB2D8F480-->B1F6DC09 - [unknown_code_page] | Inline - RelativeCall |
| tcpip.sys-->ndis.sys-->NdisCloseAdapter | 0xB2DB33A8-->B1F6DC6D - [unknown_code_page] | IAT modification |
| tcpip.sys-->ndis.sys-->NdisOpenAdapter | 0xB2DB33D4-->B1F6DC1F - [unknown_code_page] | IAT modification |

그림 15. Rustock B형 네트워크 드라이버 후킹 탐지 결과
Fig. 15. Hooking detection result of network-base driver of the Rustock type B

| Driver Name | Loaded from (could be only name) | Address | Size | Hidden | References |
|---|---|---|---|---|---|
| ?_empty_? | ?_empty_? | 0xB1F6BE45 | 256 | Yes | |

그림 16. Rustock B형 드라이버 은닉 탐지 결과
Fig. 16. Detection result of hidden driver of the Rustock type B

| | | |
|---|---|---|
| ntkrnlpa.exe-->IofCallDriver | 0x8054E200-->B1F6A89D - [unknown_code_page] | Address change |

그림 17. Rustock B형의 기타 후킹 결과
Fig. 17. Another hooking detection result of the Rustock type B

감염되지 않은 컴퓨터의 경우 Rustock B형을 실행함으로써 실행과정 및 동작을 분석하는데 활용될 수 있으며 CreateFileA 함수 호출 시 간단한 리버싱을 통하여 경로를 변경함으로써 드라이버 파일을 추출할 수 있다. 그림 18은 CreateFileA 함수 호출 시 생성할 파일 경로 및 파일명을 변경한 결과이며, 그림 19는 변경한 파일명으로 추출한 드라이버에 대하여 IDA를 이용하여 그 임포트 함수 리스트를 출력한 결과이다.

감염된 컴퓨터의 경우에는 은닉된 파일을 추출하는 도구를 이용하여 복구가 가능하며, 실험을 위해 Rootkit Unhooker를 이용한 방법과 그 결과를 그림 20, 21에 나타내었다.

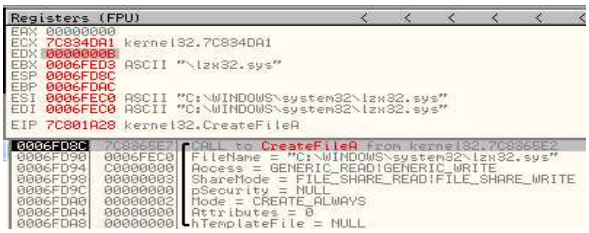



그림 18. Rustock B형 드라이버 파일 추출을 위한 경로 변경

Fig. 18. Modified path to abstract hidden driver file of the Rustock type B

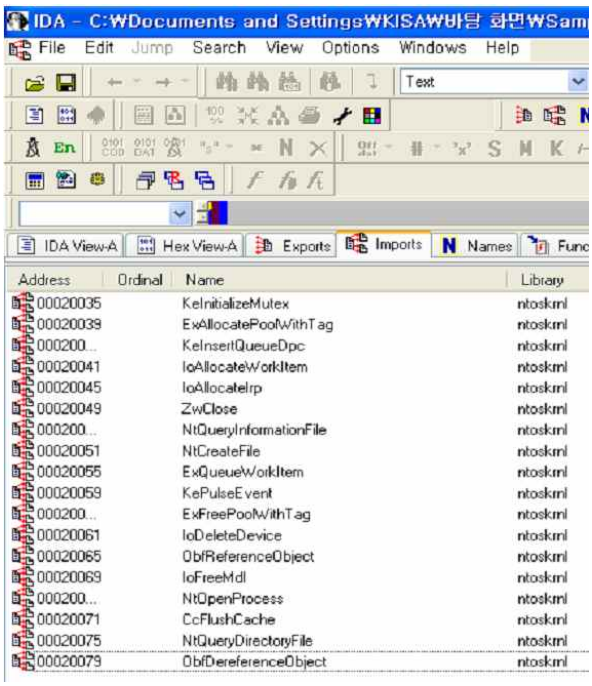



그림 19. 추출한 Rustock B형의 임포트 함수 리스트

Fig. 19. Imported functions list of abstracted hidden driver of the Rustock type B

