Windows 네트워크 서비스 함수 수준의 취약점 검증

*박정민 *송용호 **최영한 **김형천 **홍순좌
*한양대학교 정보통신대학원 **국가보안기술연구소
{jmpark, yhsong}@enc.hanyang.ac.kr {yhch, khche, hongsi}@etri.re.kr

Function-Level Verification of Security Vulnerabilities in Windows Network Services

*Jung Min Park *Yong Ho Song **Young Han Choi **Hyoung Chun Kim **Soon Jwa Hong
*Hanyang University **National Security Research Institute

요 약

최근 Windows 관련 취약점으로 인한 피해는 매우 커지고 있다. 그러나 그에 대한 조치는 사후처리식의 수동적 형태였다. 따라서 본 논문에서는 Windows 관련 취약점에 능동적으로 대처하기 위하여 Windows 관련 취약점 공격 유형을 분석하고, 그 정보를 바탕으로 주요 취약점의 동작 원리를 분석하여 사전에 예상되는 취약점의 취약성 유무를 검증하는 기법을 제안한다.

1. 서 론

오늘날 Windows는 데스크탑 운영체제로 가장 널리 사용되고 있다. 이에 따라 Windows 운영체제와 직접 연동이 용이한 Windows 기반 인터넷 서비스의 수와 범위는 점차 증가하고 있다. 이와 동시에 악의적인 취약점 공격도 증가하여 보안상 많은 취약점이 발견되고 있으며, 이로 인한 피해가 점차 증가하고 있다.

Windows 시스템은 Linux 운영체제와 달리 소스코드가 공개되어 있지 않아 보안상의 취약점을 파악하기가 용이하지 않다. 만일 취약성이 의심되는 부분이 발견될 경우 Microsoft사에 이를 수정하기 위한 패치를 제공할 때까지 사용자는 별다른 조치를 취하기 어려운 것이 현실이다.

Windows 기반 서버 시스템에 대한 성공적인 공격이 이루어진 경우 그 피해 정도는 제공하고 있는 서비스의 유형에 따라 큰 차이를 보일 수 있으며, 극단적인 경우 국가 시스템을 마비시킬 수 있을 정도 규모가 될 수 있다[1].

본 논문에서는 취약점에 대한 능동적인 대처를 위해 기존의 주요 취약점을 파악하고 그 취약점들의 공격 동작 원리를 이용하여 예상 취약점을 검증하는 기법을 제안한다. 논문에서 제시되는 기법을 통해 검증되는 취약점에 대해서는 Microsoft를 통하여 패치 형태로 시스템에 보완 및 적용될 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 기존 취약점들 중 보편화된 취약점인 버퍼오버플로우(Buffer Overflow)의 동작 원리를 설명한다. 3절에서는 취약점 검출방법에 대해 살펴보고, 4절에서는 취약점을 분석하기 위해 사용되는 디버깅 툴에 대해 설명한다. 5절에서는 예상 취약점의 취약성 유무를 검증하기 위해 실험을 통하여 취약점 검출 방법 및 취약점 분석 툴의 적용 방법을 제안한다. 마지막으로 6절에서는 본 논문의 연구결과 및 향후의 연구 방향를 설명한다.

2. 기존 취약점 동작 원리 분석

오늘날 전체 네트워크 보안 침입 사례 중 50%를 차지할 만큼 보편화된 취약점이 버퍼오버플로우(Buffer Overflow)이다[2]. 버퍼오버플로우는 주어진 버퍼의 크기보다 큰 데이터를 버퍼에 기록하려 할 때, 버퍼의 크기나 범위를 고려하지 않고 데이터를 기록함으로써 버퍼와 인접한 영역에 정의되어 있는 다른 자료 구조의 값을 비정상적으로 변경하게 되는 현상을 말한다. 데이터를 저장할 때 크기 또는 경계에 대한 검사를 자동적으로 수행하지 않는 언어를 이용하여 프로그램을 작성할 경우 이와 같은 문제에 취약하다.

버퍼오버플로우 공격의 가장 단순한 형태는 스택 영역에 저장된 데이터에 대해 비정상적인 변형을 가하는 것이다. 함수 호출 시 스택은 함수 호출 전 상태로 복귀하는데 필요한 복귀 주소 및 상태 정보를 저장하는 역할을 수행한다. 또한 호출된 함수에서는 일시적으로 사용하는 지역 변수를 정의하는 목적으로 스택을 사용한다. 이 때 동일한 스택을 사용하여 두가지 서로 다른 목적을 갖는 정보를 인접한 영역에 저장하게 됨에 따라 지역 변수로 인하여 발생하는 버퍼오버플로우 현상으로 말미암아 복귀 주소 및 상태 정보의 값이 비정상적으로 변경될 수 있다. 스택에서의 버퍼오버플로우 공격은 복귀 주소 영역에 공격 코드의 시작 주소가 저장되도록 인위적인 오버플로우를 만듦으로써 이전 함수로의 복귀 시에 공격 코드가 실행될 수 있도록 한다. 이 때 공격 코드는 일반적으로 시스템 관리자 권한의 획득을 시도한다. 그리고 이에 성공할 경우, 향후 용이한 접근을 위한 사용자 계정 또는 백도어를 만들어 두기도 한다.

