(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2008년12월24일
(11) 등록번호 10-0875997
(24) 등록일자 2008년12월18일

(51) Int. Cl.
H04L 12/28 (2006.01) *H04L 29/06* (2006.01)
G06F 17/00 (2006.01) *G06F 15/00* (2006.01)
(21) 출원번호 10-2007-0062325
(22) 출원일자 2007년06월25일
심사청구일자 2007년06월25일
(65) 공개번호 10-2008-0043209
(43) 공개일자 2008년05월16일
(30) 우선권주장
1020060111858 2006년11월13일 대한민국(KR)
(56) 선행기술조사문헌
KR1020060059759 A*
KR1020060059757 A
JP10269105 A
KR100391932 B1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
한국전자통신연구원
대전 유성구 가정동 161번지
(72) 발명자
정계옥
대전 유성구 관평동 대덕테크노밸리 쌍용스윗닷홈 409동 302호
홍순좌
대전 유성구 신성동 럭키하나아파트 101동 301호
(74) 대리인
권태복, 이화익

전체 청구항 수 : 총 5 항 심사관 : 전용해

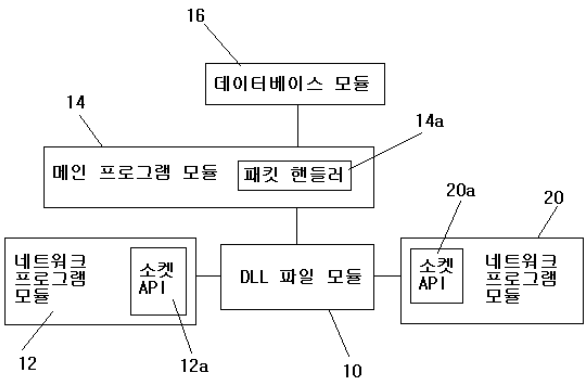
(54) 윈도우 소켓 응용프로그램 인터페이스 후킹을 이용한네트워크 퍼징 방법

(57) 요 약

본 발명은 윈도우 소켓 API(Application Programming Interface) 후킹을 이용한 네트워크 퍼징 시스템 및 그 방법에 관한 것으로, MS 윈도우 운영체제에서 동작하는 네트워크 프로그램을 대상으로 동적 링크 라이브러리 주입(Dynamic Linked Library injection; 이하, DLL 주입이라 함)을 통하여 네트워크 프로그램에 소켓 API 후킹 기능을 삽입함으로써, 네트워크 프로그램이 소켓 API 함수를 사용하여 상대편 네트워크 프로그램과의 송수신하는 패킷을 가로챈 다음, 그 패킷 내의 데이터에 다양한 퍼징 데이터 셋을 추가하여 해당 패킷을 조작하거나 비정상적인 패킷으로 만들어 송신함으로써 범용적인 프로토콜뿐만 아니라, 알려지지 않은 프로토콜에 대해서 네트워크 퍼징을 수행한다.

본 발명에 의하면, 종래의 프로토콜 분석과 퍼저 제작을 거치지 않고, 임의의 프로토콜을 사용하는 네트워크 프로그램을 이용하여, DLL 주입을 통한 소켓 API 후킹으로 네트워크 퍼징을 자유롭게 할 수 있을 뿐만 아니라, 네트워크 퍼징을 수행하기 위해 프로토콜 분석부터 퍼저의 제작까지 많은 인력과 시간이 소모되는 종래의 문제점을 완전 해소할 수 있다.

대 표 도 - 도1



특허청구의 범위

청구항 1

패킷 핸들러 모듈을 내장한 메인프로그램 모듈, DLL 파일 모듈 및 데이터베이스 모듈을 포함하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 시스템에 있어서,

윈도우즈 운영체제에서 네트워크 프로그램이 실행되면, 상기 메인프로그램 모듈이 상기 네트워크 프로그램에 상기 DLL 파일 모듈을 주입하는 제1 단계와;

상기 DLL 파일 모듈이 상기 메인프로그램 모듈의 명령에 따라 상기 네트워크 프로그램의 소켓 API를 후킹하는 제2 단계와;

특정 필드에 삽입가능한 퍼징 데이터 셋을 생성하여 보관하고 있는 데이터베이스 모듈이 상기 네트워크 프로그램의 송수신 패킷들을 데이터베이스로 저장하고 분석하는 제3 단계를 포함하는 것을 특징으로 하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법.

청구항 2

제1항에 있어서,

상기 DLL 파일 모듈의 주입은 CreateRemoteThread() 함수를 이용한 DLL 주입법으로 수행되는 것을 특징으로 하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법.

