Attacking Antivirus

By Feng Xue

편역: vangelis@securityproof.org

ABSTRACT

안티바이러스 솔루션은 이제 컴퓨터 시스템의 일반적인 구성요소이다. 하지만 안티바이러스 소프트웨어 그 자체에 존재하는 보안 문제는 안티바이러스 벤더와 컴퓨터 사용자의 충분한 주목을 여태까지는 받지 못했다.

이 글은 왜 안티바이러스 소프트웨어가 다양한 공격에 취약하고, 왜 그 보안이 아주 중요한지 다룰 것이다. 이 글에서는 안티바이러스 솔루션들에 존재하는 취약점들을 찾아내는 공격자들이 사용하는 몇 가지 툴과 테크닉, 특히 fuzzing 테크닉을 알아볼 것이다. 그리고 공격자가 안티바이러스 솔루션의 취약점들을 공격하는 방법들을 살펴볼 것이다.

궁극적으로 이 글은 보안 제품의 보안에 대한 의식수준을 높이는데 그 목표가 있다.

키워드: Antivirus, Audit, Exploitation, Fuzzing[1], Security product

1. 도입

그림1과 같이 미국 National Vulnerability Database[2]에 따르면 165개의 취약점이 과거 4년 동안 안티바이러스 소프트웨어에서 보고되었다.

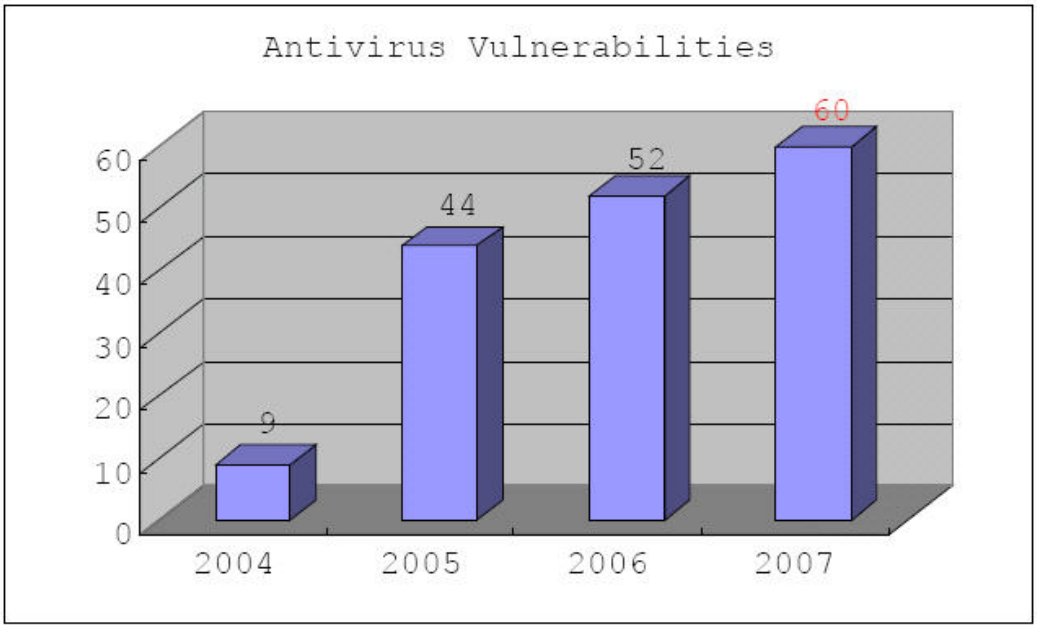


그림1. National Vulnerability Database

---

[1] (역자 주) Fuzzing에 대한 한국어 자료는 Securityproof 오프라인 세미나에서 발표된 자료들이나 잡지 "**해킹과 보안**"의 창간호에 실린 fuzzing 관련 글을 참고하길 바란다.

[2] http://nvd.nist.gov/home.cfm

그래서, 안티바이러스 소프트웨어도 컴퓨터 시스템의 다른 구성요소나 서비스들과 마찬가지로 공격의 대상이 될 수 있다는 것이 분명하다.

섹션2는 왜 안티바이러스 소프트웨어가 공격에 취약한지 다룰 것이다. 섹션3은 안티바이러스 소프트웨어의 취약점들을 분석하는데 사용되는 테크닉들, 예를 들어 소스 코드 분석, 또는 리버스 엔지니어링(reverse engineering), 그리고 퍼징(fuzzing) 등을 다룰 것이다. 공격 테크닉은 섹션4에서 다룰 것이다.

2. 안티바이러스 소프트웨어가 완벽한 공격대상이 되는 이유

2-1. 안티바이러스 소프트웨어에 대한 사람들의 완벽한 믿음

안티바이러스 소프트웨어 사용은 신념 행위의 어떤 것이 되었다. 사람들은 더 안전한 운영체제 또는 최신 패치보다는 시스템에 설치된 어떤 안티바이러스 소프트웨어에 더 안전함을 느끼는 듯하다.

모든 컴퓨터 사용자들의 81%가 그들의 컴퓨터에 안티바이러스 소프트웨어를 설치하고 있다라는 최근 연구[3]결과가 나왔다. 확실히 안티바이러스 소프트웨어는 대부분의 사용자들에게 필수적인 것이 되었다.

그러나 문제들이 있다. 안티바이러스 소프트웨어만으로 충분한가? 그와 같은 맹목적인 믿음은 정당화될 수 있는가? 공격자가 운영체제 대신 안티바이러스 소프트웨어 그 자체를 공격한다면 어떻게 될까?

실행파일, 문서 파일, 미디어 파일 등과 같은 파일을 가지고 있는 평균적인 사용자를 예를 들어보자. 그의 컴퓨터에 설치된 안티바이러스는 컴퓨터로 들어오는 파일들을 자동으로 스캐닝할 것이다(사용자는 만약 파일이 의심스러우면 직접 스캐닝을 실행할 수 있다.). 이때 안티바이러스는 incoming 파일들에 대해 보안 게이트로 역할을 할 것이다. 그림2를 보자.

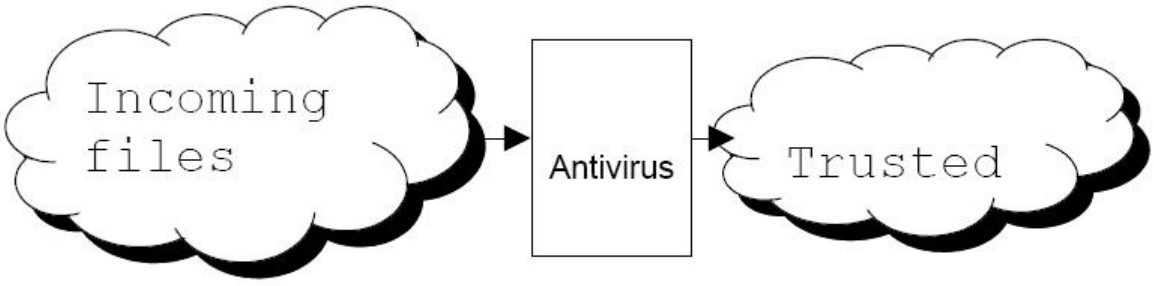


그림2. AV 안티바이러스는 들어오는 파일들에 대해 보안 게이트 역할을 함

[3] http://www.bsacybersafety.com/news/2005-Holiday-Online-Shopping.cfm

사용자가 모르는 것은 많은 안티바이러스 솔루션들이 과거에 개발되었으며, 전반적인 보안을 염두에 두지 않고 개발되었다는 것이다. 개발자들은 신뢰받지 못한 파일들이 그들이 개발한 소프트웨어에 의해 안전하게 스캐닝 된다는 것을 추정한다. 하지만 바로 그 파일들이 그들의 솔루션 소프트웨어 그 자체에 피해를 줄 수 있다면 어떻게 할 것인가?

