AxMan 소개 및 사용법

By poc@securityproof.net

http://metasploit.com/users/hdm/tools/axman/ 에 있는 소개 글의 필수적인 부분에 대한 편역과 세부 설명을 덧붙였습니다. 잘못된 부분이 있으면 지적 부탁드립니다. 그리고 부족한 부분이 있다면 스스로 채워보시기 바랍니다.

[소개]

AxMan은 H.D.Moore에 의해 개발된 웹 기반의 ActiveX fuzzing 엔진이다. AxMan의 목적은 Internet Explorer를 통해 노출된 COM 오브젝트에 존재하는 취약점을 발견하는 것이다. AxMan은 Internet Explorer 6에만 사용되도록 디자인 되었다.

[사용 방법]

먼저 AxMan은 등록된 COM object들과 관련 typelib 정보를 열거하는 것으로 시작한다.

```
C:\>axman.exe result                    // 관리자 권한으로 실행
```

Result 파일이 만들어지는 이 과정에서 새로운 프로세스들이 생성되고, 팝업 창이 뜰 수 있다. 이때 팝업 창을 닫거나 전체 과정이 다 끝날 때까지 기다려도 된다.

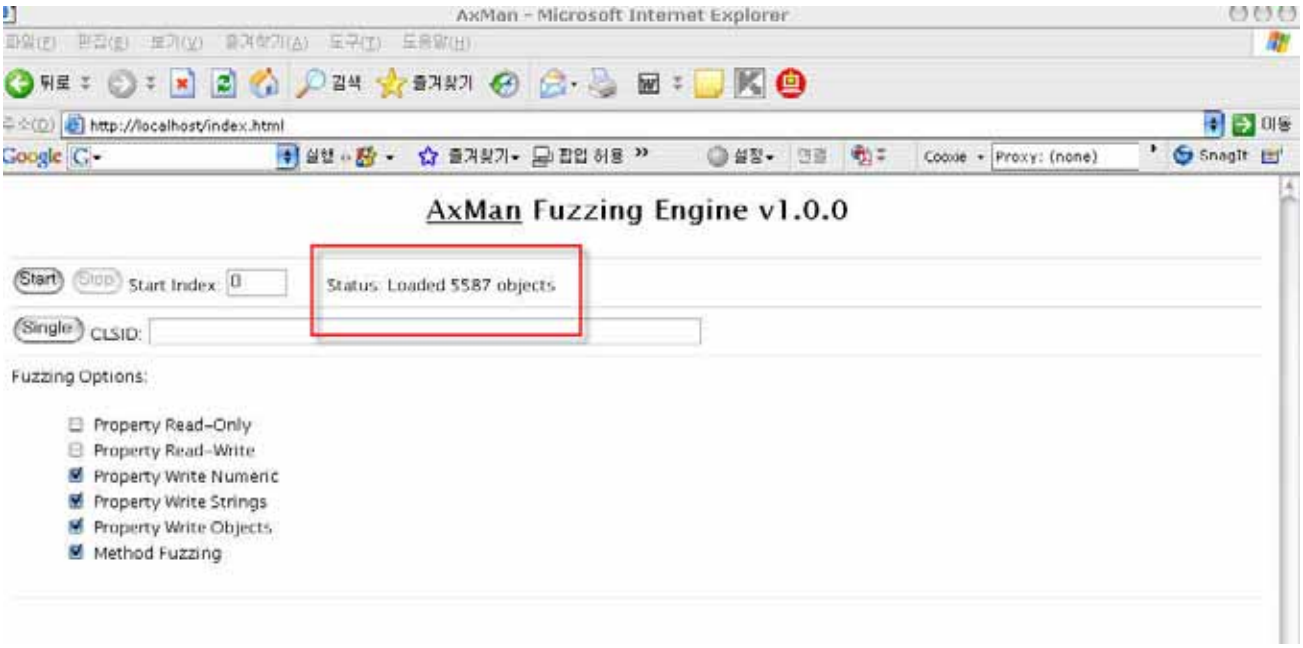
이 과정이 끝나면 result란 디렉토리가 생성되며, 그 안에는 모든 등록된 COM object에 대한 자바스크립트 소스 파일을 가진 objects란 파일이 생성된다.