청구항 3

제1항에 있어서,

상기 DLL 파일 모듈은 상기 네트워크 프로그램의 송수신 패킷들을 차단하거나 또는 각 필드별로 분해하여 특정 필드에 임의의 데이터를 삽입, 변조하여 외부로 송신하는 것을 특징으로 하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법.

청구항 4

삭제

청구항 5

제1항에 있어서,

사용자가 판단하는 적정한 수준의 패킷들이 수집되면, 상기 메인프로그램 모듈이 상기 패킷 핸들러 모듈을 통해 비정상적인 패킷들을 생성한 후 외부로 송신함으로써 퍼징 대상 네트워크 프로그램에 문제점이 존재하는지를 판단하는 제4 단계를 더 포함하는 것을 특징으로 하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법.

청구항 6

제5항에 있어서,

상기 제4 단계는,

상기 데이터베이스 모듈이 내부에 저장하고 있던 퍼징 데이터 셋과 분석된 패킷의 데이터 영역에 관한 정보를 상기 메인프로그램 모듈내의 패킷 핸들러 모듈에 전달하는 과정과,

상기 패킷 핸들러 모듈이 상기 퍼징 데이터 셋을 상기 분석된 패킷의 데이터 영역에 적절하게 삽입하는 과정과,

상기 메인프로그램 모듈이 상기 분석된 패킷을 상기 DLL 파일 모듈로 전달하여 정상적인 소켓 API를 통하여 외부로 송신하도록 하는 과정을 포함하는 것을 특징으로 하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법.

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

<8> 본 발명은 네트워크 퍼징(Fuzzing)에 관한 것이며, 더욱 상세히는 윈도우 소켓 응용프로그램 인터페이스(Application Programming Interface; 이하, API라 함) 후킹을 이용한 네트워크 퍼징 방법에 관한 것이다.

<9> 퍼징(Fuzzing)은 특정 대상으로부터 취약점을 찾기 위해서, 다양한 입력을 만들어내어 특정 대상에게 주입하는 기법이다.

<10> 네트워크 퍼징은 네트워크로 연결된 프로그램이 사용하는 특정 프로토콜을 겨냥하여 모든 종류의 비정상적인 패킷들을 생성하여 주입하거나, 특정 프로토콜의 통신 순서를 교란함으로써 임의의 네트워크 프로그램에서 취약점을 찾는 보편화된 방법 중의 하나로서, 네트워크상의 프로토콜을 흉내 내어 다양한 패킷들을 네트워크 프로그램으로 전송하여 취약점이 존재하는 것을 테스트한다.

<11> 네트워크 퍼징을 위해서는 네트워크 프로그램이 사용하는 프로토콜을 준수하는 퍼저(Fuzzer)라고 하는 별도의 프로그램을 제작하여야만 한다.

<12> 일반적으로 네트워크 퍼징을 수행하는 퍼저들은 퍼징 작업을 위해, 먼저 목표가 되는 프로토콜을 사용하는 프로그램에 대해서, 프로토콜 분석을 선행한다. 즉, 프로토콜의 개략적인 통신 순서와 패킷구성 등에 대해 조사하여 그에 알맞은 네트워크 퍼징 순서 또는 패킷을 생성하는 과정을 거친다. 이 과정을 바탕으로 프로토콜에 적절한 퍼저를 생성하게 되는데, 프로토콜 분석에서부터 퍼저 제작까지의 과정이 실제 네트워크 퍼징을 하는 것보다 많은 노력과 시간을 소비하게 되어 있다.

<13> 현재 다양한 종류의 네트워크 퍼저들이 존재하고 있지만, 이들이 사용하는 방법은 기존에 잘 알려진 프로토콜을 그대로 에뮬레이트하는 것으로, 이미 분석된 프로토콜을 조작해서 사용할 수 있도록 만들어 놓은 것이 대부분이다.

<14> 또한, 알려지지 않은 프로토콜에 대한 퍼징 작업을 수행하려고 하면, 먼저 프로토콜 분석을 선행하여, 프로토콜에 대한 통신 순서와 패킷 구조 등을 모두 새로이 작성해야만 한다.

<15> 다른 한편, 네트워크 퍼저 프레임 워크 형태의 퍼저 제작 도구들이 역시 많이 알려져 있으며, 그러한 제작 도구들은 새로운 프로토콜에 대한 퍼징을 위해, 정형화된 패킷 송수신 기법을 제공하고 있지만, 전적으로 프로토콜 분석은 사용자의 몫이고, 패킷 구조의 생성 또한 사용자에게 전부 의존한다. 결국 현재 출시되어 있는 네트워크 퍼저들은 잘 알려진 프로토콜에 대해서만 퍼징 작업을 수행할 수 있다. 알려지지 않은 프로토콜에 대한 퍼징 작업은 반드시 새로운 형태의 퍼저가 제작되어져야만 가능하다.