안티바이러스 보안에 대한 위협은 다음 두 가지에 의해 강화된다.

* 사용자들이 안티바이러스를 특효약인 것처럼 맹목적으로 받아들임
* 모든 파일들에 대해 자신의 안티바이러스 제품은 감염되지 않는다는 것에 대한 안티바이러스 제조업체의 지나친 자신감

2-2. 안티바이러스 프로세스들의 에러발생 가능성

안티바이러스 소프트웨어는 가장 복잡한 어플리케이션들 중의 하나이다. 안티바이러스 소프트웨어는 많은 파일 타입과 포맷을 다루어야 한다:

* 실행 파일: exe, dll, msi, com, pif, cpl, elf, ocx, sys, scr, 등
* 문서 파일: doc, xls, ppt, pdf, rtf, chm, hlp, 등
* 압축 파일: arj, arc, cab, tar, zip, rar, z, zoo, lha, lzh, ace, iso, 등
* 실행 가능한 패커: upx, fsg, mew, nspack, wwpack, aspack, 등
* 미디어 파일: jpg, gif, swf, mp3, rm, wmv, avi, wmf, 등

이 파일 포맷들 각각은 아주 복잡할 수 있다. 그래서 안티바이러스 소프트웨어가 이 모든 포맷을 적절하게 처리하는 것은 아주 어렵다.

이것은 안티바이러스 취약점들에 대한 최근 연구를 보면 아주 명백하다. 대부분의 취약점들이 다음 두 요소에 존재한다는 것을 최근 연구를 통해 알 수 있다.

* 실행파일 압축 해제[4]
* 데이터 압축 해제[5]

안티바이러스 소프트웨어는 압축된 실행파일과 데이터를 처리하기 전에도 압축을 풀려고 시도할 것이다.

---

[4] http://en.wikipedia.org/wiki/Executable_compression

[5] http://en.wikipedia.org/wiki/Data_compression

실행파일과 데이터의 압축을 푸는 것의 문제는 그 처리과정이 아주 복잡하다는 것이다. 안티바이러스는 복잡한 연산을 하고, 메모리를 할당하고, 그 연산에 따라 데이터를 추출한다. 이 과정에서 어떤 실수가 발생한다면 취약점으로 이어질 수 있다.

이런 점이 공격자들이 안티바이러스 솔루션을 공격 대상으로 삼게 하는데 충분하지만 안티바이러스라는 '갑옷'에 존재하는 '구멍'을 어떻게 발견하는가?

3. 안티바이러스 취약점 찾기

기본적으로 안티바이러스 소프트웨어에는 4가지 종류의 취약점이 있다.

* 로컬 권한 상승 취약점
* ActiveX 관련 취약점
* 엔진 기반의 취약점
* 관리 인터페이스 관련 취약점

이 취약점들에 대한 안티바이러스 프로세스를 점검하는 방법론은 서로 아주 다르다.

3-1. 로컬 권한 상승 문제 점검

3-1-1. 취약한 DACL

취약한 DACL을 점검하는 것은 가장 쉬운 것 중의 하나인데, 가끔 어떤 툴의 도움 없이 직접 할 수 있다.

안티바이러스 설치 디렉토리(파일들)가 이 문제에 취약한지 여부를 확인하기 위해 그 디렉토리(파일들)에 오른쪽 클릭을 하여 "보안" 태그를 따라 들어간다. 만약 "Everyone" 그룹이 전체를 통제할 수 있는 퍼미션 권한을 가지고 있다면 그것은 아마도 새로운 로컬 root 취약점이 될 수 있다.

서비스들의 ACL을 점검하는 것은 설치 디렉토리와 약간 다르다. 이를 위해 Microsoft의 sc.exe[6]를 이용할 수 있으며, 그 접근 방식은 다음과 같다.
권한이 없는 사용자로 로그인하여 다음 명령을 실행한다.

C:₩sc config "antivirus service" binpath=D:₩attack₩attack.exe

---

[6] http://technet2.microsoft.com/windowsserver/en/library/0a658e97-51d5-4109-b461-a474c799964e1033.mspx

만약 에러 번호 5가 뜨지 않는다면, 축하한다!

"antivirus service"는 공격자가 공격하고자 원하는 서비스의 이름이며, binpath는 공격자가 통제하고 있는 바이너리 파일(트로이, 루트킷, 또는 그 어떤 것)이다.

이 명령을 실행한 후 공격자는 Windows 서비스의 바이너리 경로를 성공적으로 변경했을 것이며, 이것은 공격자의 바이너리가 상승된 권한(일반적으로 SYSTEM 권한)으로 실행되는 결과로 이어진다.

3.1.2 Driver IOCTL handler

보안 연구자들은 2007년에 드라이버의 많은 문제점들을 발견했다.

큰 문제들 중의 하나가 IOCTL 드라이버 핸들러 문제이다. 이 문제는 디바이스 드라이버의 IOCTL 핸들러 내부에 충분하지 못한 주소 공간 검사에 의해 발생한다.

Fuzzing을 통해서도 쉽게 IOCTL 드라이버 문제를 점검할 수 있다. ioctlizer[7]와 같은 몇몇 Win32 IOCTL fuzzer들이 공개되어 있다.

ReverseMode의 Kartoffel[8] 역시 IOCTL 드라이버의 보안과 신뢰성을 테스트하는데 목적을 둔 툴이다.

그래서 우리가 해야 할 것은 안티바이러스 소프트웨어에 의해 설치된 드라이버를 퍼징하고, 모든 BSOD(blue screen of death)를 주의 깊게 조사하는 것이다.

3-2. ActiveX 문제 점검

안티바이러스 소프트웨어의 ActiveX 문제를 점검하는 것은 다른 어플리케이션의 ActiveX 문제를 점검하는 것과 다르지 않다. 이 문제를 점검하기 위해 fuzzing을 이용하거나 또는 직접 점검하는 방법이 있다.

*** Fuzzing**

ActiveX 기반의 취약점들은 전보다 2007년에 더 많이 유행했다. 이것은 부분적으로 ActiveX fuzzer들이 유행했기 때문이다. 이 분야에서 인기 있는 툴 2개를 고른다면 AxMan[9]과 ComRaider[10]가 있다. AxMan이 더 강력하며, 반면 ComRaider는 사용자 친화적이다. 안티바이러스 소프트웨어를 설

---

7 http://code.google.com/p/ioctlizer/

8 http://kartoffel.reversemode.com/

9 http://www.metasploit.com/users/hdm/tools/axman/

10 http://labs.idefense.com/software/fuzzing.php#more_comraider

치한 후 단일 CLSID를 선택하거나 디렉토리를 지정함으로써 특정 ActiveX control을 퍼징할 수 있다.

*** 직접 점검(manual auditing)**

fuzzing이 많은 'memory corruption' 문제를 찾아낼 수 있는 반면, 직접적인 점검은 디자인 레벨에서의 다른 흥미로운 취약점들, 예를 들어 unsafe-method 등을 드러낼 수 있다.

직접적인 점검에 사용되는 툴들에는 Oleview[11], Filemon[12], RegMon[13], TcpView[14], 그리고 Wireshark[15] 등이 있다.

OleView는 레지스트리에 포함되어 있는 정보들을 더 높은 차원에서 볼 수 있도록 하고, 친숙한 이름들로 된 tree control의 기능을 가지고 있다. 또한 어떤 control이 안전한 것으로 표시되어 있는지 아닌지 점검하거나 제공된 method들을 평가하는데 사용될 수 있다.