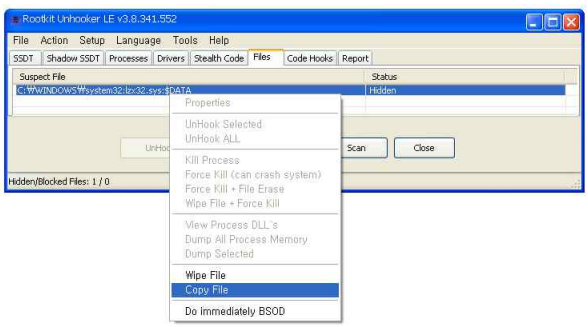



그림 20. 은닉 파일 복구 도구를 이용한 Rustock B형 드라이버 추출 방법

Fig. 20. Abstraction of the hidden driver of the Rustock type B by hidden file recovery tool


그림 21. 은닉 파일 복구 도구를 이용한 Rustock B형 드라이버 추출 결과

Fig. 21. Abstracted result of hidden driver of the Rustock type B by hidden file recovery tool

마지막으로 감염되지 않은 컴퓨터와 감염된 컴퓨터 모두 추출할 수 있는 방안은 콘솔모드로 부팅하여 강제적으로 파일을 복사하여 추출이 가능하며, 이를 삭제함으로써 치료가 가능하다.

### 3-2 Rustock C형 분석

#### 3-2-1 초동 분석

Rustock C형은 레지스트리 탐지, 후킹 탐지로 초동 분석이 구성된다.

· 레지스트리 탐지

새로이 생성되거나 접근이 불가한 레지스트리를 살펴보면 graide32에서 키를 열 수 없다는 오류를 출력하며, 이를 그림 22에 나타내었다.

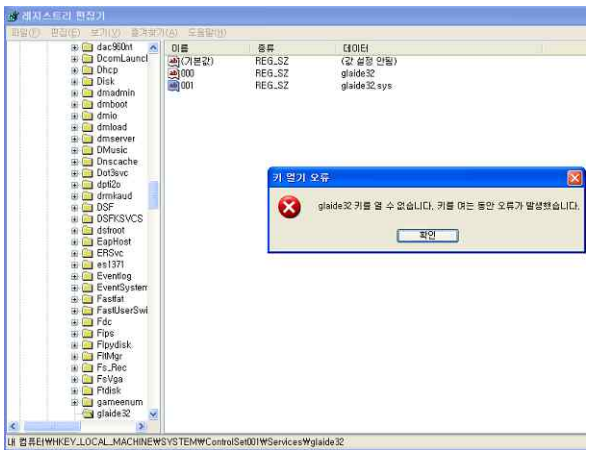



그림 22. Rustock C형 레지스트리 오픈 오류

Fig. 22. Registry open fault of the Rustock type C

출력 결과에서 확인할 수 있듯이 의심스러운 행위를 하는 서비스와 드라이버가 존재하고 있으며, 접근 통제를 하고 있다. 이를 확인하기 위해서 콘솔모드로 부팅하여 이름을 수정하거나 경로를 변경하였으며, 악성코드로 의심되는 glaide32 드라이버가 로드 되지 않은 상태에서 레지스트리를 확인한 결과는 그림 23과 같다.


그림 23. Rustock C형 레지스트리 정보
Fig. 23. Registry information of the Rustock type C

레지스트리 정보를 살펴보면 windows\system32\drivers 디렉토리 내 glaide32.sys 커널 모드 드라이버 파일을 시스템이 시작할 때 로드 하는 것을 의미하며 로드 후 glaide32 드라이버는 악의적인 기능을 수행한다.

· 후킹 탐지

후킹 탐지를 위한 도구 Rootkit Unhooker를 이용하여 확인한 결과 IAT 후킹, 커널 오브젝트 변경을 탐지하였으며, 기존에 존재하는 정상적인 윈도우즈 드라이버인 beep.sys의 후킹도 확인되었다. 이를 그림 24, 25에 나타내었다.

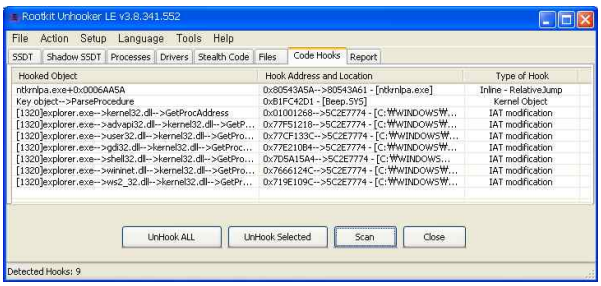



그림 24. Rustock C형 후킹 탐지 결과 1
Fig. 24. Hooking detection result 1 of the Rustock type C

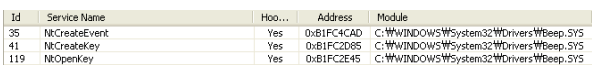



그림 25. Rustock C형 후킹 탐지 결과 2
Fig. 25. Hooking detection result 2 of the Rustock type C

3-2-2 상세분석

초동 분석을 토대로 더욱 상세한 분석을 위하여 Rustock C형을 직접 실행함으로써 감염 절차 및 파일 은닉 등 악성행위에 대해 분석하였다.

