idefense

Win32 메시지 취약점 다시 보기

여전히 위협적인 파편 공격

번역 및 내용보충 : 와우해커 (http://www.wowhacker.org)

목차

초록

약 1년전, Chris Paget, aka Foon은 윈도우 이벤트 모델의 구조 상에서 근본적인 문제점에 대한 두개의 논문을 발간하였다. Paget은 이런 문제점들로 인해 그거 "파괴 공격"(Shatter Attack)이라고 이름붙인 공격이 가능해지게 되었고, 이 공격은 광범위하며 치료할 수 없다고 주장하였다. 확고한 주장과 Paget의 노골적인 반-마이크로소프트 어조는 파편 익스플로잇(Shatter Exploit)이 미디어의 전면에 알려지도록 만들었고, 보안 커뮤니티안에서 이 주장의 정확성과 중요성에 대한 상당한 논의가 이뤄졌다. 이런 분위기에 제압당한 마이크로소프트는 더이상 파편 공격을 불가능하게 만들 것이라고 믿게 만든 보안 고시 MS02-071(security bulletin MS02-071)과 패치를 펴내게 되었다.

한 전문가가 지적했듯이, 정확한 사실 대신, 운영체제에 대한 열광적인 지지 또는 상업적인 관심이 판을 치는 현실에서는 파괴 공격에 대한 잘못된 정보들이 사람들을 오도할 수 있다. 이 문서에서는 윈도우 이벤트 모델의 문제점이 정확히 무엇이며 일반적인 소프트웨어에서 일어날 수 있는 이에 관련된 취약점을 설명하고 이런 "치료 불가능한" 문제를 해결하는 방법을 제안하겠다.

시작하며

윈도우 GUI와 같은 이벤트 구동 시스템(event-driven system)은 이벤트라고 불리는 정보의 단위를 통해 사용자와 상호작용을 하게 된다. 마우스 클릭, 키입력 등의 시스템과 사용자간의 통신의 경우 이벤트을 통해 물리적인 하드웨어로부터 소프트웨어로 전달되게 된다. 하지만 꼭 하드웨어의 요청이 없더라도, 윈도우의 내용물을 그린다던지 종료 요청 등에도 이벤트를 사용한 모델이 유용하게 쓰일 수 있다.

윈도우 용어로 이벤트는 "윈도우 메시지"다. 이 윈도우 메시지는 운영체제나 다른 윈도우가 한 윈도우가 반응을 해야할 어떤 일이 생겼다는 것을 알려주기 위한 정보의 단위다. 윈도우 기반의 컴퓨터에 나타난 각 윈도우들은 윈도우에 보내진 모든 메시지를 받는 윈도우 프로시져를 가지고 있어 각 메시지에 대해 적절히 대응할 수 있다. 윈도우 프로시져의 대부분은 윈도우즈 운영체제 자체에 미리 정의되어 있다. DLL속에 있는 이런 프로시져들은 특정 형태의 모든 윈도우에 적용되곤 한다. 예를 들어 윈도우즈 상의 모든 스크롤바는 스크롤바의 위치를 바꾸기 위한 키보드나 마우스 움직임에 대한 반응을 위해 공통적인 윈도우 프로시져를 공유하고 있다.

윈도우 메시지는 전부가 현실 세계에서 만들어진 것이 아니라는 점에 주의해야 한다. 윈도우 메시지는 프로그램과 윈도우즈 운영 체제 사이의 상호작용을 담당하고 있다. 예를 들어 대화 상자의 컨트롤의 값을 변화시키거나 재시도하는 것과 같은, 이벤트라기 보다는 전통적인 함수 호출과 유사한 작업을 할 때에도 프로그램에서는 윈도우 메시지를 이용한다.

불행히도 윈도우 메시지 시스템은 개인용 컴퓨터의 보안이 중요하게 다뤄지지 않던 시절에 만들어졌다. 대화 상자 컨트롤의 값을 변화시키는 것은 일반적으로 다이얼로그 박스를 가지고 있는 프로그램이나 운영체제에서만 접근할 수 있어야 한다. 하지만 한 윈도우에서 다른 아무 윈도우로 아무 메시지나 보낼 수 있다는 점은 절대적으로 좋지 않은 결과를 가져올 수 있다. 현실적인 입장에서는 사소한 것일지 모르지만, 이런 점은 명확하게 윈도우즈의 디자인 오류다.