발명이 이루고자 하는 기술적 과제

<16> 본 발명은 상기한 종래의 문제점을 해결하기 위한 것으로서, 본 발명의 목적은 본 발명은 MS 윈도우 운영체제에서 동작하는 네트워크 프로그램을 대상으로 동적 링크 라이브러리 주입(Dynamic Linked Library injection; 이하, DLL 주입이라 함)을 통하여 네트워크 프로그램에 소켓 API 후킹 기능을 삽입함으로써, 네트워크 프로그램이 소켓 API 함수를 사용하여 상대편 네트워크 프로그램과의 송수신하는 패킷을 가로챈 다음, 그 패킷 내의 데이터에 다양한 퍼징 데이터 셋을 추가하여 해당 패킷을 조작하거나 비정상적인 패킷으로 만들어 송신함으로써 범용적인 프로토콜뿐만 아니라, 알려지지 않은 프로토콜에 대해서 네트워크 퍼징을 수행하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법을 제공하는 것이다.

발명의 구성 및 작용

<17> 상기와 같은 본 발명의 목적을 달성하기 위하여 본 발명에 따른 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법은, 패킷 핸들러 모듈을 내장한 메인프로그램 모듈, DLL 파일 모듈 및 데이터베이스 모듈을 포함하는 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 시스템에 있어서, 윈도우즈 운영체제에서 네트워크 프로그램이 실행되면, 상기 메인프로그램 모듈이 상기 네트워크 프로그램에 상기 DLL 파일 모듈을 주입하는 제1 단계와; 상기 DLL 파일 모듈이 상기 메인프로그램 모듈의 명령에 따라 상기 네트워크 프로그램의 소켓 API를 후킹하는 제2 단계와; 상기 데이터베이스 모듈이 상기 네트워크 프로그램의 송수신 패킷들을 데이터베이스로 저장하고 분석하는

제3 단계를 포함하는 것을 특징으로 한다.

<18> 일례로, 상기 DLL 파일 모듈의 주입은 CreateRemoteThread() 함수를 이용한 DLL 주입법으로 수행된다.

<19> 상기 DLL 파일 모듈은 상기 네트워크 프로그램의 송수신 패킷들을 차단하거나 또는 각 필드별로 분해하여 특정 필드에 임의의 데이터를 삽입, 변조하여 외부로 송신하는 것을 특징으로 한다.

<20> 바람직하게, 상기 데이터베이스 모듈은 상기 특정 필드에 삽입가능한 퍼징 데이터 셋을 생성하여 보관하고 있다.

<21> 사용자가 판단하는 적정한 수준의 패킷들이 수집되면, 상기 메인프로그램 모듈이 상기 패킷 핸들러 모듈을 통해 비정상적인 패킷들을 생성한 후 외부로 송신함으로써 퍼징 대상 네트워크 프로그램에 문제점이 존재하는지를 판단하는 제4 단계를 더 포함한다.

<22> 상기 제4 단계는, 상기 데이터베이스 모듈이 내부에 저장하고 있던 퍼징 데이터 셋과 분석된 패킷의 데이터 영역에 관한 정보를 상기 메인프로그램 모듈내의 패킷 핸들러 모듈에 전달하는 과정과, 상기 패킷 핸들러 모듈이 상기 퍼징 데이터 셋을 상기 분석된 패킷의 데이터 영역에 적절하게 삽입하는 과정과, 상기 메인프로그램 모듈이 상기 분석된 패킷을 상기 DLL 파일 모듈로 전달하여 정상적인 소켓 API를 통하여 외부로 송신하도록 하는 과정을 포함하여 수행된다.

<23> 이하, 본 발명의 실시예를 첨부한 도면을 참조하여 더욱 상세하게 설명한다.

<24> 먼저, 본 발명의 실시예를 구체적으로 설명하기에 앞서, 본 발명에 적용되는 DLL 주입과 API 후킹에 관하여 설명하면 다음과 같다.

<25> DLL 주입은 MS 윈도우 운영체제에서 동작하는 프로그램에 임의의 동적 라이브러리 파일을 주입시키는 것으로, 프로그램에 특정한 기능을 추가하고자 할 때 사용하는 기법이다.