여기서 생성된 COM 프로세스들을 죽이고, 시스템의 자원을 원 상태로 돌리기 위해 시스템을 다시 부팅해야 한다.

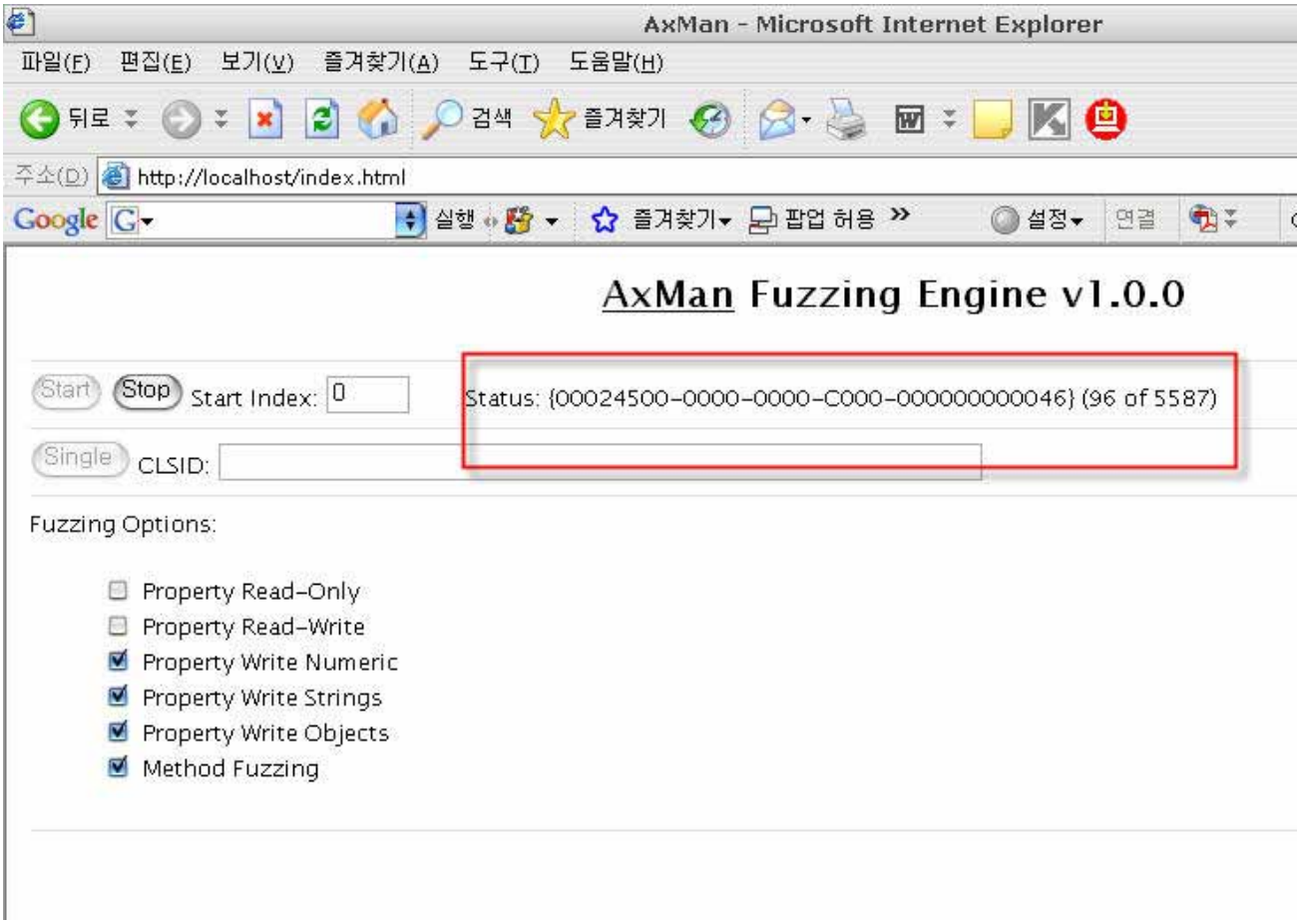
다시 부팅 한 후 AxMan user 인터페이스를 호스팅할 웹 서버를 준비한다.
로컬 시스템에서 웹 서버를 돌리는 것이 가장 효율적이지만, HTM과 자바스크립트 파일을 사용할 수 있는 웹 서버라면 어떤 것이든 상관없다.