윈도우를 생성하는 사용자보다 더 높은 권한에서 실행되는 프로그램의 경우 이 문제가 심각해질 수 있다. 최근의 윈도우즈 버젼에서는 사용자는 제한된 권한을 갖는다. 같은 컴퓨터의 다른 사용자에 상당히 간섭할 수 있는 기능은 멀티 유저 운영체제의 분명한 특징이지만, 윈도우에는 이런 것이 불가능하다. 그러나 몇몇 프로그램은 일반적인 사용자 프로그램이 할 수 없는 작업들을 팔 필요가 있다. 바이러스 스캐너는 워드 프로세서에서는 하지 않는 작업들을 한다. 이런 프로그램들은 보통 그 영향에 관계 없이 아무 일이나 할 수 있는 시스템 서비스로 구현되어 있따.

파편 공격(shatter attack)을 다룬 Chris Paget(aka Foon)의 문서에서, 작은 프로그램을 돌리는 사용자가 전체 시스템의 모든 권한을 얻을 수 있도록 할 때 이런 윈도우 메시지의 특성을 사용하였다. 높은 권한의 프로그램을 생성하는 위조된 WM_TIMER 윈도우 메시지를 윈도우즈에 보냄으로써 공격자는 자신의 권한에 관계없이 원하는 프로그램을 모두 사용할 수 있게 되었다. Paget은 다양한 메시지를 이용해 이런 작업을 할 수 있다고 주장했다. 그러나 다른 예들은 단지 추측에 근거한 것이고 실제로는 그가 제안한대로 동작하지는 않는다. 마이크로소프트에서 처음에는 이 문제가 실제로 문제가 되지는 않는다고 하였지만 그 후 WM_TIMER 윈도우 메시지의 행동 양식을 바꾸는 패치를 릴리즈했다. Paget의 다른 주장이 부정확했기 때문에 이 문제는 표면적으로는 해결된 것 처럼 보였다.

WM_TIMER를 사용하지 않는 파편 공격

버그가 있는 프로그램을 이용해 높은 권한을 획득하기 위해서 공격자는 다음 두가지를 해야 한다.

먼저 새로운 CPU 명령어를 높은 권한을 가진 프로그램의 어드레스 공간(address space) 사이에 집어 넣어야 한다. 사용자 명령은 - 제대로 다뤄질 경우 -일련의 CPU 명령어처럼 해석되기 때문에, 어드레스 공간에 집어넣는 일은 일반적으로 매우 쉽다. 윈도우 메시지의 경우 더 간단하다. 윈도우 메시지는 사용자 입력과 프로그램간에 사용하는 윈도우즈의 메카니즘이고, 따라서 priori 메카니즘을 이용하면 새로운 명령어를 집어넣을 수 있다.

두번째로, 제공된 명령을 프로그램이 실행시킬 수 있도록 만들어야 한다. 소프트웨어 상의 다양한 보안 문제점은 공격자가 프로그램의 실행을 임의의 메모리 위치로 바꿀 수 있게 해주는 디자인이나 프로그래밍 오류로 인해 나타난다.

Paget의 파편 공격은 매우 간단하다. 운영 체제의 타이머가 만료되었을 때 윈도우에 전송하는 WM_TIMER 윈도우 메시지는 메시지에 대해 반응할 함수의 위치만을 가지고 있다. 이

주소를 공격자가 준비해놓은 명령어들이 있는 위치로 바꿔준다면, 원하는 모든 것을 할 수 있게 된다. 윈도우에서 메시지의 발신지에 대해 아무런 정보도 가지고 있지 않기 때문에, 운영체제에서 보내진 것이나 적대적인 프로그램에서 보내진 것을 구별할 방법이 없다.

WM_TIMER자체의 성질로 인해 원래의 "파편 공격"은 매우 고치기 쉬운 문제다. WM_TIMER안에 전송되는 주소는 SetTimer() API 함수를 이용해 운영체제에 등록된 것이라고 생각할 수 있다. WM_TIMER에서 받은 주소와 등록된 주소를 비교함으로써 Microsoft에서는 이런 공격을 방지할 수 있었다. 이런 점은 한가지 의문을 낳는다. 다른 윈도우 메시지드도 WM_TIEMR처럼 실행의 방향을 바꿀 수 있을 것인가?