<26> API 후킹은 MS 윈도우 운영체제에서 정의된 특정 라이브러리 함수들을 사용자가 원하는 방식으로 바꾸고자 할 때 사용되는 기법으로, 어떤 응용프로그램에서 API가 호출되는 것을 가로채서 별도로 개발한 프로그램의 함수가 실행되도록 하는 메커니즘을 말한다. 어떤 윈도우즈용 응용프로그램을 개발하기 위해서 여러 종류의 언어나 도구들을 사용할 수 있다. Visual Basic, Visual C++, Delphi, C++ Builder, Power Builder 등 수많은 언어와 도구들이 존재한다. 이런 도구들을 사용해 작성된 응용프로그램들에서 어떤 작업을 수행하기 위해서 사용하는 방법은 서로 다를지라도 그 내부로 들어가 보면 결국에는 윈도우즈 운영체제에서 제공하는 API를 호출하게 된다. 예를 들어 화면에 문자열을 출력하려고 한다면 MFC(Microsoft foundation class)에서는 CDC::DrawText를 사용하고, Delphi에서는 TCanvas::TextOut을 사용한다. 개발도구마다 사용하는 형식은 이렇게 다르지만 결국에는 윈도우즈 운영체제가 제공하는 TextOutA/W API를 호출한다. TextOutA/W API는 gdi32.dll에 구현되어 있는 함수이고, 윈도우즈 운영체제가 제공하는 API이다. 윈도우즈는 기본적으로 3개의 DLL(kernel32.dll, user32.dll, gdi32.dll)에서 대부분의 API를 구현하여 제공하고, 응용프로그램은 실행 중에 자신의 프로세스 주소 공간으로 이들 DLL을 매핑한 후 사용한다.

<27> API 후킹은 이렇게 사용되어 지는 DLL 내부의 API 함수를 가로채는 것을 지칭한다. 후킹되는 대상은 반드시 윈도우즈의 API일 필요는 없고, 단지 DLL에서 구현하여 제공하는 함수이면 API 후킹이 가능하다.

<28> API 후킹은 어떤 언어로 개발된 프로그램에도 적용될 수 있으며 디버깅이나 버그 추적 작업, 모니터링 작업 등에 사용할 수 있으며, 소스 코드가 없는 프로그램에 기능을 추가하기 위한 용도로도 사용할 수 있다.

<29> API 후킹을 하기 위해서는 크게 세 가지 방법, 즉 레지스트리(Registry)를 이용하는 방법, 시스템 전역 윈도우즈 후크(System-wide Windows Hooks)를 이용하는 방법, 그리고 'CreateRemoteThread()' 함수를 이용한 DLL 주입(injection) 방법이다.

<30> 상기 레지스트리(Registry)를 이용하여 API 후킹을 하는 것은 극히 제한된 경우에만 가능하다. 이 방법은 USER32.DLL을 사용하는 애플리케이션에 대해서만 적용할 수 있다. 왜냐하면 USER32는 특정한 레지스트리 키의 값을 읽어 DllMain 안에서 이들 DLL에 대해 'LoadLibrary()' 함수를 호출하기 때문이다. 다른 제약은 이 메카니즘이 NT와 2K 운영체제에 대해서만 지원된다는 것이다. 게다가 API 후킹을 활성화하고 비활성화하기 위해서는 윈도우즈 운영체제를 리부팅해야 하며, 모든 응용프로그램에 대해서 API 후킹을 수행하므로, 오버헤드가 발생할 수 있다.

<31> 상기 시스템 전역 윈도우즈 후크(System-wide Windows Hooks)를 이용하는 방법은 목표로 하는 프로세스에 DLL을

침투시키는 매우 인기가 있는 기술이다. MSDN(MicroSoft Developer Network)에 자세히 설명되어 있듯이, 후크는 시스템 상의 메시지 처리 메커니즘을 가로채는 것이다. 응용프로그램은 사용자 정의 필터 함수를 설치하여 시스템에서의 메시지 교환을 감시하고 특정한 종류의 메시지가 목표로 하는 윈도우에 도달하기 전에 처리할 수 있다. 여기서 말하는 윈도우즈 후크는 API 후크와 전혀 다른 용어로서 MSDN에서 설명하고 있는 메시지 후크를 뜻한다. 상기 시스템 전역 윈도우즈 후크를 이용하는 방법은 레지스트리 메커니즘과 달리 후크 서버가 DLL이 더 이상 필요 없다고 결정하여, 'UnhookWindowsHookEx()' 함수를 호출하게 되면 DLL을 언로드할 수 있다. 그러나, 윈도우즈 후크는 시스템이 수행하여야 하는 메시지의 처리 과정을 증가시키므로 전체 시스템의 성능을 확연하게 저하시키는 단점을 가진다. 또한 시스템 전역 후크를 디버그하는 것은 많은 노력을 필요로 한다.