3. 취약점 검출 방법

취약점의 존재가 의심스러운 시스템에 대해 취약점을 검출하는 방법에는 크게 Manual testing, 퍼징(Fuzzing),

실시간 분석 기법(Runtime analysis)등이 있다[3].

본 논문에서는 취약점 존재가 의심스러운 시스템에 퍼징을 이용하여 취약점 존재 유무를 파악하고 실시간 분석 기법으로 정밀하게 분석하여 취약점을 검출하는 단계별 분석 방법을 이용한다.

3.1 실시간 분석 기법

실시간 분석 기법 기법은 디버깅 환경에서 취약점 검증 대상 시스템을 실행하고 실행 시간 동안 코드를 검사하는 방법이다. 디버깅 툴을 사용하여 사용자가 직접 프로그램의 흐름에 따라 변화하는 데이터를 추적해가면서 의심스러운 현상을 분석해 나간다.

이 기법은 많은 시간을 소요하고, 매우 긴 코드 라인 수 때문에 취약점 원인 규명을 위한 정밀한 분석을 위해 사용한다[3].

3.2 퍼징 기법

퍼징은 취약점 검증 대상 시스템에 대한 입력의 패턴을 인위적으로 변화시킴으로써, 이로 인한 시스템의 에러 메시지나 오동작으로 취약성 유무를 판단하는 기법이다.

일반적으로 예상되는 취약 함수의 파라미터로 임의의 입력을 전달해 주고 그 함수를 호출하는 응용 프로그램의 에러 메시지나 오동작으로 취약점의 유무를 판단한다. 자동화된 퍼징을 위해 대상 시스템에 맞게 퍼징 도구를 제작하여 사용한다[4].

4. 취약점 분석 도구

취약점 분석 및 시스템 프로그램 디버깅 등에 흔히 사용되는 도구로는 WinDBG, SoftICE, OllyDBG를 들 수 있다.

WinDBG는 Windows 에서 사용되는 대표적인 커널 디버거로 Windows NT/95/98/2000/XP 에서 Win32 응용 프로그램을 디버깅하는데 사용한다. 소스 수준 디버깅을 수행할 수 있으며 파일의 내용을 읽을 수 있고 마이크로소프트사의 서버에 접속하여 운영 체제 내부 코드에 대한 심볼 이름을 전송받아 보여줄 수 있다.

Device driver 및 user mode native application 에서도 사용할 수 있는 WinDBG는 운영체제의 내부 자료구조에 관한 정보를 보여주기 때문에 커널 디버깅과 애플리케이션 디버깅에 모두 이용 가능하다.

하지만 커널 코드에 대한 분석을 위해서는 디버거용 호스트와 타겟 호스트가 구성되는 별도의 시스템으로 원격 디버깅 방식을 사용 해야 한다는 단점이 존재한다[5].

SoftICE는 WinDBG와 마찬가지로 소스 수준에서 시스템 디버깅이 가능한 도구이다. 기존의 디버거는 분산 환경에서 다중 쓰레드로 동작하는 애플리케이션을 처리하지 못하였다. SoftICE를 이용하면 거의 모든 형태의 코드 디버깅이 가능하다. 또한 인터럽트 루틴과 디바이스 드라이버까지도 지원한다. Symbol Loader Utility는 컴파일 된 모듈의 디버그 정보를 .NMS 심볼 파일로 번역한 뒤 이 정보와 소스 파일을 SoftICE로 로딩한다.

WinDBG와 달리 단일 컴퓨터에서 커널 디버깅이 가능하고, 전체 시스템 뷰 및 제어 기능은 기존의 SDK/DDK 툴이 가진 제약에서 벗어나 부트 드라이버에서 애플리케이션까지 시스템 수준의 프로그램을 디버깅할 수 있다.

WinDBG에서는 어셈블리 코드의 단계별 분석을 빠른 속도로 진행할 때 낮은 성능의 사용자 인터페이스로 인해 디버깅 정보 분석이 불편하다. 이에 비해 OllyDBG는 우수한 사용자 인터페이스 성능으로 인해 디버깅 정보의 분석이 편리하여 디버깅 시간을 단축할 수 있다. 그리고 OllyDBG는 backtrace 기능이 있어 단계별 분석 중 이전 코드를 다시 보고 싶을 때 역추적이 가능하고, 원하는 위치에서의 상대 주소값을 더블클릭 한번으로 쉽게 얻을 수 있다[6].