· IAT 함수

Rustock C형이 임포트하여 사용하는 라이브러리 함수를 IDA 디버거를 이용하여 조사하였으며 그 결과를 표 3에 나타내었다.

표 3. Rustock C형의 IAT 함수 목록
Table 3. IAT function list of the Rustock type C

| 라이브러리명 | 함수명 |
|---|---|
| kernel32.dll | GetModuleHandleA |
| | GetCommandLineA |
| | GetVersionExA |
| | ExitProcess |
| | GetProcAddress |
| | WriteFile |
| | GetStdHandle |
| | GetModuleFileNameA |
| | UnhandledExceptionFilter |
| | FreeEnvironmentStringsA |
| | GetEnvironmentStrings |
| | FreeEnvironmentStringsW |
| | WideCharToMultiByte |
| | GetLastError |
| | GetEnvironmentStringsW |
| | SetHandleCount |
| | GetFileType |
| | GetStartupInfoA |
| | HeapDestroy |
| | HeapCreate |
| | VirtualFree |
| | LoadLibraryA |
| | HeapAlloc |
| | GetACP |
| | GetOEMCP |
| | GetCPInfo |
| | VirtualAlloc |
| | HeapReAlloc |
| | LCMapStringA |
| | MultibyteToWideChar |
| | LCMapStringW |
| | GetStringTypeA |
| | GetStringTypeW |
| | RtlUnwind |
| | GetLocaleInfoA |
| | VirualProtect |
| | GetSystemInfo |
| | VirtualQuery |

· 파일 은닉 분석

간단한 리버싱을 시도한 결과를 보면, Rustock C형은 beep.sys와 null.sys를 후킹하고, 새로운 드라이버

glaide32.sys를 생성한 후 실행했던 파일을 삭제한다. beep.sys 파일을 복사하기 위해 새로운 파일 1.tmp를 생성하고 beep.sys의 코드를 복사한 후 beep 서비스를 시작한다. 이에 대한 과정을 그림 26, 27, 28에 나타내었다.

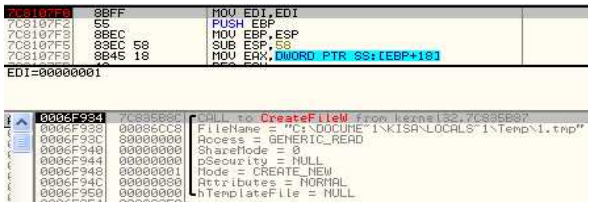


그림 26. Rustock C형의 새로운 파일 1.tmp 생성 과정
Fig. 26. Creation of new file(1.tmp) of the Rustock type C

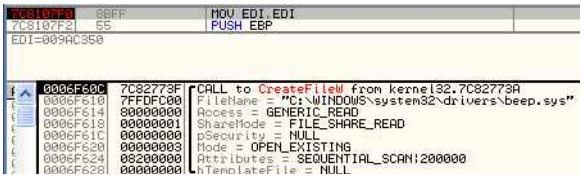


그림 27. Rustock C형의 beep.sys 파일 오픈 과정
Fig. 27. File open process of beep.sys of the Rustock type C

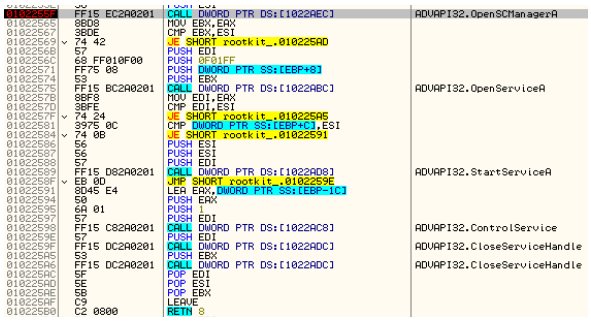


그림 28. Rustock C형의 beep 서비스 시작 과정
Fig. 28. Beginning process of beep service of the Rustock type C

beep.sys 파일 복사 및 서비스를 시작하고 나면 null.sys 파일 복사를 위해 2.tmp 파일을 생성하고 null.sys 코드를 생성한 파일에 복사하며 null 서비스를 시작한다. 이를 그림 29, 30, 31에 나타내었다.