<32> 상기 'CreateRemoteThread()' 함수를 이용한 DLL 주입(injection) 방법은 가장 보편화된 방법으로 리모트 쓰레드를 이용하여 DLL을 침투시키는 것이다. 이 방법은 제프리 릿처(Jeffrey Ritcher)의 아이디어이며, 그가 쓴 문서인 "Load Your 32-bit DLL into Another Process's Address Space Using INJLIB"에 잘 설명되어 있다. 어떤 프로세스도 LoadLibrary() API를 호출하여 DLL을 동적으로 로드할 수 있다. 문제는 어떻게 프로세스의 쓰레드에 대해 전혀 접근을 하지 못하면서 외부의 프로세스가 적절하게 'LoadLibrary()' 함수를 호출하도록 만드는가 하는 것이다. 해답은 하나의 리모트 쓰레드를 생성하는 'CreateRemoteThread()' 함수이다. 여기서 약간의 속임수가 필요하다. 쓰레드 함수의 원형을 보면, 이 함수의 포인터가 매개변수(LPTHREAD_START_ROUTINE)로 'CreateRemoteThread()' 함수에 넘겨진다:

<33> DWORD WINAPI ThreadProc(LPVOID lpParameter);

<34> 그리고, 여기 LoadLibrary API의 원형이 있다.

<35> 이 두 함수들은 동일한 형태를 지닌다. 이 함수들은 같은 호출 규약 WINAPI를 사용하고, 하나의 매개변수를 받아 동일한 크기의 값을 반환한다. 이러한 일치점들은 'LoadLibrary()' 함수를 쓰레드 함수로 사용하여 리모트 쓰레드가 생성된 후에 실행시킬 수가 있다는 힌트를 준다.

<36> hThread = ::CreateRemoteThread(hProcessForHooking, NULL, 0,

<37> pfnLoadLibrary, "C:₩₩MyHook.dll", 0, NULL);

<38> 쓰레드 함수의 매개변수로 DLL의 전체 경로명을 지정하고 LPVOID로 형변환한다. 리모트 쓰레드가 시작할 때 DLL의 이름이 쓰레드 함수(LoadLibrary)의 매개변수로 넘겨진다.

<39> 다시 부연 설명하면, 'CreateRemoteThread'라는 함수는 다른 프로세스에 쓰레드를 생성하는 함수인데, 이 함수는 쓰레드 생성시 초기화를 위한 콜백 함수인 ThreadProc의 주소를 매개변수로 받는다. 그런데 이 콜백 함수의 원형이 LoadLibrary와 매우 유사하고 ThreadProc는 생성시 단 한번 실행되므로 ThreadProc에 LoadLibrary의 주소를 넘겨 주면, 결국 목표로 하는 프로세스가 LoadLibrary를 호출하여 프로세스의 주소 공간에 DLL을 로드하게 된다는 것이다.

<40> 본 발명에서는 이와 같은 두 가지 기법을 사용하여, 특정 프로그램에 주입하고자 하는 DLL 파일 내부에 소켓 API 함수들의 추가적인 기능을 첨부하여 네트워크 패킷을 생성/차단/수정하는 네트워크 퍼징을 수행할 수 있게 된다.

<41> 도 1은 본 발명에 따른 윈도우 소켓 응용프로그램 인터페이스 후킹을 이용한 네트워크 퍼징 시스템을 나타낸 블록도이다.

<42> 도 1을 참조하면, DLL 파일 모듈(10)은, 예컨대 'CreateRemoteThread()' 함수를 이용한 DLL 주입(injection) 방법으로 내부에 별도로 정의된 소켓 함수들을 가지고 윈도우즈 운영체제에서 동작하는 네트워크 프로그램에 소켓 API 후킹 기능을 내장한 DLL 파일을 주입하여 서로 다른 네트워크 프로그램 모듈(12,20)의 소켓 API(12a,20a)들이 송수신하는 패킷들을 후킹하여 하기의 메인프로그램 모듈(14)로 보내주거나, 하기의 메인프로그램 모듈(14)이 원하는 특정 패킷을 서로 다른 네트워크 프로그램 모듈(12,20)의 소켓 API(12a,20a)들을 통해 외부로 송신하고, 하기의 메인프로그램 모듈(14)에 의해 정의된 규칙에 따라 서로 다른 네트워크 프로그램 모듈(12,20)의 소켓 API(12a,20a)들이 송수신하는 패킷들을 차단하거나 송수신되는 패킷들을 각 필드별로 분해하여 특정 필드에 임의의 데이터를 삽입, 변조하여 외부로 송신한다.

