Native API 빈도를 이용한 DoS 탐지 기법

황현진*, 두선정*, 배성재**, 김낙훈*
*동덕여자대학교 컴퓨터학과
**고려대학교 정보경영공학전문대학원
email : 20051438@dongduk.ac.kr

DoS detection using a Native API Frequency

Hyun-Jin Hwang*, Sun-Jeong Doo*, Seong-Jae Bae**, Nak-Hoon Kim*
*Dept of Computer Science, Dong-Duk Women's University
**Graduate School of Information Management and Security, Korea University

요 약

네트워크가 광범위 하게 발달함에 따라 DoS 공격 기법은 더욱 다양해지고 있고 지능화되고 있다. 따라서 네트워크에 기반한 DoS 공격의 탐지는 더욱 어려워지고 있다. 본 논문에서는 이러한 DoS 공격에 대해 호스트에 기반한 Native API 의 빈도수를 이용한 침입 탐지 매커니즘에 관하여 기술한다.

1. 서론

최근 인터넷 보안 체계를 공격하는 경향은 한 프로세스가 시스템 또는 네트워크 자원을 모두 독점하거나 소비하게 하여 시스템이 다른 프로세스의 서비스를 제공받지 못하도록 하는 서비스 거부 공격 (Denial of Service) 이 증가 하고 있다. 특히 최근에는 많은 수의 호스트들에 패킷을 범람시킬 수 있는 Denial of Service (DoS) 공격용 프로그램들이 분산 설치되어 이들이 서로 통합된 형태로 어느 목표 시스템 또는 네트워크에 대하여 일제히 데이터 패킷을 범람시켜서 그 표적 시스템의 성능저하 및 시스템 마비를 일으키는 분산 서비스 거부 공격 (Distributed DoS) 이 많이 이루어 지고 있다.

네트워크가 DoS 공격에 취약한 원인은 공격이 여러 개의 소스에서 이루어지고 특정소스에서는 매우 적은 수의 패킷을 사용 하고 또 TCP, UDP, ICMP 등 다양한 형태의 패킷을 공격에 이용하기 때문에 공격자의 추적은 물론, 탐지와 방어에 어려움이 있다는 것이다[1]. 따라서 네트워크 계층에서의 트래픽 분석을 통한 이상탐지에는 한계가 있다.

본 논문에서는 윈도우 환경에서 DoS 공격에 의한 Native API 를 추출하고, 추출된 Native API 의 빈도와 정상 행위시 발생한 Native API 의 빈도수 분석을 통해 DoS 공격 여부를 판단한다. 이를 기반으로 윈도우 시스템에서 DoS 공격에 대하여 호스트에 기반한 침입 탐지 가능성을 제시할 것이다.

본 논문의 구성은 다음과 같다. 제 2 장 관련 연구에서는 DoS 공격의 기법과 Native API 에 대해 기술한다. 제 3 장에서는 DoS 공격과 정상 행위시 발생한 Native API 를 추출하여 빈도수 분석을 통한 DoS 공격 탐지 기술을 제안한다. 마지막으로 제 4 장에서는 본 연구의 의의 및 결론과 향후 연구 과제를 논의한다.

2. 관련 연구

2.1. SYN Flooding 공격기법

SYN Flooding 공격은 TCP 의 연결 설정 과정을 그대로 따르며 많은 운영 체제에서 TCP 를 구현하는데 필요한 과정을 악용한 공격이기 때문에 이 공격으로부터 완전히 해방되는 방법은 없는 것이 현실이다[2].

DoS 공격은 다른 공격들과 달리 시스템의 관리자 권한 획득, 시스템에 있는 데이터의 파괴 등을 행하지 않으면서 서비스를 사용할 수 없게 만든다. DoS 공격은 로그를 남기는 공격이 아니기 때문에 문제가 발생했을 때 추적하기 어려우며, 때문에 이를 해결하기가 어렵다는 문제점이 있다. 또한 공격 방법이 정형화되어 있지 않으며, 경우에 따라 공격 방법은 다양하게 존재한다. 뿐만 아니라 서버의 성능과 네트워크 구현 방법에 따른 차이에 기인하여 DoS 공격을 받는 시스템마다 그 결과가 다르게 나타날 수 있다.