그림 29. Rustock C형의 새로운 파일 2.tmp 생성 과정
Fig. 29. Creation of new file(2.tmp) of the Rustock type C

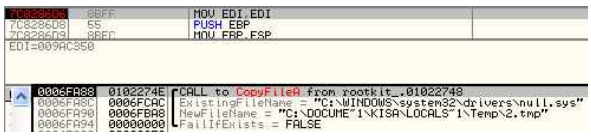


그림 30. Rustock C형의 null.sys 파일 오픈 과정
Fig. 30. File open process of null.sys of the Rustock type C

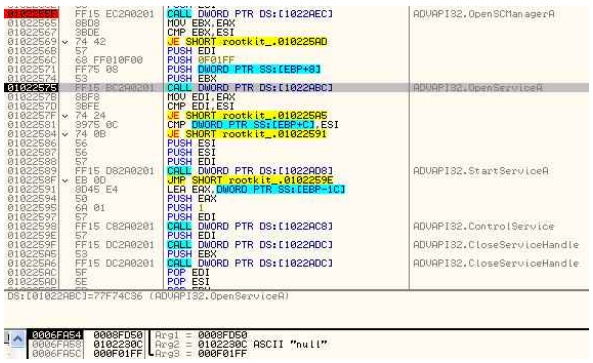


그림 31. Rustock C형의 null 서비스 시작 과정
Fig. 31. Beginning process of null service of the Rustock type C

마지막으로 드라이버 파일 glaide32.sys을 생성한 후 코드를 복사한다. 이를 그림 32, 33에 나타내었다.

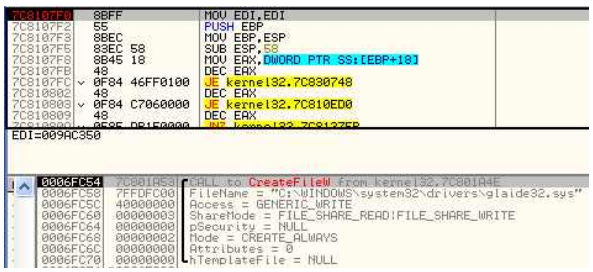


그림 32. Rustock C형의 glaide32.sys 파일 생성 과정
Fig. 32. Creation of driver file(glaide32.sys) of the Rustock type C

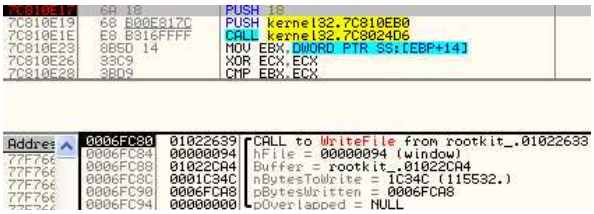


그림 33. Rustock C형의 glaide32.sys 파일 복사 과정
Fig. 33. Copy of driver file(glaide32.sys) of the Rustock type C

복사된 파일을 살펴보면 첫 두 바이트가 0x4D5A로 이루어져 있으므로 PE 파일 포맷임을 확인할 수 있다.

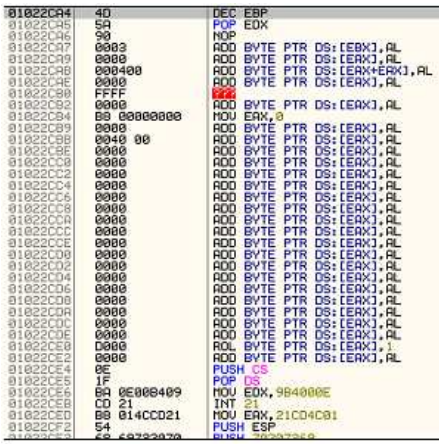


그림 34. Rustock C형의 복사된 glaide32.sys 파일 내부 정보
Fig. 34. File information of copied driver file(glaide32.sys) of the Rustock type C

파일이 정상적으로 복사되었다면 glaide32 서비스를 시작하고, 최초 실행하였던 응용 프로그램을 삭제한다.