FileMon과 RegMon 역시 아주 유용하다. 이 둘은 어떤 ActiveX control이 초기화될 때 어떤 종류의 오퍼레이션들과 레지스터리 오퍼레이션이 일어나고 있는 지를 점검하는데 사용될 수 있다. ActiveX control은 사용자가 통제하는 파라미터들에 의해 지정된 어떤 파일/레지스트리 key들을 읽고, 쓰려고 시도하는가?

TcpView는 어떤 ActiveX가 어떤 네트워크 활동을 가지고 있는지 여부를 확인해준다. 이 ActiveX control은 어떤 TCP/IP 포트를 listen하고 있는가? 그것이 파라미터에 의해 지정된 어떤 IP 주소에 연결하려고 하는가?

전체 분석 과정 동안 Wireshark를 계속 실행시키는 것을 추천한다. 이것은 ActiveX control이 어떤 웹사이트, IP 주소에 연결하려고 시도하는지 여부를 알려주거나 또는 어딘가로부터 어떤 파일을 다운로드하여 실행하려고 하는지 여부를, 또는 당신의 컴퓨터로부터 어딘가로 파일을 업로드하려고 하는지 여부를 알려줄 수 있다.

3-3. 안티바이러스 소프트웨어 엔진 점검

엔진은 안티바이러스 소프트웨어의 가장 복잡한 구성요소라서 엔진을 점검하는 것은 힘든 일이 될 수 있다.

---

[11] http://msdn2.microsoft.com/en-us/library/ms693754(VS.85).aspx

[12] http://technet.microsoft.com/en-us/sysinternals/bb896642.aspx

[13] http://technet.microsoft.com/en-us/sysinternals/bb896652.aspx

[14] http://technet.microsoft.com/en-us/sysinternals/bb897437.aspx

[15] http://www.wireshark.org/

기본적으로 세 가지 점검 방법이 있다:

*** 소스코드 점검**

말 그대로 소스코드를 점검하는 방법이고, 점검하는 사람은 안티바이러스 소프트웨어의 소스 코드에 접근할 필요가 있다.

하지만, ClamAV[16]를 제외하고는 보안 연구자들이 대부분의 소스코드를 구하는 것은 힘들다.

ClamAV는 소스 코드 점검에는 아주 좋은 타깃이 된다. 취약점 관련 데이터를 보면 ClamAV에 대해 CVE에 49개가 등록되어 있다.

말할 필요도 없이 소스 코드 점검은 시간을 소비하는 일이다. 소스 코드 점검을 위해 FlawFinder, RATS, ITS4, SPLINT, CodeScan, 그리고 Coverity 등과 같은 좋은 툴들이 있다.

직접 소스 코드를 분석하면서 위험한 API와 정수 계산, 메모리 작동, 그리고 논리적 오류 등을 찾아낸다.

*** Reversing engineering**

대부분의 상용 안티바이러스 솔루션들은 소스가 공개되어 있지 않기 때문에 보안 연구자가 소스 코드를 분석하는 것은 거의 불가능하다.

리버싱 엔지니어링은 최선의 선택들 중 하나이다. 연구자들은 어셈블리어 코드를 직접 분석하고, 잠재적인 취약점들을 찾아낸다.

반면 안티바이러스 소프트웨어 엔진을 리버싱 엔지니어링으로 분석하는 동안 모든 종류의 파일 포맷들을 분석하는데 필요한 구성요소에 초점을 맞추어야 한다.

이 구성요소들은 보통 독립된 플러그인으로 구현되어 있으며, 여기에 두 가지 예가 있다.

- Kaspersky: Arj.ppl, base64.ppl, cab.ppl, lha.ppl, rar.ppl 등
- Bitdefender: arc.xmd, arj.xmd, bzip2.xmd, cab.xmd, docfile.xmd 등

다른 리버싱 엔지니어링 작업과 마찬가지로 리버싱 엔지니어링을 통해 안티바이러스 엔진을 점검하는 것은 아주 많은 시간을 요구한다. 좋은 소식은 Hex-rays[17](IDA pro plug-in)가 decompiling을 훨씬 더 쉽게 만든다는 것이다.

---

[16] http://www.clamav.net/

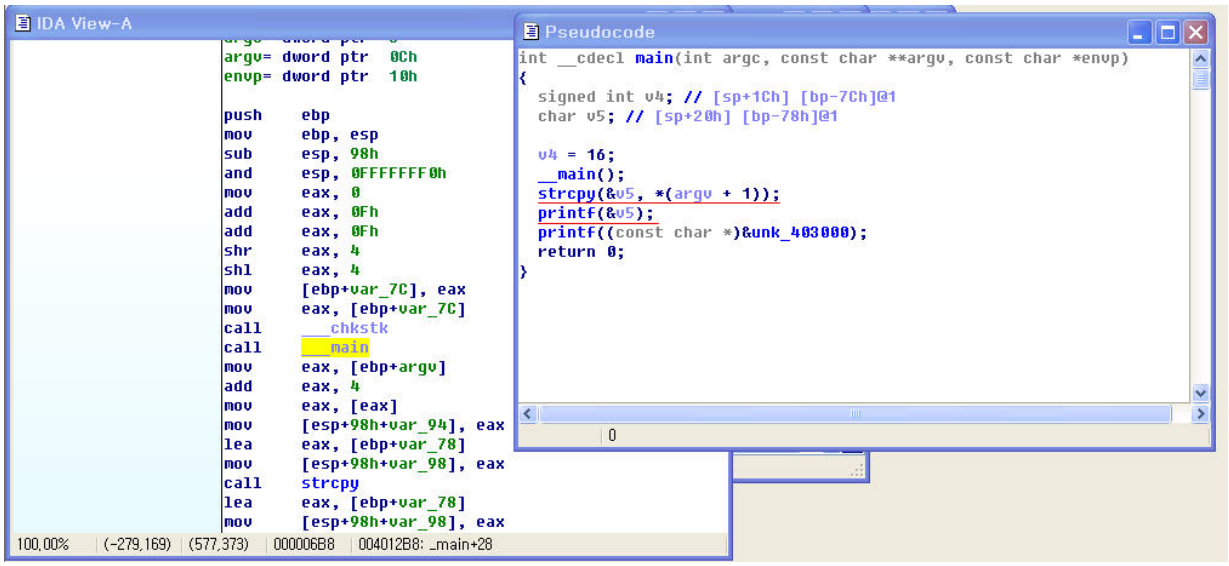
[17] http://www.hex-rays.com/

******** **[역자 첨가 – Hex-rays를 사용했을 경우]** **********

다음은 오버플로우와 포맷 스트링 취약점을 가진 간단한 소스이다. 소스 컴파일은 Dev-C++로 하였다.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
	char buf[100];

	strcpy(buf, argv[1]);
	printf(buf);

	printf("₩n");