인터넷 위협의 전형으로 DoS/DDoS 중에서도 SYN Flooding 공격은 특정 시스템에 불법적인 권한을 얻는 능동적인 방법이 아니라 시스템이 정상적으로 동작을 할 수 없게 만드는 다소 수동적인 방법으로 서비스 거부 공격 중의 하나이다.

이 공격은 TCP 가 데이터를 보내기 전에 연결을 맺어야 하는 신뢰성 연결 지향 방식으로, 그 원인이 시스템의 취약성에 바탕을 둔 것이 아니라 TCP/IP 프로토콜 오류에 있기 때문에 여전히 그 취약점은 해결되지 않은 채 남아 있다[3].

즉, SYN Flooding 은 TCP 의 연결 과정인 Three-way handshaking 를 이용하여 공격자가 대상시스템에 발신지 IP 주소를 spoofing 하여 SYN 패킷을 특정 포트로 전송하게 되면 이 포트의 대기 큐를 가득 차게 하여 이 포트에 들어오는 연결요청을 큐가 빌 때까지 무시하도록 하는 방법이다[4].

최근에 SYN Flooding 공격 방법은 더욱 진화하여, 목적지 IP 주소 보다는 발신지 IP 주소를 다양하게 변조하면서 폭주하는 SYN 패킷들을 감염된 호스트의 성능보다 더욱 빠른 속도로 발생시킨다. 그 이유는 특정 호스트에 접속하여 2 차 감염을 시키고자 하는 것 보다 오로지 공격을 통한 네트워크 무력화만이 목적이기 때문이다.

2.2. Native API

Windows 운영체제의 Application Programming Interfaces (API) 는 Dynamic Link Library (DLL) 에 포함되어 있는 프로그램 가능한 함수들이며, 사용자 모드 또는 커널 모드에서 동작한다. 커널 모드에서 동작하는 Native API 는 WIN32 API 중에서 OS 에서 지원하는 Subsystem 에서 NT Executive (Kernel) 의 도움을 받고자 할 때 사용되는 함수들의 모임이다[5].

NT 계열 운영체제에서는 다른 운영체제에서 작성된 애플리케이션의 호환성을 유지하기 위하여 다양한 서브시스템 (Win32, POSIX, OS/2, DOS/WoW 등) 들을 제공해준다[6]. 각각의 서브시스템은 다른 운영체제를 에뮬레이션 해주며 커널의 도움이 필요할 때 별도의 함수들을 이용한다. Native API 는 모든 서브시스템에게 동일한 인터페이스와 기능을 제공하며, 이로 인하여 다른 운영체제의 어플리케이션들은 서브시스템에 의존하여 NT 계열 운영체제에서 실행이 가능한 것이다.

윈도우 운영 체제에는 Native API 에 관한 데이터를 수집해 주는 Audit 항목이 존재하지 않기 때문에 시스템에서 발생되는 Native API 데이터를 수집하기 위해서는 커널 후킹 기법이 필요하다. 커널 후킹 기법에는 System Service Dispatch Table (SSDT) 를 이용하는 기법과 Interrupt Descriptor Table (IDT) 를 이용하는 기법, 그리고 디바이스 드라이버 오브젝트의 Major Function 후킹 기법이 있다[7].

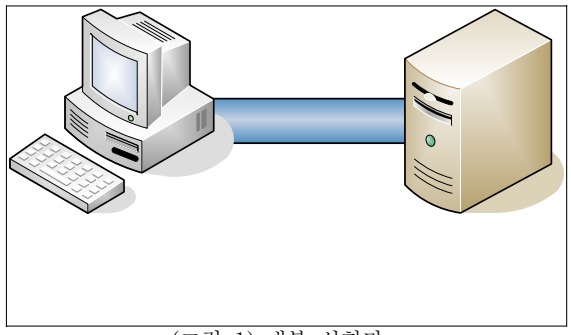
IDT 커널 후킹 기법은 인터럽트 핸들러의 주소를 담고 있는 IDT 를 변조하여 API 정보를 얻는 방법으로. 모든 인터럽트를 제어하기 위해서는 IDT 를 후킹 해야 한다.