<43> 메인 프로그램 모듈(14)은 DLL 파일 모듈(10)을 경유하여 현재 윈도우즈 운영체제에서 상기 서로 다른 네트워크 프로그램 모듈(12,20)에 의해 동작 중인 네트워크 프로그램 목록을 수집하여 보여주고 DLL 파일 모듈(10)을 제어하여 사용자가 선택한 네트워크 프로그램에 DLL 파일을 주입시키게 하는 로더 역할을 수행하고, 네트워크 퍼

징을 위해 상기 DLL 파일 모듈(10)이 후킹하여 보내준 송수신 네트워크 프로그램의 패킷을 보고 수정하거나, 특정 규칙에 의해 자동적으로 패킷을 수정하거나, 별도로 지정된 규칙으로 상기 DLL 파일 모듈(10)이 패킷을 차단하거나 송수신되는 패킷들을 각 필드별로 분해하여 특정 필드에 임의의 데이터를 삽입하여 변조할 수 있게 하며, 지정된 패킷을 전송하는 기능도 구비하는 패킷 핸들러(14a)를 내장하고 있다.

<44> 데이터베이스 모듈(16)은 메인프로그램 모듈(14)이 수집하는 송수신 네트워크 프로그램의 패킷을 데이터베이스로 저장하고 패킷 분석을 수행하며, 네트워크 퍼징을 위해 패킷 내의 각 필드에 삽입 가능한 퍼징 데이터 셋을 생성하여 보관하고, 보관된 퍼징 데이터 셋은 사용자에 의해 정의된 내용이 수록되며, 사용자가 정한 규칙에 따라 메인프로그램 모듈(14)이 다루는 패킷의 각 필드에 차례대로 삽입된다.

<45> 데이터베이스 모듈(16)은 메인프로그램 모듈(14)이 자동화된 네트워크 퍼징을 수행하는 경우, 패킷의 각 필드에 삽입되는 퍼징 데이터 셋을 차례대로 메인프로그램 모듈(14) 내의 패킷 핸들러(14a)에 전달해 준다.

<46> 상기와 같이 구성되는 본 발명에 따른 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 시스템은 도 2에 나타낸 방법에 의해 다음과 같이 작동한다.

<47> 도 2를 참조하면, 윈도우즈 운영체제에서 본 발명에 따른 네트워크 퍼징 시스템에 포함된 네트워크 프로그램 모듈(12)에 의해 동작하는 네트워크 프로그램과 네트워크로 연결되어 정해진 프로토콜로 통신하는 상대편 네트워크 프로그램 모듈(20)에 의해 동작하는 네트워크 프로그램이 존재한다.

<48> 여기서, 네트워크 퍼징의 대상은 이 두 네트워크 프로그램이 사용하는 프로토콜이지만, 실제 대상은 상기한 상대편 네트워크 프로그램이다.

<49> 두 네트워크 프로그램이 P2P 또는 메신저 프로그램 등과 같이 동일한 인터페이스를 가지고 동일한 역할을 할 수도 있고, 아니면 각기 서버와 클라이언트 역할을 수행하는 별도의 인터페이스를 가진 프로그램일 수도 있다.

<50> 참고로, 본 발명에 따라 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 방법에서는 이러한 프로그램 역할에 전혀 구애받지 않으며, 단순히 네트워크 퍼징의 대상을 선정하면 된다.

<51> 상기와 같이 본 발명에 따른 네트워크 퍼징 시스템에 포함된 네트워크 프로그램 모듈(12)에 의해 동작하는 네트워크 프로그램과 네트워크로 연결되어 정해진 프로토콜로 통신하는 상대편 네트워크 프로그램 모듈(20)에 의해 동작하는 네트워크 프로그램이 존재할 때, 먼저 상기 본 발명에 따른 네트워크 프로그램 모듈은 윈도우즈 운영체제에서 동작하는 상대편 네트워크 프로그램, 즉 퍼징 대상 네트워크 프로그램이 동작한 후에 자신의 네트워크 프로그램을 실행한다(S10).

<52> 이어서, 상기 메인프로그램 모듈(14)은 상기 DLL 파일 모듈(10)을 제어하여 본 발명에 따른 네트워크 프로그램 모듈에 의해 실행된 네트워크 프로그램을 찾아서 윈도우 소켓 API 후킹 기능을 내장한 DLL 파일을 주입한다(S20).