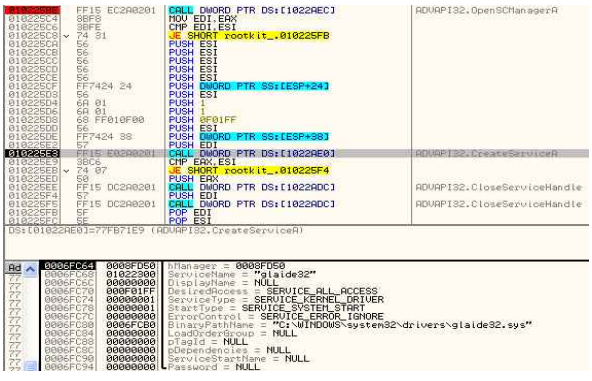
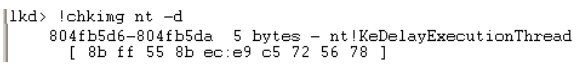


그림 35. Rustock C형의 glaide32 서비스 시작 과정
Fig. 35. Beginning process of galide32 driver

그림 36. Rustock C형의 최초 실행 파일 삭제 과정
Fig. 36. File delete process of infecting file of the Rustock type C

· 기타 행위

WinDbg를 이용하여 함수의 무결성을 점검한 결과 초동분석에서 발견되지 않았던 추가적인 후킹이 발견되었으며 이를 그림 37, 표 4에 나타내었다.



```
lkd> !chkimg nt -d
    804fb5d6-804fb5da  5 bytes - nt!KeDelayExecutionThread
      [ 8b ff 55 8b ec:e9 c5 72 56 78 ]
```

그림 37. WinDbg를 이용한 Rustock C형에 감염된 컴퓨터의 함수 무결성 검사 결과
Fig. 37. Integrity check result of infected computer by the Rustock type C

표 4. WinDbg를 이용한 Rustock C형에 감염된 컴퓨터의 함수 무결성 검사 리스트
Table 4. Integrity check list of infected computer by the Rustock type C

| 모듈명 | 함수명 |
|---|---|
| ntoskrnl.exe | KeDelayExecutionThread |
| | ExAllocatePool |
| | ExAllocatePoolWithTag |
| | NtDuplicateObject |
| | NtOpenProcess |
| | NtOpenThread |

결과를 살펴보면 nt!KeDelayExecutionThread 함수 0x804FB5D6번지의 5바이트는 0x8B 0xFF 0x55 0x8B 0xEC이어야 하지만 분기 명령인 0xE9 0xC5 0x72 0x56 0x78로 변경되어 있으며 이는 인라인 후킹임을 의미한다.

### 3-2-3 은닉 파일 추출

Rustock C형은 파일은 은닉하지 않고 있으므로 별도의 추출과정에 대해서는 생략하였다.

## Ⅳ. 결 론

본 논문은 악성코드에 대한 재빠른 대응을 위한 감염 절차 분석 및 은닉된 파일을 추출하는 방안에 대해 연구하였으며 이를 위하여 최근 스팸메일을 보내는 가장 악명 높은 악성코드인 Rustock의 B형과 C형에 대해 실험하였다. 분석은 초동분석과 상세분석으로 이루어지는데 1차의 초동분석을 통하여 추출한 결과와 2차의 상세분석으로부터 얻은 결과를 비교함으로써 악성코드의 존재 및 감염 절차를 분석할 수 있었으며, 은닉된 파일을 추출하는 일련의 과정 및 방법에 대하여 확인하였다. <그림 38>에서 확인할 수 있듯이 마이크로소프트에서 2011년 1분기 Rustock 치료 결과를 살펴보면 거의 0에 가까운 메시지를 발송하는 것을 확인할 수 있다. 이는 본 논문의 분석과 같이 악성코드의 초동조사 후 빠르게 대처할 수 있다면 악성코드에 의한 피해를 많이 감소시킬 수 있기 때문에 매우 의미 있는 접근방법이라 할 수 있다.

향후, 분석된 결과를 토대로 악성코드를 분석하기 위한 일련의 프로세스를 정의하고, 그 절차에 따라 초동조사를 시행하는 것만으로 악성코드의 초기 진압을 수행하며 이후의 상세분석을 통하여 악성코드의 구조와 동작을 파악할 수 있도록 하기 위한 프레임워크가 설계되어야 하겠다.