일반적으로 메시지에서는 간단한 데이터만을 다루고, 아무리 이런 데이터가 문자열이나 구조체에 대한 주소라고 해도, 윈도우즈는 그 주소에 있는 데이터로 이동하는 것 같다. 예를 들어 윈도우의 제목을 바꿀 때 사용되는 WM_SETTEXT메시지의 경우, 악의적으로 사용되기 어렵다. WM_SETTEXT 메시지에 잘못된 주소를 보내도, 전송하는 프로그램은 페이지 오류를 내며 종료하게 된다; 윈도우즈는 그 메시지가 전달되기 이전에 그 메시지를 읽으려고 시도한다. 의심할 여지 없이, 윈도우즈는 메시지가 한 프로그램에서 다른 프로그램으로 전달될 때 메시지의 주소를 다르게 취급한다. 따라서 위험해보이는 윈도우 메시지가 사실은 상당히 안전하게 유지되고 있다.

WM_TIMER는 이 메시지가 데이터 조각의 주소가 함수의 주소를 담고 있다는 점에서 대부분의 다른 윈도우 메시지들과는 다르다. 하나의 프로그램에서 허용된 함수 주소는 일반적으로 다른 함수에서도 허용된다. 한 프로그램에서 다른 프로그램으로 함수를 옮기는 것은 명백히 불가능하기 때문에, 윈도우가 이 메시지에 할 수 있는 일은 아무것도 없다.한 프로그램이 다른 프로그램에 의해 만들어진 함수의 주소를 가진 메시지를 보낸다면, 윈도우즈는 그 주소가 수신자에게 맞지 않는 주소라도 메시지를 전송한다. WM_TIMER 메시지로 인해 생기는 취약점은 메시지가 메모리 주소를 가지고 있다는 사실 자체로 인한것이 아니라, 그 주소가 함수를 가리키고 있는 것 때문에 일어난다.

윈도우즈 SDK 문서를 한번 빨리 찾아보면, 다른 많은 메시지들도 함수의 주소를 가지고 있는 것을 알 수 있다. 한 예로, 문서에서 단어들의 한계를 판단하기 위해 에디트 컨트롤이 호출하는 함수의 주소를 지정하기 위해 사용되는 EM_SETWORDBREAKPROC 또한 함수의 주소를 담고 있다. EM_SETWORDBREAKPROC는 에디트 컨트롤에서만 사용할 수 있지만, 한 윈도우를 생성한 프로그램이 에디트 컨트롤과 같은 기본적인 컨트롤을 가지고 있을 수 있다.

익스플로잇
아래 소스코드 부분의 목록 1에는 마이크로소프트의 WM_TIMER 패치가 적용된 시스템에서도 여전히 파편 공격이 동작할 수 있다는 것을 보여주는 소스가 있다. 이 유틸리티는 에디트 컨트롤을 만든 프로그램의에서 임의의 명령어로 실행 방향이 전환되도록 만들어주고 다음 프로그램을 실행하는 시스템에서 높은 권한을 가질 수 있다는 것을 보여준다.

- Kerio Personal Firewall 2.1.4
- Sygate Personal Firewall Pro 5.0
- McAfee Virus Scan 7.0
- WinVNC 3.3.6 및 그 파생 프로그램

이 프로그램들은 마이크로소프트의 보안 예제와는 다르게 인터렉티브한 데스크탑에서 윈도우를 생성하는 권한을 가진 서비스로써 동작한다. 이 윈도우들의 대부분은 EM_SETWORDBREAKPROC 윈도우 메시지에 의해 공격받을 수 있는 에디트 컨트롤을 가지고 있기때문에 이 프로그램을 실행할 수 있는 어느 시스템에서나 공격자가 거의 제한되지 않는 LocalSystem의 권한을 얻을 수 있다.

위에 나열된 네개의 프로그램이 이를 종류의 공격에 취약한 유일한 프로그램은 절대 아니다. 또 EM_SETWORDBREAKPROC 윈도우 메시지만이 이런 공격을 할 수 있는 것도 아니다. 이런 종류의 공격이 윈도우 자체의 결함에 의해 생긴 것이기 때문에, 일반적이지 않은 사용자 인터페이스를 사용하는 높은 권한을 가진 프로그램들은 이런 공격에 대해 고스란히 노출되어 있다. Paget의 원래 글이 거의 1년전에 나왔음에도 불구하고, 마이크로소프트의 반론으로 인해 많은 윈도우즈 프로그램들은 여전히 안전하지 않은 방식으로 동작하고 있다.