<53> 이때부터, 상기 DLL 파일 모듈(10)은 본 발명에 따른 네트워크 프로그램 모듈(12)에 의해 실행된 네트워크 프로그램이 사용하는 소켓 API 함수, 예컨대 Winsock 1.1 함수(send, sendto, recv, recvfrom)와 Winsock 2 함수(WSASend, WSASendto, WSARecv, WSARecvFrom) 등을 후킹하여, 별도로 정의된 소켓 함수들로 대체한다(S30). 별도로 정의된 소켓 함수들은 메인프로그램 모듈의 명령에 따라 상기 본 발명에 따른 네트워크 프로그램 모듈(12)에 의해 실행된 네트워크 프로그램이 송수신하는 패킷들을 허용/수정/차단할 수 있다.

<54> 상기와 같이 소켓 API 후킹이 수행되면, 네트워크 퍼징을 수행하기 전에, 메인프로그램 모듈(14)의 명령에 따라 DLL 파일 모듈은 단순히 송수신되는 네트워크 프로그램의 패킷을 중간에 임시 저장하고, 이 패킷들을 메인프로그램 모듈(14) 내의 패킷 핸들러(14a)에 전달하며, 이에 따라서 패킷 핸들러(14a)는 이렇게 수집된 패킷을 데이터베이스 모듈(16)로 보내어 데이터베이스로 저장하고 분석되어 퍼징 데이터 셋을 생성하게 만든다(S40).

<55> 상기 데이터베이스에 저장되는 송수신 네트워크 프로그램에 대하여 사용자가 판단하는 적정한 수준의 프로토콜 패킷이 수집되어 분석되고 난 후부터, 상기 본 발명에 따른 네트워크 프로그램 모듈(12)에 의해 실행된 네트워크 프로그램이 송수신하는 패킷들과는 별도로, 메인프로그램 모듈(14)은 패킷 핸들러(14a)를 이용하여 비정상적인 패킷들을 강제로 송신하여 비정상적인 패킷을 수신하는 퍼징 대상 네트워크 프로그램에 문제점이 존재하는지를 판단하게 한다(S50).

<56> 비정상적인 패킷의 송신과정은 다음과 같이 진행된다.

<57> 데이터베이스 모듈(16)은 내부에 저장하고 있던 퍼징 데이터 셋과 분석된 패킷의 데이터 영역에 관한 정보를 메

인프로그램 모듈(14) 내의 패킷 핸들러(14a)에 전달하여 자동화된 네트워크 퍼징을 수행하며, 패킷 핸들러(14a)는 퍼징 데이터 셋을 분석된 패킷의 데이터 영역에 적절하게 삽입한다. 그러면 메인프로그램 모듈(14)은 그 패킷을 DLL 파일 모듈(10)이 가져가 정상적인 소켓 API(12a,20a)를 통하여 외부로 송신하게 만든다. 이때, 상기 데이터베이스 모듈(16)은 패킷의 각 필드에 삽입되는 퍼징 데이터 셋을 차례대로 메인프로그램 모듈(14) 내의 패킷 핸들러(14a)에 전달한다.

<58> 상기 상대편, 즉 퍼징 대상 네트워크 프로그램이 강제 송신된 패킷에 대해서 응답을 하게 되면, 그에 대한 통제도 역시 DLL 파일 모듈(10)을 제어하는 메인프로그램 모듈(14)에 의해서 가능하다. 메인프로그램 모듈(14) 내의 패킷 핸들러(14a)는 모든 송수신되는 패킷들을 허용/수정/차단할 수 있다.

<59> 이상에서 설명한 본 발명에 따른 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 시스템 및 그 방법은 상기한 실시예에 따른 서비스에 한정되지 않고, 이하의 특허청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 본 발명이 속하는 분야에서 통상의 지식을 가진 자라면 누구든지 다양하게 변경하여 실시할 수 있는 범위까지 본 발명의 기술적 정신이 있다고 할 것이다.

발명의 효과

<60> 상술한 바와 같은 본 발명에 따른 윈도우 소켓 API 후킹을 이용한 네트워크 퍼징 시스템 및 그 방법에 의하면, 종래의 프로토콜 분석과 퍼저 제작을 거치지 않고, 임의의 프로토콜을 사용하는 네트워크 프로그램을 이용하여, DLL 주입을 통한 소켓 API 후킹으로 네트워크 퍼징을 자유롭게 할 수 있다.

<61> 또한, 네트워크 퍼징을 수행하기 위해 프로토콜 분석부터 퍼저의 제작까지 많은 인력과 시간이 소모되는 종래의 문제점을 완전 해소할 수 있다.