	return 0;
}
```

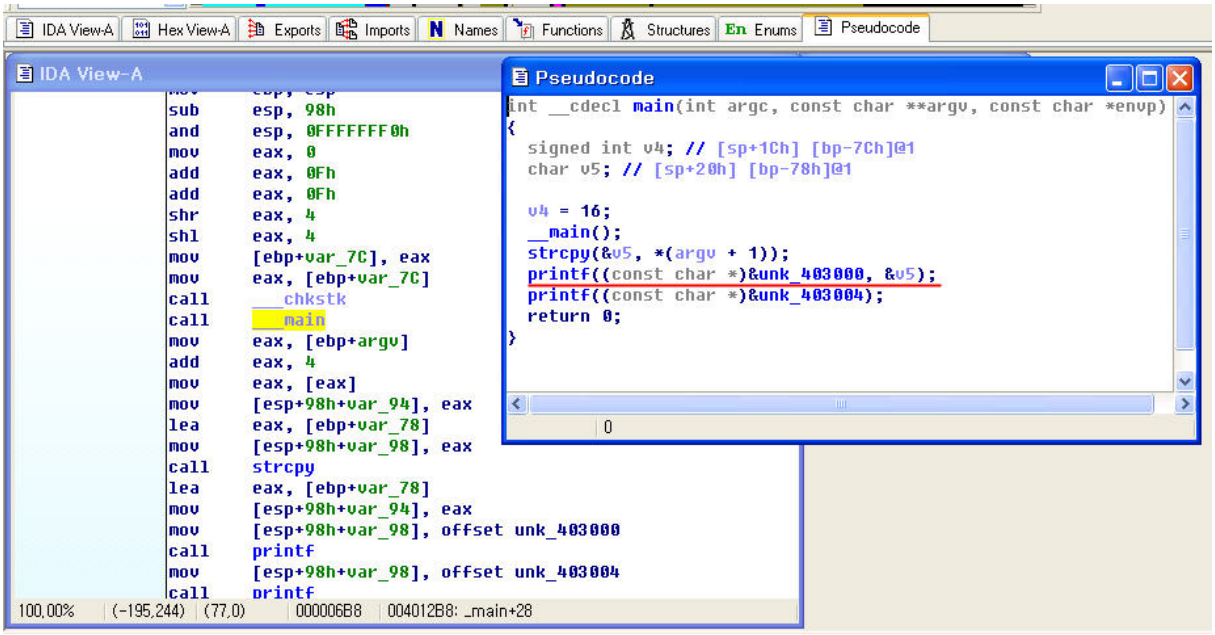
이것을 Hex-rays를 사용해 IDA로 분석한 결과이다. Hex-rays를 설치하면 view 섹션에 Pseudocode가 생긴다. 다음은 Pseudocode로 본 것이다.



위의 그림에서 오른쪽 창 Pseudocode를 보면 스택 기반의 오버플로우와 포맷 스트링 취약점이 존재함을 알 수 있다. Hex-rays를 사용하면 선언된 변수 부분에 대해서는 다소 정확하지 않으나

어떤 함수가 어떻게 사용되고 있는지는 정확하게 나온다. 그리고 위의 소스 코드에서는 'char buf[100]'이라고 선언되었고, Pseudocode 결과를 보면 'char v5'로 나온다. 이 결과를 보면 적어도 변수의 타입은 알 수 있다. 그리고 참고할 수 있는 다른 정보도 있으므로 변수 선언 부분의 다소 완벽하지 않음은 크게 문제 될 것 같지는 않다.

소스 코드 중에서 포맷 스트링 취약점 부분을 " printf("%s₩n",buf); "로 수정 후 다시 컴파일 하여 그 결과를 보면 다음과 같다.



포맷 스트링 취약점을 가지고 있을 때는 "printf(&v5);" 였던 것이 "printf((const char *)&unk_403000, &v5);"로 수정되었다. 이것을 봐도 Hex-rays의 기능이 막강하다는 것을 알 수 있다. 소프트웨어에 존재하는 취약점을 발견하는데 IDA pro와 Hex-rays의 만남은 연구자들에게 많은 도움을 줄 수 있을 것이다.

****

Alex Wheeler는 이 분야에서 뛰어난 작업을 해왔다. 더 많은 정보는 그의 Black Hat 발표 자료 "Owing Anti-Virus"[18]를 확인해보길 바란다.

*** Fuzzing**

fuzzing은 놀라운 테크닉이며, 과거 몇 년 사이에 소프트웨어 보안을 촉진시켜왔다.

연구자들은 이미 미디어 플레이어들과 웹 브라우저 등을 퍼징하는 것을 발표했으며, 많은 fuzzer

[18] http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-wheeler.pdf

들이 발표되었다. 그러나 안티바이러스 소프트웨어를 퍼징하는 것에 대해서는 논의가 많이 되지 못했다.

필자가 알기로는 공개된 안티바이러스 소프트웨어 fuzzer는 겨우 하나밖에 없다. 그것은 Tavis Ormandy에 의해 공개된 vxfuzz[19]이다.

n.runs사는 전에 Fuzzer-Framework v1.0[20]이라는 이름이 붙은 회사 내에서만 사용되는 fuzzer를 가지고 있다고 언급했다.

대부분의 엔진 기반의 취약점들이 압축을 푸는 과정에서 존재하므로, 안티바이러스 엔진을 퍼징한다는 것은 다양한 압축이 풀린 데이터와 실행파일을 퍼징하는 것을 의미하며, 이것이 그 밖의 다른 것을 퍼징하는 것보다 훨씬 더 쉬우며, 필자의 관점에서 퍼징을 위해 필요한 것들은 다음과 같다:

A. 대용량의 하드 디스크
   이것은 안티바이러스 소프트웨어가 많은 다른 파일 포맷을 다루어야 하기 때문이며, 지원되는 모든 파일 포맷을 퍼징하는 것이 좋다. 대용량의 하드 디스크가 필요한 것은 fuzzer에 의해 생성된 모든 파일들을 저장해야 하기 때문이다.

B. Debugger
   디버거(이 작업을 위해서는 windbg를 추천한다)를 선택해라. Ollydbg와 Immunity Debugger 둘 다 좋은 선택이다. 디버거의 사용법은 나중에 소개된다.

C. Fuzzer
   실제로, 안티바이러스 fuzzer를 구현하는 것은 아주 쉽다. 안티바이러스 fuzzer는 exception을 핸들링 하거나 반복적으로 타깃 어플리케이션을 시작시킬 필요가 없다. 이것이 다른 어플리케이션(웹 브라우저, 미디어 플레이어 등)을 퍼징하는 것과 안티바이러스 소프트웨어를 퍼징하는 것 사이의 주요 차이점이다.

   스크립트 언어(파이썬 또는 펄)를 이용해 fuzzer를 만드는 것은 빠르고 재미있다. 하지만, 만약 생성할 파일이 아주 많을 경우 C 언어가 성능의 관점에서 완벽할 것이다.

   실제로, 안티바이러스 소프트웨어용 fuzzer는 그냥 파일 생성기에 불과하다. 예를 들어, 파일들을 생성하기 위해 아주 간단한 안티바이러스 fuzzer는 샘플을 읽고, 그 샘플을 퍼징 문자열로 바이트씩 대체하고, 그런 다음 지정된 디렉토리에 그것들을 저장하기만 하면 된다.

---

[19] http://my.opera.com/taviso/blog/

[20] http://events.ccc.de/camp/2007/Fahrplan/attachments/1324-AntivirusInSecuritySergioshadownAlvarez.pdf

퍼징 문자열을 선택하는 것은 보통 디자이너의 경험에 기반을 두고 있다. 그 퍼징 문자열은 세그먼트 폴트(integer overflow, 스택 기반의 overflow, 등)를 일으킬 수 있는, 0xFFFFFFFF, 0x7FFFFFFF, 0x0000, 0x00000000, 'B'*256, 0xCCCCCCCC와 같은 매직 값을 포함하고 있어야 한다.

CRC checksum에 대해서는 주의를 해야 한다. rar와 zip과 같은 파일 포맷들에 대해서 안티바이러스 소프트웨어는 먼저 그 파일(또는 어떤 부분)의 CRC 체크를 하고, 그리고 매치가 되지 않으면 더 이상의 프로세스가 일어나지 않는다.

특정 파일 포맷들에 대해서는 맞춤식 CRC 함수가 fuzzer 내부에 구현될 수 있다.

D. 좋은 샘플들

좋은 샘플들은 퍼징에 중요하다. 안티바이러스 소프트웨어는 아주 많은 파일 포맷들을 처리하기 때문에 점검자는 가능한 많은 샘플들을 수집할 필요가 있다.

구글에서 "파일타입: 확장자"를 검색해보아라. 그 샘플들은 직접 만들 수 있다. 그래서, 점검자는 모든 소프트웨어가 필요했던 것들 – WinRAR, PowerISO, MakeCAB, 그리고 다양한 PE packer(UPX, FSG, ASPack, 등)을 수집할 필요가 있다.

일단 이것들이 준비가 되면 다음 4단계를 따라야 한다.

A. 테스트 케이스 만들기

이것은 안티바이러스 fuzzer(파일 생성기)를 이용해 할 수 있다. 생성된 테스트 케이스들은 지정된 디렉토리(대용량의 하드 디스크)에 저장되어야 한다.

B. 안티바이러스 소프트웨어 다운로드 및 설치

퍼징할 안티바이러스 소프트웨어를 다운로드하여 설치한다. 안티바이러스 벤더들은 보통 그들의 웹사이트에 트라이얼 버전을 제공하고 있으므로, 점검자는 그것을 다운로드하여 설치할 수 있다. 설치 후 스냅샷을 찍어두는 것을 잊지 마라.

C. 스캔

테스트 케이스에 대해 스캐닝을 한다. 이때 안티바이러스 소프트웨어의 스캐닝 프로세스에 디버거를 attach해라. 스캐닝 프로세스가 어떤 것인지 구분하기 힘들면 스캐닝을 시행하여 CPU 사용도를 확인한다.

D. 잠을 좀 자라

이 스캐닝 과정은 테스트 케이스의 수에 따라 몇 시간 또는 며칠이 걸릴지도 모른다. 잠을 자고, 일어나자마자 exception이 발생했는지 디버거를 확인해라. 만약 exception이 발행했다면 각 exception을 깊게 분석하여 공격이 가능한지 확인해야 한다.

3.4 점검 관리 인터페이스

* 클라이언트/서버 관리