AxMan을 다운받아 압축을 풀면 bin, html, source 이렇게 3개의 디렉토리가 있다. 여기서 html 디렉토리의 파일들 전체를 web root로 복사하고, result 디렉토리에 있는 objects 파일을 'conf'라는 디렉토리에 복사한다. conf라는 디렉토리는 없으므로 사용자가 직접 만들어야 한다.

로컬 시스템에서 웹 서버를 실행하기 위해서 필자는 이근상님의 APM_SETUP을 사용했다.
모든 준비가 끝나면 웹 서버를 실행한 후 Internet Explorer를 실행한 후 index.html 파일을 브라우징한다. 만약 모든 것이 제대로 설치되고 준비되었다면 얼마나 많은 COM object 정의들이 로딩되었는지를 나타내는 상태 메시지가 아래와 같이 보인다.



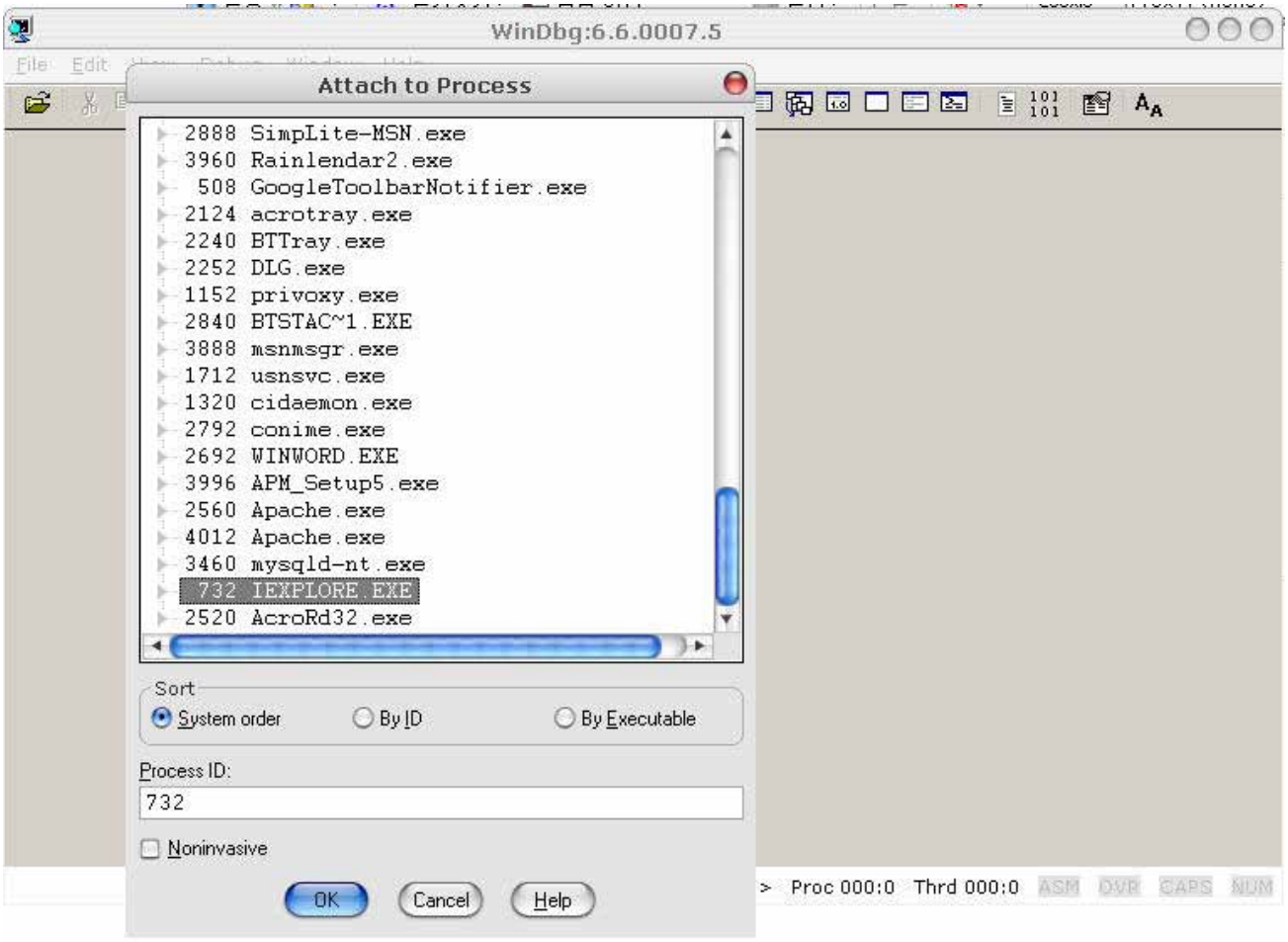
이제 fuzzing 과정을 시작하기 위해 Start 버튼을 클릭한다. 그러면 fuzzing 상태가 아래와 같이 보인다.