취약한 프로그램 찾기

단지 개념 정리를 위한 프로그램이지만, 목록 1의 익스플로잇 코드는 EM_SETWORDBREAKPROC 윈도우 메시지에 기반한 공격에 대해 취약한 프로그램을 충분히 찾아낼 수 있다. 프로그램을 테스트 하기 위해서는 다음과 같은 과정을 거친다.

1. 프로그램을 높은 권한으로 실행시키고 유저 인터페이스를 통해 보여준다. 이런 것은 윈도우즈의 taskmgr.exe(장치관리자)를 통해 쉽게 할 수 있다.

2. 목표로 하는 프로그램의 에디트 컨트롤을 가진 윈도우를 찾는다. 이 것은 직접 프로그램을 살펴보거나 SPY++라는 마이크로소프트 비주얼 스튜디오에 있는 프로그램을 통해 알

아볼 수 있다. 한 예로 WinVNC의 설정 창을 사용한도록 하겠다.(이 것은 WinVNC의 트레이 아이콘을 오른쪽 클릭하고 properties를 누르면 된다.)

3. 익스플로잇 유틸리티를 실행시킨다. 유틸리티는 에디트 콘트롤을 가진 윈도우의 제목을 표시한다; 여기서는"WinVNC: Current User Properties"이다.(윈도우를 볼 수 있어야 한다.) 그 후 유틸리티는 쉘코드 어드레스를 표시해준다.

4. 프로그램의 메모리 이미지 중에서 쉘코드가 어디에 있는지 판단한다. 가장 선호하는 디버거를 실행시키고 프로세스에 그것을 부착시킨다. (SeDebugPriviledge를 가지고 있어야 한다.) 익스플로잇에서 사용되는 처음 쉘코드의 4바이트는 ASCII 문자열 "xeno"이다. WinDbg 디버거를 사용하면 프로세스에서 이 문자열을 다음과 같이 찾을 수 있다.

이 경우 쉘코드의 주소는 0x0015d250이다. 쉘코드는 NOP 명령으로 길게 나열되어 있기 때문에 실제 주소에서 변형을 설명하기 위해 0x00160000으로 반올림될 수 있다.

5. 성공? 익스플로잇 유틸리티에 쉘코드를 넣고 엔터를 누른다. 주소가 옳을 경우 타겟 프로그램와 같은 사용자의 명령 프롬프트라 나오는 것을 볼 수 있어야 한다. 주소가 옳진 않지만 취약한 프로그램의 경우 페이지 오류를 내며 멈추게 된다. 다시 시도해 보라.

익스플로잇이 어떻게 동작하는가?

목록 1의 익스플로잇 코드는 매우 간단하다. 이 프로그램의 의도가 자동으로 높은 권한을 획득하기 위한 것이 아니라 단지 취약한 프로그램이가 테스트 해보기 위한 것이므로 C코드로는 세줄로도 충분하다.

sendMessage(hWndChild, WM_SETTEXT, 0, (LPARAM)sc);

먼저, 익스플로잇에서 LPARAM에 쉘코드의 주소를 가진 WM_SETTEXT 윈도우 메시지를 목표로 하는 에디트 컨트롤에 전송한다.

SendMessage(hWndChild, EM_SETWORDBREAKPROC, 0L, (LPARAM) lExecAddress);

다음 LPARAR을 목표로 하는 프로그램에서 쉘코드의 주소로 하여 EM_SETWORDBREAKPROC 메시지를 에디트 컨트롤에 보낸다.

SendMessage(hWndChild, WM_LBUTTONDBLCLK, MK_LBUTTON, (LPARAM)

0x000a000a);

마지막으로 WM_LBUTTONDBLCLICK 메시지를 에디트 컨트롤에 전송한다. 왼쪽 마우스로 더블 클릭을 하면 마우스 포인터 아래에 있는 단어가 선택되기 때문에 메시지 컨트롤에 대한 응답으로 대응되는 단어의 경계를 찾기 위해 EM_SETWORDBREAKPROC 메시지가 사용되어 우리가 넣은 주소의 0x000a000a를 호출하게 될 것이다. 이 시점에서 타겟 프로그램에서 실행하는 명령어가 무엇이든지 간에 실행 방향은 우리의 쉘코드로 건너 뛰게 된다.