대부분의 클라이언트/서버 기반의 관리 프로토콜들은 벤더가 독점적으로 소유하고 있어 관련 RFC나 문서들을 구하기가 쉽지 않다. 패킷을 캡처하여 클라이언트와 서버가 어떤 통신을 하고 있는지 이해하는 것은 어려울 것이다. 트래픽은 아주 랜덤하게 보이거나 어떤 방식으로든 암호화되어 있을 수 있다.

이와 같은 상황에서는 fuzzing이 좋은 선택이 될 수 있다. Spike[21]와 Sulley[22]는 뛰어난 fuzzing framework이다. 더 많은 정보는 레퍼런스를 참고해라.

* 웹 인터페이스
대부분의 관리용 웹 서버들은 안티바이러스 업체에 의해 내부적으로 개발되기 때문에 제대로 점검이나 분석이 되지 않을 수 있다. 그래서 Fuzzing은 항상 유용하고 시도할 가치가 있다. Webfuzz[23], Spike, 그리고 Sulley와 같이 공개적으로 구할 수 있는 많은 웹 fuzzer들이 있다.

안티바이러스 소프트웨어를 점검함으로써 필자는 유명한 안티바이러스 프로그램들에서 몇 가지 취약점들을 발견했다. 대부분의 이 취약점들은 fuzzing을 통해 발견된 것이다.

* AhnLab AV Remote Kernel Memory Corruption
* TrenMicro AV UUE Decoding Format String Vulnerability
* Avast! AV TGZ Parsing Heap Corruption
* NOD32 Heap Overflow(이 글을 쓰고 있는 현재 공개되지 않은 0day임)

앞으로 안티바이러스 제품에 존재하는 취약점들이 더 많이 발견될 것으로 예상된다.

4. 안티바이러스 공격

안티바이러스 취약점들을 공격하기 위해 공격자들이 사용하는 테크닉은 경우에 따라 다르다.

4.1 로컬 권한 상승

안티바이러스 소프트웨어가 직면하고 있는 로컬 권한 상승 문제는 다른 소프트웨어의 그것과 다르지 않다. 이 문제들은 다음과 같이 분류할 수 있다:

---

[21] http://www.immunitysec.com/resources-freesoftware.shtml

[22] http://code.google.com/p/sulley/

[23] http://fuzzing.org/wp-content/WebFuzz.zip

4.1.1 취약한 DACL

취약한 DACL 문제는 설치 디렉토리와 설치된 서비스들에서 발생했다.

설치 디렉토리가 문제가 되는 이상 설치 동안 적용될 ACL(Access Control List) 설정에 취약점들이 존재한다. 안티바이러스 소프트웨어가 "Everyone" 그룹에 "Full Control" 퍼미션을 줄 경우 어떤 사람도 설치된 파일들을 수정할 수 있다. 거의 모든 안티바이러스 소프트웨어들은 몇몇 시스템 서비스들을 실행하고, 이 사실 때문에 공격자들은 시스템에 설치된 서비스 파일을 SYSTEM 권한을 가지고 나중에 실행될 수 있는 자신의 악의적인 코드(트로이 또는 루트킷)로 간단하게 대체할 수 있다.

이 문제는 McAfee, Symantec, TrendMicro, VBA32, Panda, PC Tools, CA eTrust, ZoneAlarm, AVG, BitDefender, AVAST!, 그리고 Kaspersky에 국한되지 않고 거의 모든 안티바이러스 벤더들이 직면한 문제이다.

Case: CVE-2005-1107 *McAfee Internet Security Suite 2005 Insecure File Permission Vulenrability*[24]

McAfee Internet Security Suite 2005는 설치된 파일들에 대해 안전하지 않은 디폴트 ACL들을 사용하며, 로컬 사용자들이 어떤 파일을 수정함으로써 권한을 획득하거나 보호 장치를 무력화하는 것을 허용한다.

설치된 서비스들이 관련되어 있는 이상 취약점들은 SERVICE_CHANGE_COBFIG 퍼미션이 "Everyone"에 부여되어 있는 것 때문에 보통 발생한다. 공격자들은 관련 프로그램을 수정함으로써 상승된 권한을 획득하기 위해 이 취약점을 공격할 수 있다. 공격자는 Microsoft가 만든 SC.exe를 공격 과정에서 이용할 수 있다.

다음은 예이다.

C:₩sc stop "취약한 안티바이러스 서비스"
C:₩sc config "취약한 안티바이러스 서비스" binpath=D:₩attack₩attack.exe
C:₩sc start "취약한 안티바이러스 서비스"

취약한 DACL 문제는 과거 몇 년 사이에 드물어졌다.

4.1.2 Driver IOCTL Handler 문제

과거 2년 사이에 driver IOCTL handler 문제는 극적으로 증가했다. 보안 연구자들과 해커들은 2006년에 이 문제로 이동했으며, 많은 취약점들이 보안 제품들, 특히 안티바이러스 소프트웨어와

---

[24] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1107

개인 방화벽에서 발견되었다.

driver IOCTL handler 문제는 안티바이러스 소프트웨어에 의해 설치된 디바이스 드라이버들의 IOCTL handler 내에 충분하지 않은 주소 공간 검사에 의해 보통 발생한다. 공격자들은 임의의 메모리를 덮어쓰고, 그런 다음 커널 권한으로 임의의 코드를 실행함으로 driver IOCTL handler 문제를 이용할 수 있다.

보안 연구자들은 좀처럼 사용되지 않는 시스템 콜을 후킹하거나 GDT(Global Descriptor Table)에 있는 콜 게이트를 추가함으로써 이 문제를 안정적으로 공격할 수 있는 방법을 성공적으로 입증했다[25].

예:

CVE-2007-3673 ***Symantec Antivirus symtdi.sys Local Privilege Escalation Vulnerability***[26]

Symantec symtdi.sys 7.0.0 이전 버전은 \\symTDI\에 대한 IOCTL 0x83022323에서의 조작된 Irp(Interrupt Request Packet)을 통해 메모리 덮어쓰기가 가능하고, 이는 로컬 사용자들이 권한을 획득할 수 있도록 허용한다.

CVE-2007-0856 ***Trend Micro Products IOCTL Handler Privilege Escalation***[27]

TmComm.sys 1.5.0.1052는 \\.\TmComm DOS 디바이스 인터페이스에 대해 Everyone 쓰기 퍼미션을 부여하고, 이것은 로컬 사용자들이 권한을 가진 IOCTL에 접근하고, 임의의 코드 실행 또는 커널 상의 임의의 메모리 덮어쓰기를 허용한다.

CVE-2006-4927 ***Symantec Antivirus IOCTL Kernel Privilege Escalation Vulnerability***[28]

NAVENG.SYS와 NAVEX15.SYS 디바이스 드라이버 20061.3.0.12 및 후기 버전은 IOCTL 함수들 (1) 0x222AD3, (2)0x222AD7, 그리고 (3)0x222ADB에 조작된 Irp를 사용하여 중요한 시스템 주소들을 덮어씀으로서 로컬 사용자들이 권한을 획득할 수 있도록 한다.

CVE-2007-3777 ***AVG Antivirus AVG7CORE.SYS IOCTL Handler Privilege Escalation***[29]