## 참 고 문 헌


[1] 김혁준, 이상진, "네트워크 포렌식을 통한 분산서비스거부공격 비교분석," *한국정보보호학회 학회지*, 21(4), pp. 69-74, 2011년 8월

[2] Suresh, M., Anitha, R., "Evaluating Machine Learning Algorithms for Detecting DDoS Attacks," *Proceedings of the Fourth International Conference on Advances in Network Security and Applications(CNSA 2011)*, India, pp. 441-452, Jul. 15-17, 2011

[3] 김태희, 강문설, "수집과 빈도 분석을 이용한 인터넷 게시판의 스팸 메시지 차단 방법," *한국정보처리학회 논문지*, 18C(2), pp. 61-70, 2011년 4월

[4] 인터넷침해사고대응지원센터(KISC), "국내 주요 사이트 대상 분산서비스거부공격 분석보고서," *한국정보보호진흥원*, 2009년 7월

[5] *안철수연구소*, "3.4 DDoS 분석보고서," 2011년 3월

[6] *금융보안연구원*, "3.4 DDoS 공격 분석결과," pp. 2-6, 2011년 3월

[7] 이경률, 임강빈, "Stuxnet의 파일 은닉 기법 분석 및 무력화 방법 연구," *한국항행학회 논문지*, 14(6), pp. 838-837, 2010년 12월

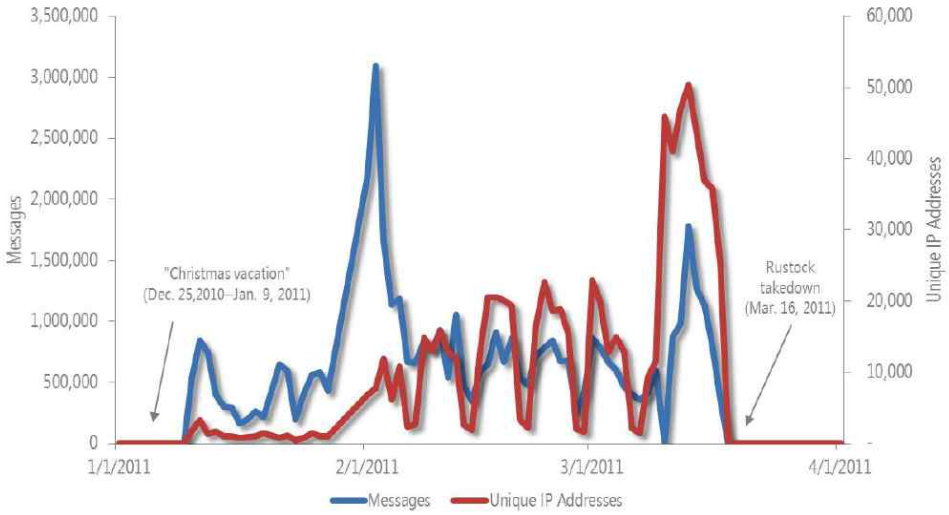




그림 38. 2011년 1분기 Rustock 활동 탐지 결과[8]
Fig. 38. Rustock activity detected in 1Q11[8]

[7] 이경률, 임강빈, "Stuxnet의 파일 은닉 기법 분석 및 무력화 방법 연구," *한국항행학회 논문지,* 14(6), pp. 838-837, 2010년 12월

[8] Microsoft, "Battling the Rustock Threat," *Security Intelligence Report Special Edition*, Jan. 2010 through May 2011

[9] 임을규 외, "악성코드 유형에 따른 자동화 분석 방법론 연구," *한국인터넷진흥원,* 2009. 6

[10] 임강빈, 이경률 외, "분석기법 우회 악성코드 분석 방법 연구," *한국인터넷진흥원,* 2010. 11

## 이 경 률 (李庚栗)

2008년 8월: 순천향대학교 정보보호학과 (공학사)
2010년 8월: 순천향대학교 정보보호학과 (공학석사)
2010년 9월~현재: 순천향대학교 정보보호학과 박사과정
2011년 5월~12월: (미)퍼듀대학교 정보보호교육연구센터 연구원
관심분야 : vulnerability analysis, virtualized obfuscation, system security, insider threats

## 임 강 빈 (任綱彬)

1992년 2월: 아주대학교 전자공학과 (공학사)
1994년 2월: 아주대학교 전자공학과 (공학석사)
2001년 2월: 아주대학교 전자공학과 (공학박사)
1999년 3월~2000년 2월: (미)아리조나 주립대학교 연구원
2010년 12월~2012년 2월 : (미)퍼듀대학교 정보보호교육 연구센터 객원교수
2003년 3월~현재: 순천향대학교 정보보호학과 교수
2005년 3월~현재: 한국정보보호학회 이사
2009년 3월~현재: 한국인터넷정보학회 이사
관심분야 : vulnerability analysis, insider threats, secure hardware architecture, homeland security