이 과정에서 많은 시간이 걸린다.

[디버깅]

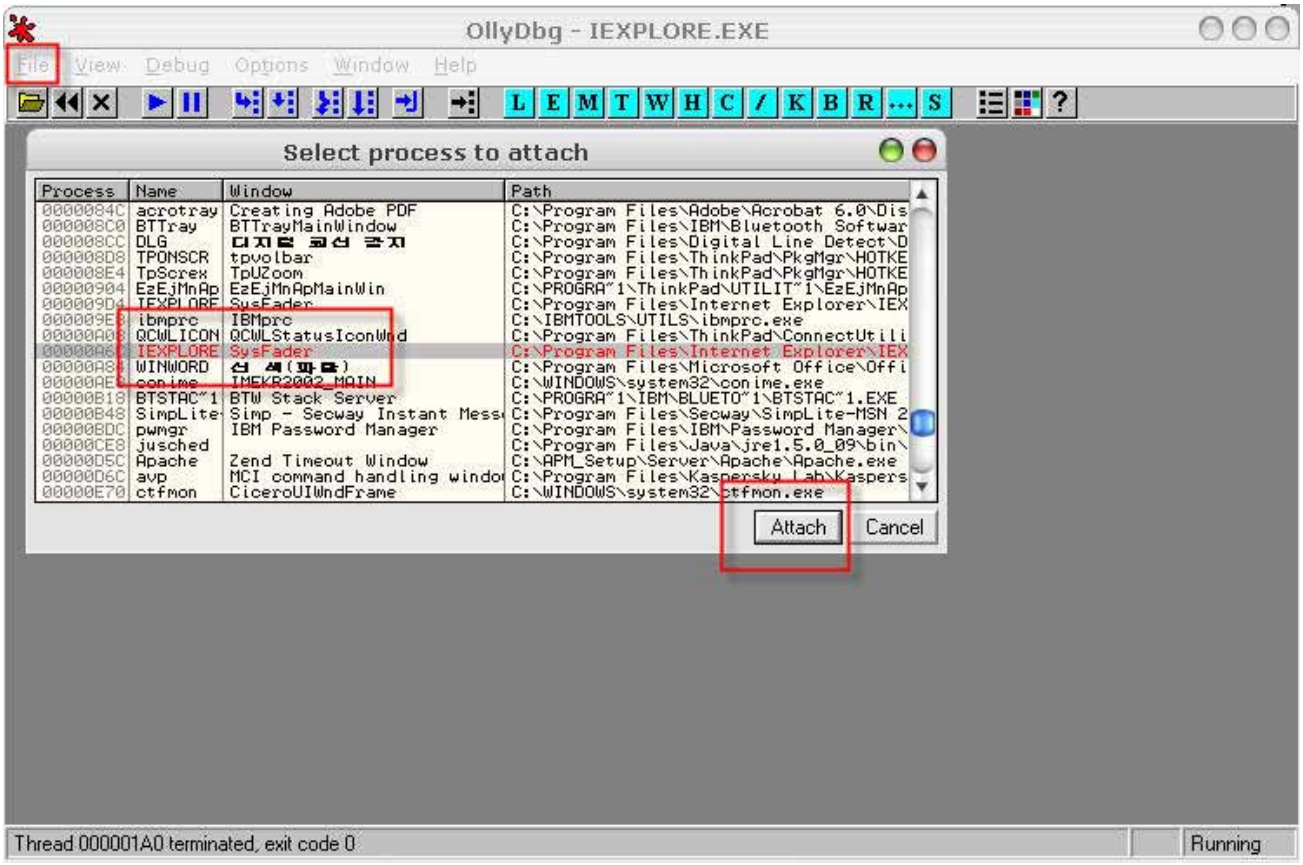
Axman은 전형적인 시스템에 많은 crash를 초래할 것이다. 무엇이 crash되었는지, 그리고 어떤 테스트 동안인지를 결정하려고 노력하는 것은 브라우저 기반의 fuzzing 엔진에게는 도전이 될 것이다. 이런 문제를 해결하기 위해 WinDbg를 설치하고, fuzzing 프로세스를 시작하기 전에 iexplore.exe에 attach한다.