Grisoft AVG Anti-Virus 7.5.448 및 Free Edition 7.5.446의 avg7core.sys 7.5.0.444는 임의의 주소에 데이터를 복사하는 내부 함수를 제공하는데, 이것은 generic DeviceIoControl handler에 대한 0x5348E004 IOCTL에 의해 제공되는 함수에 대한 임의의 주소 아규먼트들을 통해 권한을 획득할 수 있다.

---

[25] http://www.whitecell.org/list.php?id=50

[26] http://cve.mitre.org/cgi-bin/cvename.cgi?name=2007-3673

[27] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-0856

[28] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-4927

[29] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3777

4.1.3 Race condition

race condition 취약점은 Linux/Unix 플랫폼 상의 안티바이러스 소프트웨어에 보통 존재한다. 두 가지 경우가 있다:

CVE-2007-6595 ***Clam AntiVirus Race Condition Vulnerability***[30]

ClamAV 0.92는 (1) libclamav/others.c에서 cli_gentempfd 함수에 임시 파일을 또는 (2) utf16-decode가 가능할 때 sigtool의 .ascii 파일에 심볼릭 링크 공격을 통해 로컬 사용자들이 임의의 파일을 덮어쓰는 것을 허용한다.

CVE-2004-0217 ***Symantec AntiVirus Scan Engine for Red Hat Linux Insecure Temporary File Vulnerabilities***[31]

Red Hat Linux용 Symantec AntiVirus Scan Engine 4.0 및 4.3의 LiveUpdate 기능(livesupdate.sh)은 /tmp/LiveUpdate.log에 대해 심볼릭 링크 공격을 통해 로컬 사용자들이 임의의 파일을 생성하거나 또는 추가하는 것을 허용한다.

이 취약점들은 임시 파일들이 안전하지 않은 방식으로 생성되기 때문에 존재한다. 공격자들은 안티바이러스 소프트웨어가 심볼릭 링크가 걸린 파일을 덮어쓰도록 심볼릭 링크(시스템 상의 중요 파일로부터 임시 파일명까지)를 만들어 취약점을 공격할 수 있다. 이것은 공격자들이 시스템에 상승된 접근 권한을 획득하도록 해준다.

4.1.4 다른 문제들

과거에는 몇 가지 다른 문제들도 있었다. 다음은 그 예들이다.

* **Symantec** ***LiveUpdate***

  Symantec LiveUpdate는 로컬 권한 상승 문제로 오랫동안 문제를 겪어왔다. Symantec LiveUpdated에는 여태까지 거의 5개의 취약점들이 있었다: SYM04-018, CVE-2003-0994, CVE-2005-2759, CVE-2006-1836, CVE-2004-0217. 이 문제들은 신뢰받지 못한 검색 경로에서부터 SYSTEM 권한으로서 실행된 윈도우(window)까지 다양하다.

* **CVE-2006-3223** ***CA Antivirus Scan Job Description Format String Vulnerability***[32]

  스캐닝 작업을 위해 특별히 조작된 디스크립션(포맷 스트링을 포함하여)을 만들어 공격자들은 SYSTEM 권한으로 임의의 명령을 실행할 수 있다.

* **CVE-2007-3800** ***Symantec AntiVirus Corporate Edition Local Privilege Escalation Vulnerability***[33]

---

[30] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6595

[31] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0217

[32] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3223

[33] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3800

Symantec Real-Time 스캐너는 시스템에서 발견된 위협에 대한 정보를 알려주는 윈도우를 화면으로 표시하는 동안 권한을 적절하게 드롭하지 않는다. 이 취약점은 공격자들이 쉽게 권한을 상승시킬 수 있다.

4.2 ActiveX

ActiveX 기반의 취약점들은 전보다 2007년도에 더 많이 나왔다. ActiveX 취약점들은 매일 발표되어 왔다. 안티바이러스 소프트웨어도 예외는 아니다.

ActiveX control들은 보통 안티바이러스 제품 설치 동안이나 사람들이 안티바이러스 벤더들의 온라인 무료 스캐닝 서비스를 사용하려 할 때 설치된다. 가끔 ActiveX control은 다운로드 관리자에 의해 설치되기도 한다.

기본적으로 두 가지 종류의 ActiveX 문제가 있다:

* **안전하지 못한 method: design error**
  이것은 안티바이러스 소프트웨어에는 아주 일반적인 문제인데, 왜냐하면 안티바이러스 소프트웨어는 파일 오퍼레이션(생성, 삭제, 그리고 실행)과 네트워크 기반의 오퍼레이션(업로드 및 다운로드)의 기능들을 보통 포함하고 있기 때문이다.

  두 가지 예가 있다.

  A. **CVE-2006-3976** ***CA eTrust AntiVrus WebScan Automatic Update Code Execution Vulnerability***[34]
  이 문제를 확인한 Matthew Murphy에 따르면:

  "WebScan ActiveX 컴포넌트에 대한 자동 업데이트 과정에서 특정 결함이 존재한다. WebScan은 초기화 페이지가 그 컴포넌트가 'SigUpdatePathFTP' 파라미터(그리고 잠재적으로는 'SigUpdatePathHTTP' 파라미터)를 통해 업데이트를 다운로드하여 설치하기 위해 사용할 위치를 지정하도록 허용한다. 그것은 'filelist.txt' 목록을 다운로드하여 그것이 나타내는 목록의 파일들의 업데이트를 요구한다. 파일 목록이나 파일 그 자체에 있는 정보의 신뢰성을 확인하기 위해 WebScan에 의해 수행되는 검증은 없다."

  B. **CVE-2007-1112** ***Kaspersky Antivirus ActiveX Control Unsafe Method Vulnerability***
  이것은 또 다른 하나의 안전하지 않은 method 취약점의 전형적인 경우인데, 이 취약점을 이용해 공격자는 Kaspersky 안티바이러스 소프트웨어가 설치된 컴퓨터로부터 파일을 훔칠 수

---

[34] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3976

있다.

StartUploading()이라는 method가 있다. 이름이 제시하듯 공격자들은 어떤 파일을 훔칠 것인지와 어디로 업로드 할 것인지 지정할 수 있다.[35]