따라서 본 논문에서는 실시간 분석 기법을 이용한 정밀 코드 분석을 위해 OllyDBG를 사용한다.

5. 예상 취약점의 취약성 유무 검증

5.1 예상 취약 함수 추출

취약점이 존재할 것으로 예상되는 함수를 추출하기 위해서는 악의적인 사용자가 공격 코드의 삽입할 수 있는 함수들을 파악한다. 우선 1) 네트워크 서비스 함수들 중에서 외부로부터 오는 데이터를 청취하는 함수들이 있다. 그 데이터는 공격 대상이 되는 시스템이 연결 요청이나 질의를 했을 때 받는 응답 메시지이거나 새로운 연결을 감지하기 위해 지속적으로 청취하고 있는 함수들이 받는 연결 메시지일 것이다. 이러한 함수들을 추출한다. 2) 추출된 함수들 중에서 버퍼를 파라미터로 사용하는 것들을 다시 추출한다. 이는 버퍼오버플로우를 일으키는 코드가 들어올 수 있는 부분이다. 이렇게 추출한 함수들은 버퍼오버플로우 취약점이 발생할 가능성이 잠재되어 있다.

본 실험에서는 기존의 취약점이 존재하는 것으로 판명된 함수들을 나열하고, 각각에 대해 유사한 기능을 수행하거나 입출력 파라미터를 갖는 함수를 MSDN 라이브러리에서 검색하여 예상 취약 함수로 선정한다.

5.2 취약성 진단

취약성 진단을 위한 대상 함수를 포함한 프로세스에 버퍼오버플로우 취약점을 일으킬만한 코드 삽입 동작을 행한다. 이러한 방법으로 퍼징 기법을 사용한다.

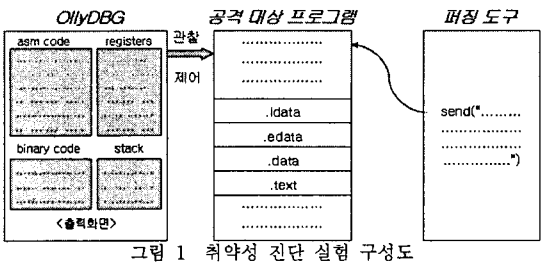

그림 1 취약성 진단 실험 구성도

본 실험에서는 취약성 진단의 과정을 보여주기 위해 퍼징 도구를 이용하여 취약점 공격 코드인 Code-Red 웜 바이러스[7]를 대상 시스템으로 전송함으로써 IIS 서버에 대한 공격을 실행하는 네트워크 서비스 공격 패킷 생성기 (Service Attack Packet Generator: SAPG)를 작성하여 실험하였다.

SAPG는 공격하려는 호스트에 HTTP 프로토콜을 이용해 Code-Red 웜 바이너리 코드를 전송하도록 제작한 프로그램이다. 공격 대상 시스템에서 청취하는 프로세스는 inetinfo.exe이다.

OllyDBG에 inetinfo.exe를 연결한 후 공격 시스템에서 SAPG를 실행하여 공격 대상 시스템으로 Code-Red 웜 바이너리 코드를 전송한다.

이후 공격 대상 시스템에서 OllyDBG를 살펴보면 그림 2와 같이 Access violation이 발생한다. Static array의 범위를 넘어서 메모리를 침범했을 때 발생하는 exception인 Access violation이 발생한 것으로 보아 시스템의 취약점이 공격으로 인해 노출된 것으로 판단할 수 있다.

Code-Red 공격에 의한 경우 OllyDBG를 이용해서 취약성을 검증하는 과정을 보이기 위해 Code-Red 웜 바이너리 코드가 실행되는 부분을 실시간 분석기법으로 추적하였다. 이 방법에 의하면 Code-Red 웜 바이러스는 ECX 레지스터를 통한 indirect procedure call 형태로 호출이 되며, 이 레지스터는 스택 내부로부터 복원되어 사용되므로 스택에서 웜에 의한 공격을 받았음을 알 수 있다. 그림 3는 ECX 레지스터 값이 변경된 후 바이러스 코드가 호출되는 모습을 보이고 있다[8].