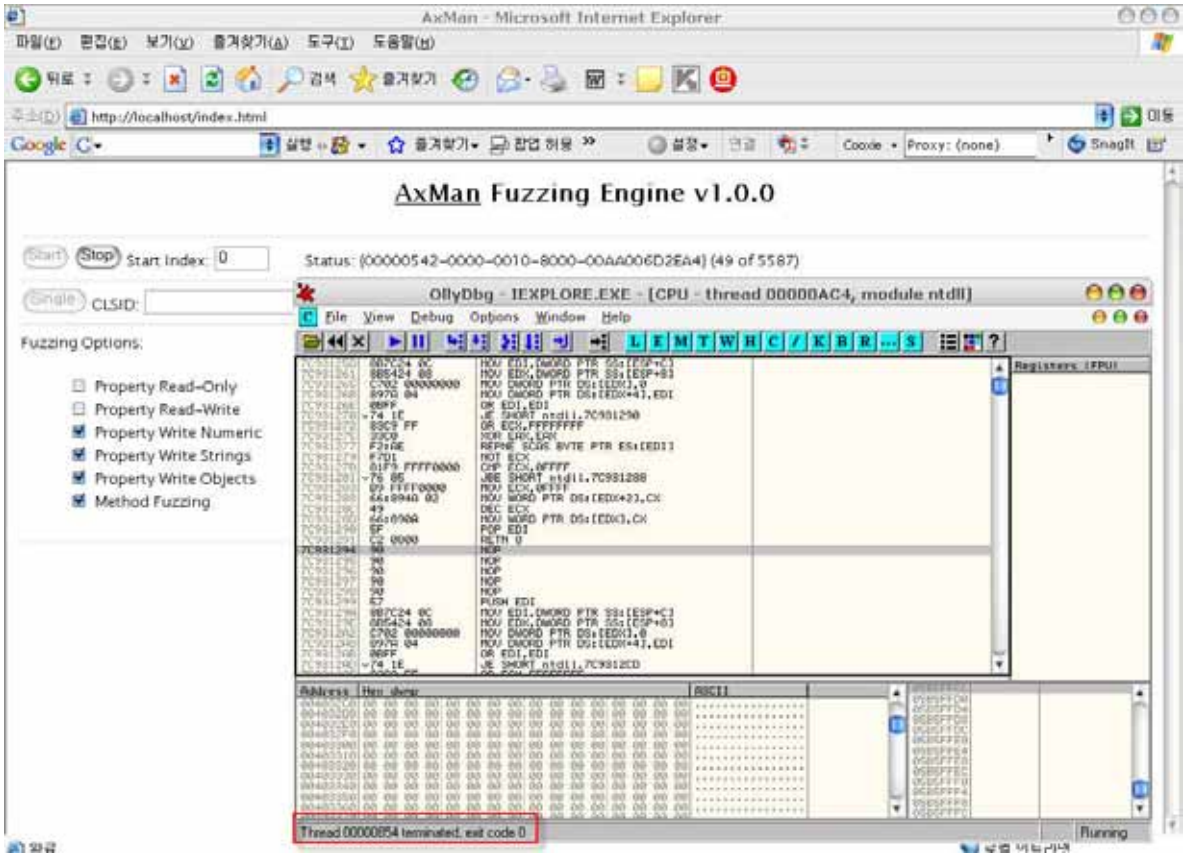
디버거를 attach한 후에 프로세스를 지속시키기 위해 F5를 누르고, Intetnet Explorer로 돌아간다.