이 익스플로잇이 완전 자동화되어있지 않다는 것이 이런 종류의 공격이 자동화하는데 상당한 어려움이 있다는 것을 의미하진 않는다. 그러나 WM_SETTEXT에 제공된 버퍼는 극단적으로 클 수 있고, 프로그램은 보통 여러 윈도우로 구성되어 있고, 이 윈도우들은 각기 모두 악의적인 명령을 실행시키기 위해 쓰일 수 있다. 목록 1에서의 쉘코드에 NOP 문자열이 확장되고 여러개의 윈도우들이 WM_SETTEXT를 이용해 쉘코드를 전송한다면 적절한 주소를 선택하는 것은 매우 쉬워진다는 것은 명백하다. 바이러스에 의한 이런 취약점으로 인해 자동화된 익스플로잇으로 바이러스 트로이안 목마나 다른 악의적인 프로그램들을 만들 수도 있다.

메시지 기반 공격 예방

악의적인 윈도우 메시지를 통하여 높은 권한을 얻는 방식의 공격을 막는 가장 효과적인 방법은 간단하다 - 유저 인터페이스를 표시하는 프로그램이 높은 권한을 갖지 않도록 하는 것이다. 이 방법은 마이크로소프트사가 추천하는 것으로 매우 간단하고도 효과적인 방법이다. 높은 권한을 필요로 하는 프로그램들은 프로세스간의 통신(IPC)을 통해 유저 인터페이스를 분리하도록 다시 만들 수 있다. 낮은 권한의 유저 인터페이스는 윈도우 메시지가 권한을 높이기 위해 사용될 우려를 방지할 수 있다. 윈도우즈 레지스트리에서 인터랙티브 서비스를 중지하거나 서비스 제어판(services control-panel appplet)을 통한다면 시스템을 이런 종류의 공격으로부터 방어할 수 있다.

현재 메시지 기반 공격에 대해 취약한 프로그램들이 상당히 많이 존재한다. 그러나 유저 인터페이스를 낮은 권한으로 분리하여 동작하게 하는 것은 상당한 양의 수고가 필요하다. 결과적으로 높은 권한을 가진 프로그램이 안전하게 유저 인터페이스를 표시하는 접근 방식이 가능한지 살펴보는 것도 흥미로울 것이다.

윈도우 메시지에 기반한 공격을 막는 일은 그 공격 방법만큼이나 간단하다. 이런 류의 공격을 막는 가장 효과적인 방법은 윈도우 메시지에 소스 정보를 포함하는 것이다. 불행히도 윈도우의 소스 코드에 접근하지 않는다면 이런 방법은 매우 어려운 일이다. 가장 간단한 대안

은 잠재적으로 위험한 윈도우 메시지를 차단하는 것이다. 그러나 많은 사람들의 생각과는 다르게 이런 방법은 쉽게 만들 수 있는 것이 아니다.

윈도우 메시지에는 큐를 사용하는(queued) 메시지와 큐를 사용하지 않는 (non-queued)메시지의 두가지 종류가 있다. 전자의 경우 프로그램의 메시지 큐에 저장이 되는 메시지들로 구성되어 있다. 일반적인 프로그램들의 경우 GetMessage()와 PeekMessage()라는 Win32 API 함수를 이용해 메시지를 큐로부터 꺼내서 DispatchMessage()함수를 이용해 적절한 윈도우 프로시져(window procedure)에 적용시키는 이벤트 루프(event loop)를 포함하고 있다. 각자가 큐에서 꺼내진 후, 일정한 검사를 한 후 어떤 기준에 의해 허가되거나 거절되게 된다.

익스플로잇에 사용되는 EM_SETWORDBREAKPROC과 같은 큐에 저장되지 않는 메시지의 경우, 메시지 큐에 들어가지 않기 때문에 적절한 메시지를 걸러내기가 더욱 어렵다. 큐에 저장되지 않는 메시지가 SendMessage()를 이용해 윈도우에 전송되면 윈도우의 윈도우 프로시저가 그 메시지를 받는 즉시 바로 실행된다. 프로그램이 이벤트 루프에서 이런 종류의 메시지를 처리하지 않기 때문에, 윈도우 프로시저가 불려지기 전에 메시지를 검사할 방법이 전혀 없다. 따라서 잠재적으로 악의적인 큐에 저장되지 않는 메시지의 경우 거절할 수 있는 확실한 방법으 존재하지 않는다. WH_CALLWNDPROC 인자와 함께 SetWindowsHookEx() API 함수를 사용하는 방법을 이용하면 큐에 저장되지 않는 메시지가 윈도우 프로시저에 도달하기 전에 검사할 수 있지만, 메시지를 거절할 수는 없다. 게다가 WH_CALLWNDPROC 윈도우들이 메시지를 전송하는 쓰레드의 실행을 정지시키기 때문에 프로그램간에는 어떻게 동작하는지 확실하지 않다.