도면의 간단한 설명

<1> 도 1은 본 발명에 따른 윈도우 소켓 응용프로그램 인터페이스 후킹을 이용한 네트워크 퍼징 시스템을 나타낸 블록도.

<2> 도 2는 본 발명에 따른 윈도우 소켓 응용프로그램 인터페이스 후킹을 이용한 네트워크 퍼징 방법을 나타낸 플로차트.

<3> <도면의 주요 부분에 대한 부호의 설명>

<4> 10: DLL 파일 모듈 12: 네트워크 프로그램 모듈

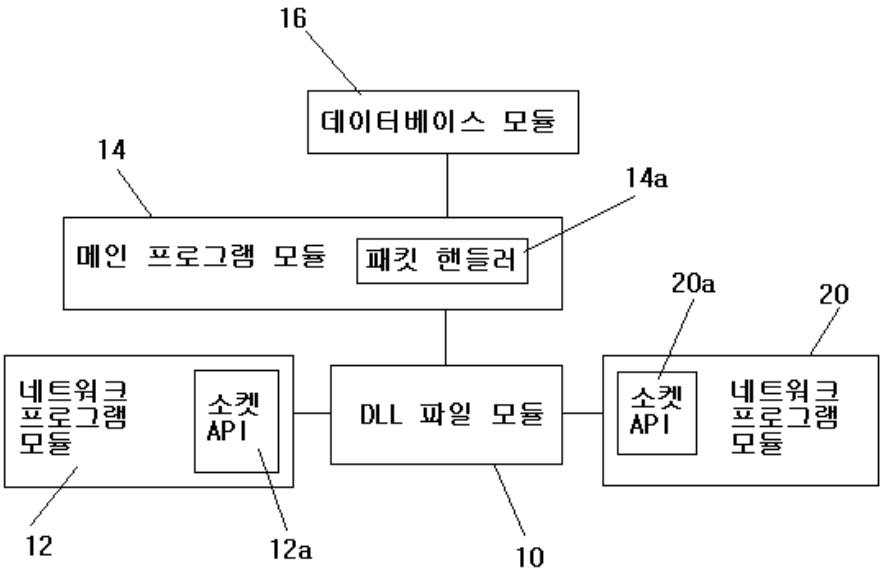
<5> 12a: 소켓 API 14: 메인 프로그램 모듈

<6> 14a: 패킷 핸들러 16: 데이터베이스 모듈

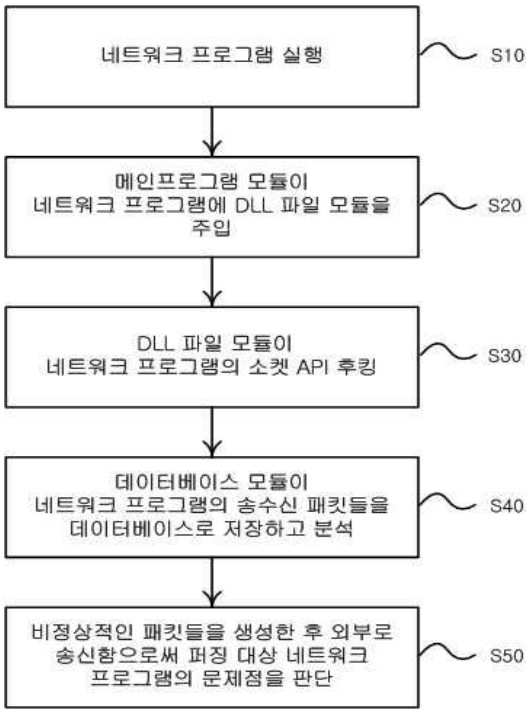
<7> 20: 네트워크 프로그램 모듈 20a: 소켓 API

도면

도면1


16
데이터베이스 모듈
14
14a
메인 프로그램 모듈
패킷 핸들러
20a
20
네트워크 프로그램 모듈
소켓 API
DLL 파일 모듈
소켓 API
네트워크 프로그램 모듈
12
12a
10

도면2


네트워크 프로그램 실행
S10
메인프로그램 모듈이 네트워크 프로그램에 DLL 파일 모듈을 주입
S20
DLL 파일 모듈이 네트워크 프로그램의 소켓 API 후킹
S30
데이터베이스 모듈이 네트워크 프로그램의 송수신 패킷들을 데이터베이스로 저장하고 분석
S40
비정상적인 패킷들을 생성한 후 외부로 송신함으로써 퍼징 대상 네트워크 프로그램의 문제점을 판단
S50