F5를 누른 후 Start 버튼을 누른다. AxMan은 현재의 CLSID와 테스트된 속성과 방법을 브라우저의 상태 바와 보고할 것이다.

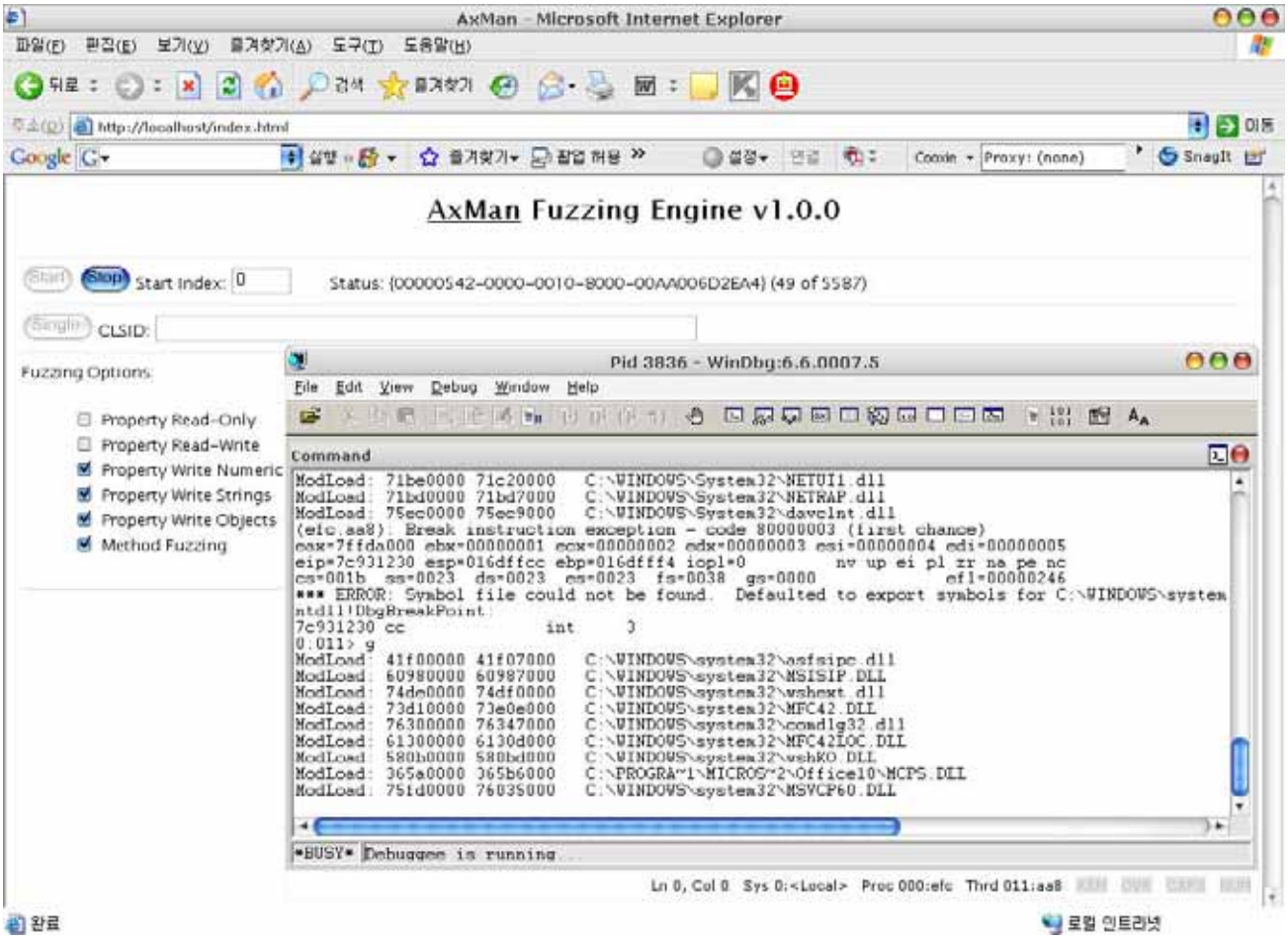
만약 OllyDbg를 이용할 경우 다음과 같이 OllyDbg를 실행한 후 iexplore.exe를 attach 한다. 먼저 메뉴 바에서 File 부분을 클릭한 후 attach를 클릭하면 실행 중인 프로세스 목록이 나오는데, 여기서 iexplore를 선택한 후 attach를 클릭한다.