3. DoS 탐지 기법 제안

본 논문에서는 DoS 공격 실험을 위하여 SYN Flooding 공격 기법을 사용하였다. 수집된 SYN Flooding 공격 코드는 두 가지 종류의 코드로 분류되며, 실험을 위해 수집된 정상 코드는 20 개로 분류된다. 수집된 SYN Flooding 공격 코드는 아래 그림과 같은 실험용 내부망에서 실행하였고, IDT 커널 후킹 기법을 이용하여 Native API 정보를 추출하였다.

<표 1> SYN Flooding 코드와 정상 코드의 분류

SYN Flooding	정상 코드	
Juno.c	Acrobat_reader	Msword
	Alftp	Nero
	Alsee	Powerpoint2007
	Alyac	Remote connect
	Alzip	Starcraft
syn_flooding.c	Daemon	Telnet
	Excel2007	Ultraedit
	Gomplayer	V3
	Hwp	Windows update
	Msaccess	Windows moviemaker



(그림 1) 내부 실험망

수집된 Native API 정보는 아래와 같은 수식을 적용하여 빈도 값을 추출한다. 각각의 빈도 값은 0 에서 1 사이의 값을 가지도록 표준화한다.

W_j : j 번째 DoS 코드/정상코드에서 발생된 전체 Native API 의 빈도수
C_{ij} : j 번째 DoS 코드/정상코드에서 발생된 i 번째 Native API 의 빈도수
F_{ij} : j 번째 DoS 코드/정상코드에서 발생된 i 번째 Native API 의 표준 빈도 값
V_i : i 번째 Native API 값의 발생 빈도 값
A : 수집된 DoS 코드/정상코드의 개수
B : Native API 종류의 수

$$V_i = (\sum_{j=1}^{A} S_{ij}) / A \quad (i = 1, \cdots, B)$$

각 Native API 발생 빈도를 0 에서 1 사이의 값을 가지도록 표준 빈도 값을 구한다.

$$F_{ij} = C_{ij} / W_j \qquad (i = 1, \cdots, B) \quad (j = 1, \cdots, A)$$

다음 표는 DoS 공격코드와 정상코드에 대한 Native API 표준 빈도 값을 추출한 결과로, DoS 공격코드의 표준 빈도 값이 가장 높은 Native API 상위 10 개를 기준으로 분류하였다.

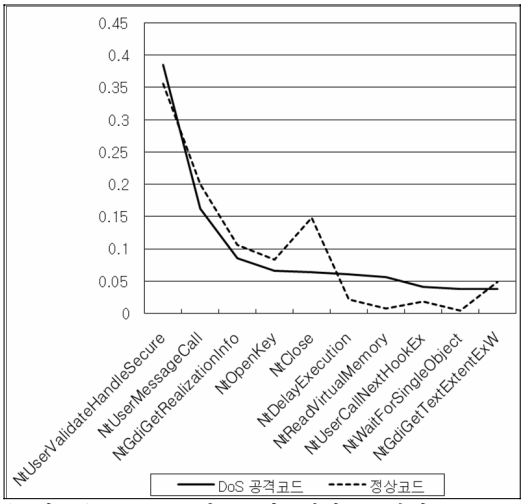
<표 2> DoS 공격코드의 Native API 표준 빈도 값

	Native API	Frequency
1	NtUserValidateHandleSecure	0.385622608
2	NtUserMessageCall	0.162652163
3	NtGdiGetRealizationInfo	0.085872975
4	NtOpenKey	0.066045066
5	NtClose	0.063886731
6	NtDelayExecution	0.060534116
7	NtReadVirtualMemory	0.056836168
8	NtUserCallNextHookEx	0.041900486
9	NtWaitForSingleObject	0.038375205
10	NtGdiGetTextExtentExW	0.038274483