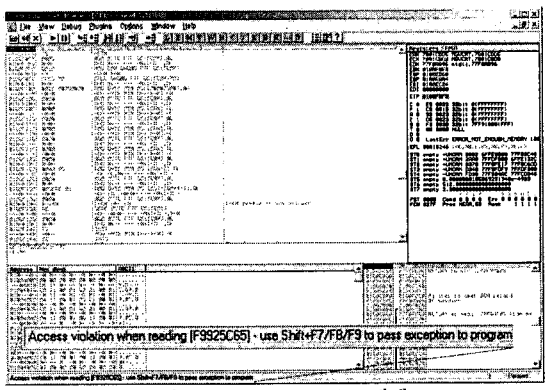

그림 2 Access violation 발생

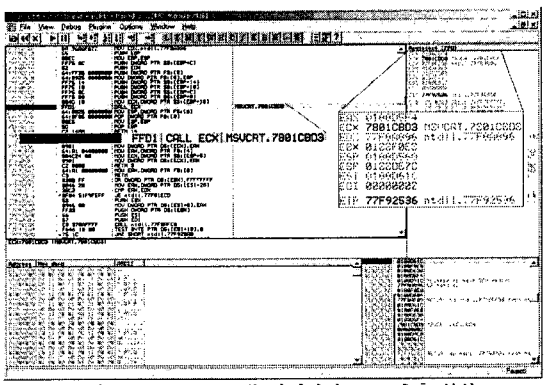

그림 3 Code-Red 웜 바이너리 코드 호출 부분

그림 4은 Code-Red 웜 감염 이후 서버의 네트워크 상태를 netstat 명령으로 살펴본 결과이다. 무작위로 추출된 IP의 80번 포트에 접속을 시도함을 확인할 수 있다. Code-Red 웜에 감염된 직후부터 웜이 계속해서 자신을 전파시키려고 시도하기 때문에 네트워크 사용량이 급격히 늘어난다.

6. 결론 및 향후 연구 방향

본 논문에서 제시하고 있는 네트워크 서비스 함수 수준의 취약성 검증 기법은 기존에 있는 취약점 검증기법을 응용하여 취약성 존재가 예상되는 네트워크 서비스 함수들의 취약성 유무를 판단하기 위한 하나의 방법이다. 이 기법을 이용하면 취약성이 존재할 것으로 예상되는 함수들을 검증해 볼 수 있다.

그러나 이 방법은 예상 취약성 함수를 사용하는 기존의 응용 프로그램이 없으면 직접 일일이 만들어 주어야 하는 불편함이 있다. 그리고 이것은 외부로부터 입력을 청취하는 네트워크 서비스 함수에 제한된 방법이다. 만약 다른 서비스 함수에 대해 취약성 유무 검증을 시도할 경우 다른 방법이 필요할 것이다. 따라서 보다 다양한 서비스 함수들의 취약성 유무 검증을 위해 다른 Windows 서비스의 특징과 취약성 검증을 위한 일반화된 기법에 관한 연구가 필요하다.

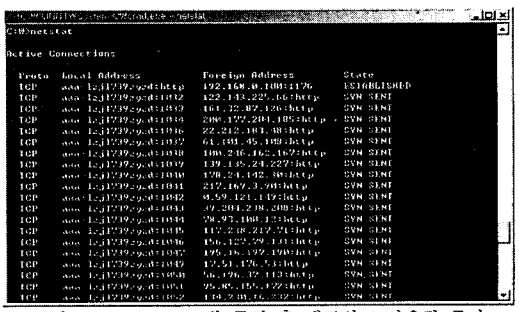

그림 4 Code-Red 웜 공격 후 네트워크 사용량 증가

참 고 문 헌

[1] 송관호, "인터넷대란과 대응방안", 한국통신학회지, 21권, 1호, 60-69쪽, 2004년

[2] Olatunji Ruwase, Monica S. Lam, "A Practical Dynamic Buffer Overflow Detector", In Proceedings of the Network and Distributed System Security (NDSS) Symposium, pages 159-169, February 2004.

[3] Halvar Flake, FX, "I shut up, you take it from here (how to become a hacker)", http://www. phenoelit.de/stuff/papers.html, 2004

[4] Michael Howard, James Whittaker, "Violating Assumptions with 퍼징", IEEE Security and Privacy, vol. 3, no. 2, pp. 58-62, March/April 2005

[5] Microsoft®, "Debugging Tools for Windows", http://www.microsoft.com/whdc/devtools/debugging/default.mspx, 2005

[6] Oleh Yuschuk, "OllyDbg Help Document"in OllyDbg v1.10, 2004

[7] eEye Digital Security®, "Code Red" Worm, http://www.eeye.com/html/Research/Advisories/AL20010717.html, July 17, 2001

[8] Sergio Alvarez, "Win32 Stack BufferOverFlow Real Life Vuln-Dev Process", Security Research & Development, IT Security Consulting, September 5, 2004