그런 다음 F9를 누르고, Start 버튼을 클릭한다. 이제 분석이 시작된다. 매 번 생성되는 프로세스의 Thread에 대한 것이 다음 그림의 빨간색으로 표시한 부분에 나온다.



이제부터 지루한 분석 과정을 지켜보아야 한다. 만약 refresh를 시키면 디버깅 과정이 처음부터 시작되므로 유의해야 한다.

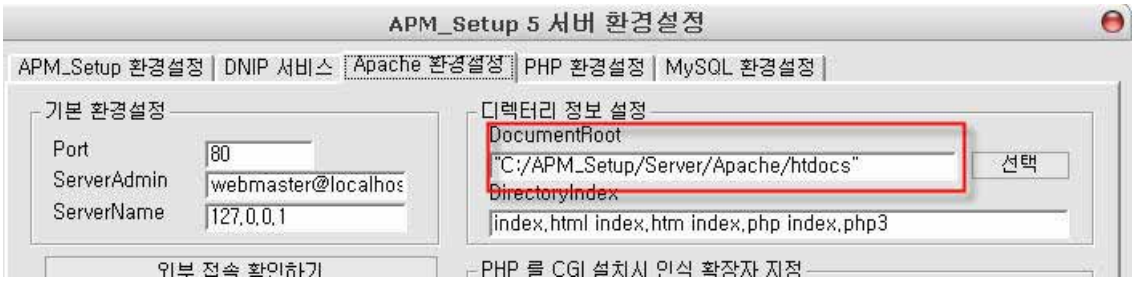


AxMan은 브라우저의 상태 바에 현재의 CLSID와 테스트된 속성이나 방법을 보고한다. 이 상태 바는 테스트 과정 중에서 실시간으로 업데이트된다.

어떤 구성요소가 crash되고, blacklist에 속성과 방법이 추가되는지 실시간으로 확인하기 위해 웹 로그를 확인을 하는데, 이를 위해 tail을 실행한다. Windows용 tail을 구하여 실행하면 된다. Windows용 tail은 Sourceforge에서 구할 수 있다. 다음 주소에서 Windows용 tail을 다운받아 압축을 푼다.