<표 3> 정상코드의 Native API 표준 빈도 값

	Native API	Frequency
1	NtUserValidateHandleSecure	0.356596449
2	NtUserMessageCall	0.200804932
3	NtGdiGetRealizationInfo	0.107038877
4	NtOpenKey	0.083633042
5	NtClose	0.149255009
6	NtDelayExecution	0.02206428
7	NtReadVirtualMemory	0.007735343
8	NtUserCallNextHookEx	0.019152823
9	NtWaitForSingleObject	0.004766798
10	NtGdiGetTextExtentExW	0.048952446

각각의 Native API 표준 빈도 값을 기반으로 DoS 공격코드와 정상코드간의 빈도 차이 분석 결과는 다음 그래프와 같다.



(그림 2) DoS 공격코드와 정상코드간의 Native API 표준 빈도 분포 그래프

실험을 통해 DoS 공격코드와 정상코드의 호스트에 기반한 Native API 의 표준 빈도 값이 차이가 있음을 확인하였다. 제안된 기법을 이용하면 DoS 탐지를 위하여 별개의 애플리케이션을 이용하지 않더라도, 윈도우 내부의 함수와 커널 명령을 분석하여 시스템 내부에 API 를 이용한 DoS 공격 탐지 효율을 더욱 높일 수 있을 것이다. 또한 지속적인 패턴 입력과 데이터 수량의 증가 및 시스템 성능 향상에 의한 Native API 분석 범위의 확장은 DoS 탐지 효율의 정확도를 더욱 향상시킬 것으로 예상된다.

5. 결론

본 논문에서는 DoS 공격에 대해 커널 단위에서 발생하는 호스트 기반 정보인 Native API 를 IDT 커널 후킹 기법을 이용하여 추출하였다. 수집된 Native API 정보는 정상 행위시 수집된 Normal Native API 정보와 표준 빈도 값을 비교하였다.

그 결과 Native API 의 표준 빈도 값을 기반으로, 발생한 API Call 의 시그너처를 이용하여 특정 DoS 공격코드의 범주를 확인하고, 이에 따른 탐지 효율을 향상시킬 수 있는 가능성을 발견할 수 있었다.

본 연구를 기반하여 추후에는 진화되고 있는 다양한 DoS 공격에 대한 가상 실험을 통해 DoS 공격기법에 대한 Native API 의 표준화 빈도와 범주를 일반화시킬 수 있도록 해야 할 것이다. 또한 커널 단위에서의 DoS 공격 탐지를 위한 분석도구가 지속적으로 연구되어야 할 것이다.

참고문헌

[1] 정철현, 변대용, “트래픽 분석을 통한 서비스 거부공격 추적”, http://www.certcc.or.kr/paper/tr2003/030115-DoS.pdf, 2003.1.

[2] Lau, F. Rubin, S.H. Smith, M.H, Trajkovic L, "Distributed denial of service attacks" Systems Man and Cybernetics, 2000 IEEE International Conference on, Volume: 3, pp. 2275-2280 vol.3, 8-11 Oct. 2000

[3] CERT/CC: Computer Emergency Response Team Coordination Center (Reporting Center for Internet Security Problem). CERT Advisory CA-1996-21

[4] 고규만, “Distributed Reflection Denial of Service”, http://www.kisa.or.kr/Critical_Information_Infrastructure/data/tech_data/tech_data_20020802_DRDoS.pdf , 2002.

[5] Mark Russinovich, “Inside Native API”, www.sysinternals.com, 2004, Visited 2007

[6]강태우, 조재익, 정만현, 문종섭, “API call 의 단계별 복합분석을 통한 악성코드 탐지”, 한국정보보호학회, 정보보호학회논문지 제 17 권 제 6 호, pp. 89 ~ 98, 2007. 12.

[7] Greg Hoglund, James Butler, Rootkits: Subverting the Windows Kernel, Pearson Education, Inc, 2006