```
Function StartUploading(
    ByVal strFilePath As String,
    ByVal strFTPAddress As String,
    ByVal strFTPUploadPath As String
) As Long
```

*** Memory corruption**

안티바이러스 소프트웨어에서 ActiveX control의 메모리 corruption 문제는 다른 어플리케이션들이 직면한 것과 다르지 않다. 공격자들은 악의적인 입력 값(보통 아주 긴 문자열)을 구성하고, 그것을 취약한 호출이나 method의 파라미터로 전달한다. 스택 기반의 오버플로우, heap 오버플로우, 또는 어떤 다른 메모리 변조 문제를 포함해 메모리 corruption은 그때 발생할 것이다.

안티바이러스 소프트웨어도 역시 이 방면에서는 나쁜 기록을 가지고 있다. 간단히 검색해보아도 Symantec, Authentium, RAV 제품들이 이 문제들에 취약한 것을 알 수 있다.

ActiveX 기반의 취약점을 공격하기 위해 공격자들은 특별히 조작된 HTML 파일을 만들어 웹 사이트에 올리고, 그런 다음 희생자들이 그 사이트를 방문하도록 한다. 취약한 ActiveX control(취약한 소프트웨어)을 컴퓨터에 설치한 희생자가 악의적인 웹 사이트를 방문하게 되면 그의 컴퓨터는 공격자에 의해 장악된다.

4.3 엔진 관련 문제들

엔진은 안티바이러스 소프트웨어의 가장 복잡한 부분이며, 그래서 대부분의 취약점들이 엔진에 존재한다.

하지만, 안티바이러스 엔진 기반의 취약점들은 또한 가장 복잡하고도 흥미로운 부분인데, 이는 다양한 방법으로 공격이 가능하기 때문이다. 그리고 실제로, 다양한 공격을 제한하는 것은 단지 당신의 상상력뿐이다.

기본적으로, 세 종류의 취약점들이 있다.

---

[35] http://www.zerodayinitiative.com/advisories/ZDI-07-014/

*** Memory corruption**

이것은 엔진 문제들 중에서 가장 위험한 문제인데, 보통 이 문제를 이용하여 시스템 전체를 장악할 수 있다.

만약 엔진의 코드가 보안을 염두에 두지 않고 개발되었다면 공격자는 메모리 corruption 상태를 야기하기 위해 주의 깊게 파일들(실행파일, 압축 패키지, 오디오, 문서, 등)을 조작할 수 있는 좋은 기회를 가질 수 있다. 이 파일들을 분석하는 동안 스택 기반의 오버플로우, 힙 오버플로우, 또는 다른 메모리 접근/수정 문제가 발생할 수 있다.

거의 모든 주류 안티바이러스 제품들은 과거에 이 문제를 겪었으며, 이 문제는 그 제품들 중의 많은 것에서 반복적으로 여러 번 발생했다.

Alex Wheeler[36]와 n.runs[37]는 안티바이러스 소프트웨어 엔진의 메모리 corruption 기반의 취약점 연구에 크게 기여했다.

*** Denial of Service**

기본적으로 두 종류의 DoS가 있다.

A. CPU DoS 문제

특별히 조작된 파일은 안티바이러스 엔진이 무한 루프 속으로 들어가게 하고, 그래서 CPU 사용율이 거의 100%에 도달하도록 한다. 악명 높은 'ZIP 폭탄'[38]을 기억하고 있는가?

다음 hex dump(그림.3)는 CHM 파일의 헤더이며, 이 CHM 파일을 NOD32로 스캐닝하는 동안 CPU 사용량이 100%에 머물 것이다.

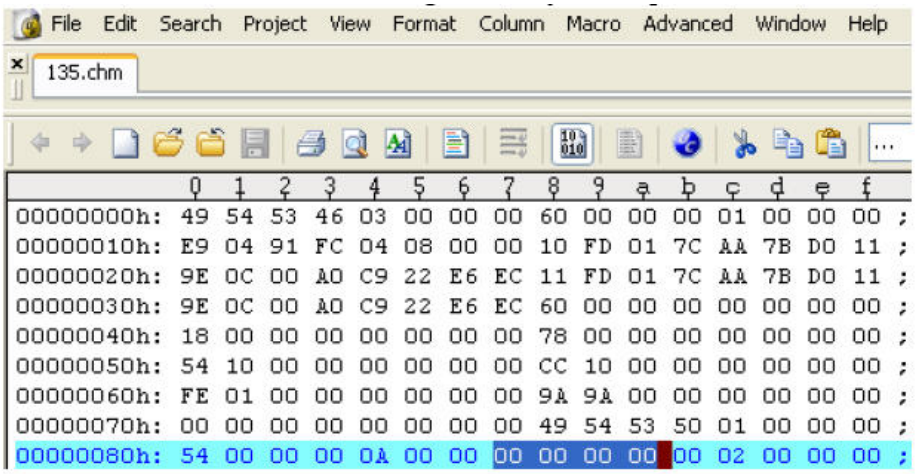


그림3. NOD32용 DoS POC(0day)

---

[36] http://secunia.com/search/?search=Alex+Wheeler+antivirus&w=0

[37] http://www.nruns.com/parsing-engines-advisories.php

[38] http://en.wikipedia.org/wiki/Zip_bomb

B. 디스크 DoS 문제

작은 파일(예를 들어, 1kb보다 작은)을 처리할 동안 안티바이러스 소프트웨어는 4GB의 디스크 공간을 소비할 수 있다. 이 문제는 보통 안티바이러스 엔진이 파일의 압축을 푸는 과정에서 발행한다. 이것은 안티바이러스 엔진이 보통 압축이 풀린 파일들에 대응하는 디스크 공간을 할당하기 위해 파일로부터 읽힌 값에 전적으로 의존하기 때문이다. 그리고 이 값은 아마도 공격자에 의해 조작될 수도 있다.

분명한 예가 다음 ARJ 파일이다. 이 파일을 스캐닝하는 동안 NOD32는 C:₩Documents and Settings₩sowhat₩Local Settings₩Temp₩NOD2.tmp에 4GB 크기의 NOD2.tmp 파일을 만들 것이다.

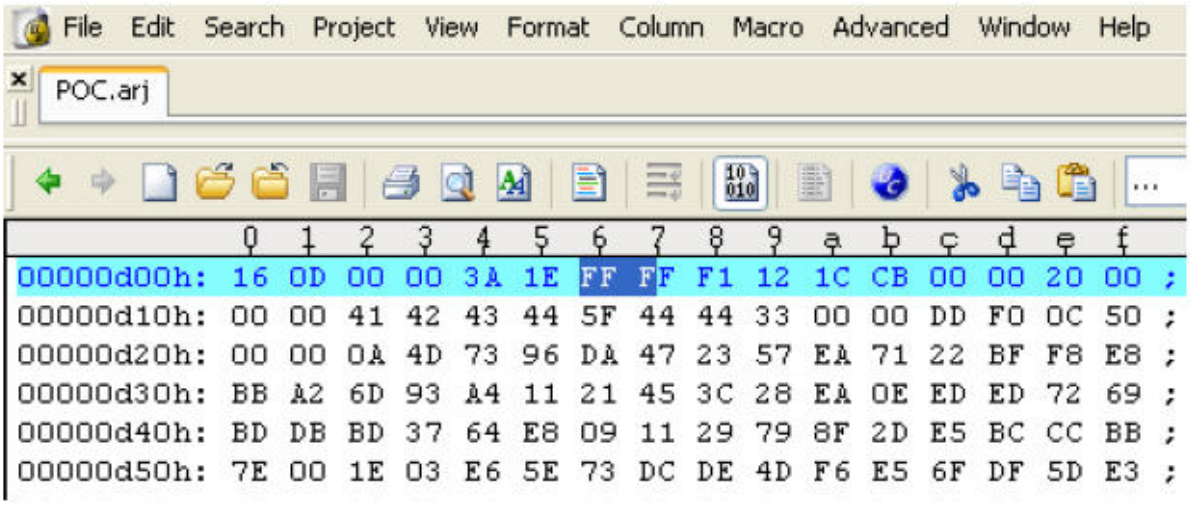


그림4. NOD32용 DoS POC(0day)

CPU와 디스크 기반 DoS의 엄정성은 데스크톱 사용자들에게는 낮은 것으로 간주될 수 있다. 하지만, 같은 상황이 들어오고 나가는 이메일(첨부파일)을 스캐닝하는 안티바이러스 엔진을 가진 메일 서버에 발행한다면 어떻게 될까? 악의적인 이메일은 그 메일 서버를 아주 불안정한 상태 또는 완전히 서비스 불가능한 상태로 남겨둘 것이다.

*** 탐지 우회**

탐지 우회는 상대적으로 낮은 엄정성을 가진 문제이다. 이것은 데스크톱 안티바이러스 소프트웨어보다는 서버 측 안티바이러스 소프트웨어에 더 중요하다.