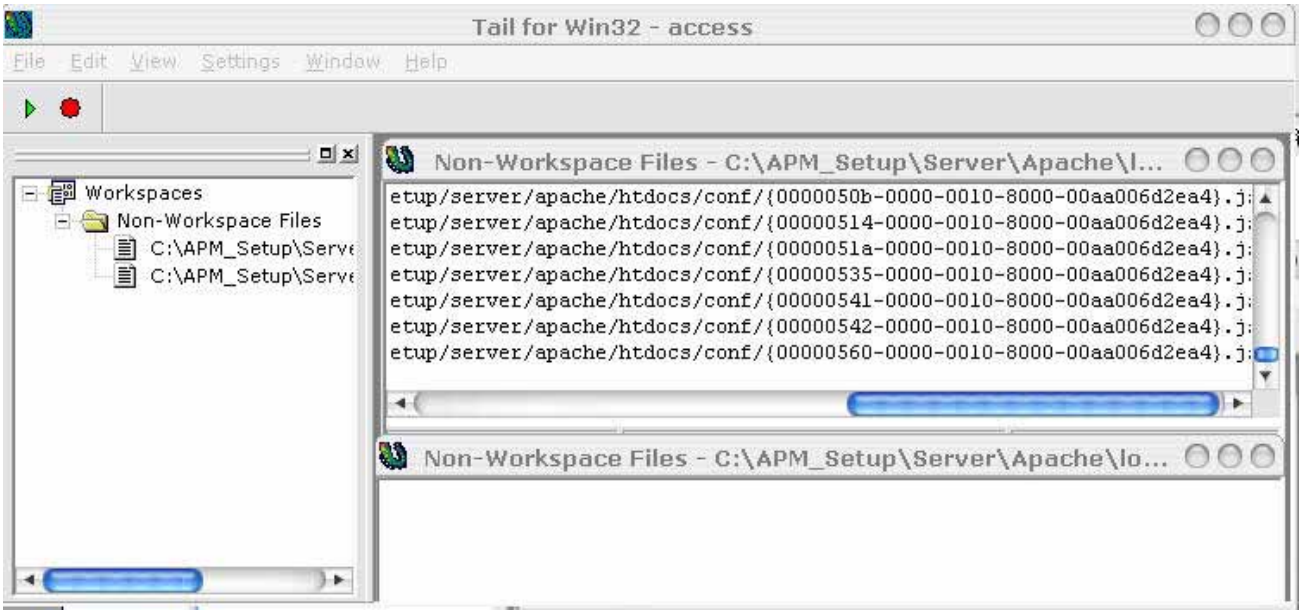
http://sourceforge.net/projects/tailforwin32

APM_SETUP를 설치할 때 **Apache 환경설정**의 **디렉터리 정보 설정**에서 다음과 같이 DocumentRoot를 "C:/APM_Setup/Server/Apache/htdocs"로 설정하는 것이 좋다.



이것은 별도의 로그 디렉토리와 파일을 만들 필요가 없기 때문이다. access와 error 로그 파일이 "C:/APM_Setup/Server/Apache/logs"에 이미 만들어져 있다. 그래서 tail을 실행하면 다음과 같은 창이 뜬다. 잘 알겠지만 여기서 실행된 tail 프로그램의 내용은 리눅스 등의 “tail –f” 와 같다.

여기서 File을 누른 다음 Open을 클릭하고, 그 다음 "C:/APM_Setup/Server/Apache/logs"에 있는 access와 error 로그를 동시에 보면 된다. 다른 자세한 사용법은 스스로 확인하길 바란다. 다음은 tail을 실행했을 때의 모양이며, access와 error 로그를 동시에 보고 있다.



이제 분석을 위한 준비가 거의 다 되었다. 이제 결과를 지켜보아야 한다. 문제가 되는 부분을 발견한다면 디버거를 통해 더욱 자세하게 분석해야 정확한 취약점을 찾아낼 수 있을 것이다. 결국 이 fuzzer를

이용한다고 해서 완벽하게 정리된 취약점을 이 프로그램이 보여주지 않는다는 것이다. 분석가의 손에 완벽한 취약점 발견이 달려 있는 것이다. 디버깅과 관련된 공부가 필요할 것이다.

[BLACKLIST]

Blacklist.js 스크립트 파일은 fuzzing 과정에서 건너뛸 모든 오브젝트, 속성, 그리고 방법들을 포함하고 있다. 취약점을 발견했다면 그 부분에 해당되는 것은 fuzzing 과정에서 제외시킬 필요가 있다, 일부러 많은 시간과 메모리를 소비할 필요가 없기 때문이다.

[Tracer]

AxMan은 속성과 방법 fuzzing 작동 방식의 일부로 독특한 문자열을 사용한다. 이것은 특정 속성이 설정되거나 또는 방법이 호출될 때 어떤 시스템 차원의 작동이 발생하는지 결정하기 위해 filemon이나 regmon과 같은 툴을 사용할 수 있게 한다. 그 디폴트 마술 문자열은 'AXM4N'으로 설정되어 있지만 이것은 axman.js에서 변경될 수 있다.