Windows API를 이용할 경우 큐에 저장되지 않는 메시지를 걸러내는 직접적인 방법이 없기 때문에, 조금 기발한 접근 방식이 필요하다. 한가지 방법으로, 프로그램이 윈도우의 윈도우 프로시저를 오버라이드할 수 있게 해주는 서브 클래스라는 윈도우 개체를 사용할 수 있다. 모든 프로그램의 에디트 컨트롤의 윈도우 프로시저를 오버라이드 함으로써, 메시지가 큐에 들어가던지 들어가지 않던지 모든 메시지들을 받자마자 시험해볼 수 있다. 그 후 메시지가 안전하다고 판단되었다면 원래의 에디트 컨트롤 윈도우 프로시져로 보내지고, 안전하지 않다면 거절된다.

목록 2에는 큐에 들어가지 않는 윈도우 메시지를 필터링하기 위해 사용할 수 있는 DLL 소스코드가 있다. DLL이 실험적이기는 하지만, 소스코드를 바꾸지 않고도 특정한 윈도우 메시지를 완전히 차단하도록 쓰일 수 있다.
위에서 언급된 4개의 프로그램들이 이 DLL을 로드하면 메시지 기반 공격을 막아낼 수 있다.

잠재적으로 악의적인 윈도우 메시지를 필터링하는 것이 프로그램의 보안을 위해 충분하지 않다고 하더라도, 이 방법을 보안을 높이기 위한 프로그래밍 방법들과 조합한다면 높은 권한의 프로세스에서도 유저 인터페이스를 상당한 정도의 안전성을 가지고 실행할 수 있을 것이다.

그럼에도 불구하고, 이 문제를 완벽하게 막을 수 있는 이상적인 해법은 존재하지 않는다. 메시지에 소스 정보를 넣지 않는다면, 명백히 위험해 보이는 메시지들을 걸러낼 수 있을 뿐이다. 표면상 안전해 보이는 메시지를 다루는 방법에서 생길 수 있는 오류의 위험이 여전히 남아있다.

결론

규모가 큰 소프트웨어 프로젝트에 참가한 사람들은 한결같이 소프트웨어 개발 시 프로그램의 결함은 피할 수 없다고들 한다. 윈도우 운영 체제가 완벽과는 거리가 멀다고는 해도, 현재 사용되는 몇가지 점들이 오래 전에 디자인 되었다는 사실만으로도 충분히 비난받을 만큼 비합리적인 모습을 보여주고 있다. 물론 비난할만한 이유는 훨씬 더 많다.

권한이 낮은 사용자가 높은 권한의 프로그램을 사용할 때마다 잠재적인 보안상의 문제점이 존재한다. setuid와 같은 것들이 수년간 보안상의 구멍을 만들어왔던 UNIX와 같은 플랫폼에서, 이런 점들은 이미 진부한 이야기가 되었다. 이런 문제점을 다루는 것은 안전한 코드를 만들기 위한 기본이고, 사용자의 원래 권한보다 높은 권한을 사용할 때마다 고려되어야 하는 부분이다.

메시지가 설계된 방식으로 인해 생긴 에러들이 프로그래머에게는 불편하기는 하지만, 이런 문제들이 절대 해결할 수 없는 것은 아니다. 여러 프로그램들이 메시지 기반 공격에 대해 취약한 문제들은 윈도우즈 운영체제로 인해 생기는 것이 아니라, 그 윈도우즈 위에서 실행되는 소프트웨어를 쓴 프로그래머로 인해 생기는 것이다.

감사의 말

Geoff Shively(PivX), Drew Copley(eEve)와 Adam Shostack(Informed Security)의 조언과 격려에 대해 감사를 표한다. 또한 언제나 Karen, Tessa, Dan에게 언제나 모든 것에 대해 감사하고 있다.

다음 각 사람 모두의 노력에 감사를 표한다.

iDEFENSE사, 취약점 기고 프로그램, 부장, Sunil James
iDEFENSE사, 기술 보도, 이사, David Endler
iDEFENSE사, 부장 편집인, Andrew Schmidt
iDEFENSE사, 편집인, Catherine Beck

소스 코드

아래에 있는 소스코드는 다음 사이트에서 받을 수 있다.
http://www.idefense.com/idpapers/shatter_paper_source.zip

목록 1: Shards.cpp
목록 2: Pyrex.cpp