그와 같은 공격은 보통 다음 상황에서 발생한다.

트로이를 가진 Zip 파일 → 조작 → 메일 서버의 AV → WinZip은 트로이를 성공적으로 추출함

공격자들은 트로이를 가진 zip 파일을 조작하고, 그런 다음 희생자에게 메일의 첨부파일로 보낸다.

조작 후 메일 서버의 안티바이러스 엔진은 이 zip 파일을 분석할 수 없으며, 아마도 전달할 합법적인 파일로 확인할 것이다. 하지만, 희생자가 이메일을 받으며 그 zip 유틸리티(예를 들어, WinZip)는 이 zip 파일을 처리할 수 있을 것이며, 그 트로이를 성공적으로 추출할 것이다.

사회 공학적인 테크닉을 함께 사용해 희생자는 이 트로이를 실행하고, 희생자의 컴퓨터는 장악된다.

엔진 기반의 취약점들은 다음 예에 국한되지는 않지만 다음을 포함한 다양한 방법을 통해 공격될 수 있다.

* 메일 서버

대부분의 메일 서버들은 안티바이러스 스캔 엔진이 나가고 들어오는 이메일들을 스캐닝하도록 설정되어 있기 때문에, 상응하는 안티바이러스 스캔 엔진에 대한 exploit을 가진 공격자들에게는 환상적인 조건을 만든다.

통합 네트워크들은 방화벽, IPS/IDS, 안티바이러스, 그리고 다른 보안 어플리케이션(소프트웨어/하드웨어)로 무장될 수 있다. 비록 공격자에게는 단지 2개의 서비스만이 열려 있는데, 하나는 그들 웹 사이트의 웹 서버이며, 다른 하나는 메일 서버이다. 공격자들은 그 메일 서버로 악의적인 파일을 첨부한 메일을 보냄으로써 LAN으로 침입할 수 있을 것이다.

다음은 가능한 절차이다:

a. Google에서 가능한 이메일 주소를 검색하거나 emailcollect_v1.3.py[39]과 같은 이메일 수집 스크립트를 사용한다.
b. 수집된 주소로 메일을 보낸다.
c. 이메일이 메일 서버에 도착하면 안티바이러스 엔진에 의해 자동으로 스캐닝될 것이다. 그러면 스캐닝 엔진은 취약점 때문에 첨부된 이메일 exploit에 의해 장악될 것이고, 이것은 메일 서버를 완전히 장악하는 것으로 이어진다.

몇몇 메일 서버들은 디폴트로 안티바이러스 스캔 엔진을 설치하는데, 예를 들어, IMail은 Bitdefender 안티바이러스를 설치한다.

Kerio 메일 서버는 버전 6.5.0[40]에서 "*possible buffer overflow in Visnetic anti-virus plug-in*"를 수정했다.

---

[39] http://www.darkc0de.com/misc/emailcollect_v1.3.py

[40] http://www.kerio.com/kms_history.html

이런 종류의 공격의 장점은 다음과 같다.

A. 공격자들은 내부 LAN의 어떤 특정 세부 내용까지 알 필요가 없다.
B. 메일 수신자가 악의적인 메일을 열 필요가 없다. 또한 이메일 클라이언트를 이용해 그 메일을 받을 필요도 없다.

실제 경우:

최근에 보고된 실제 경우[41]가 있는데, 공격자는 안티바이러스 엔진을 이용해 메일 서버를 장악했다.

요약하면, 덴마크 출신의 CISSP인 Faas M. Mathiasen은 그들의 메일 서버로부터 나오는 이상한 패턴을 목격했는데, 많은 양의 데이터가 메일 서버를 통해 그들의 네트워크에 남아 있었다.

그들의 메일 서버는 안티바이러스 소프트웨어가 설치되어 있고, 완전히 패치된 Exchange 2007이 설치되어 있었다. Faas M. Mathiasen은 Exchange 2007에 0day exploit이라도 존재하는지 궁금해했는데, 하지만 다음과 같이 밝혀졌다.

"*스푸핑된 이메일 주소를 이용하는 누군가가 공개적으로 노출된 이메일 주소로 이 파일을 보냈고, 안티바이러스 엔진 스캐너가 그 파일을 건들자마자…. **나는 한 편의 영화를 보는 것 같았다***"

*** 이메일(클라이언트 측)**
만약 공격자들이 취약한 안티바이러스 소프트웨어를 설치한 개인들을 공격 대상으로 삼는다면 이메일이 좋은 선택이 될 수 있다. 다음은 공격의 과정이다:

A. 공격자들은 이메일을 보내고, 그 이메일은 안티바이러스 엔진의 취약점을 이용한다.
B. 만약 안티바이러스 소프트웨어의 자동 보호 가능이 켜져 있다면 희생자의 컴퓨터는 그 이메일이 도착하자마자 장악될 것이다.
C. 비록 자동 보호 기능이 꺼져 있다고 해도 희생자가 첨부 파일을 스캐닝할 가능성이 여전히 남아 있으며, 특히 의심스러운 파일일 때는 특히 그렇다.

이와 같은 공격은 제한된 공격으로 실행될 것이다. 이것은 phishing 공격과 아주 비슷하다.

*** 웹**
웹 상으로 ActiveX 기반의 취약점들을 공격하는 것뿐만 아니라 안티바이러스 엔진 취약점들도 공격하는 것이 가능하다.

---

[41] http://www.securityfocus.com/archive/75/488038/30/0/threaded

웹에서 안티바이러스 엔진의 취약점을 공격하기 위해 IFRAME 태그와 .WMF 파일 확장자가 아주 유용하다.

전형적인 공격 시나리오는 다음과 같다:

a. 공격자는 exploit(예를 들어, exploit.zip이라고 하고, 이것은 안티바이러스의 ZIP 분석 취약점을 이용한다)을 exploit.wmf과 같이 이름을 변경한다.
b. 다음 내용을 포함한 웹 페이지를 만든다.

<iframe src = exploit.wmf>

c. 희생자들이 이 웹 페이지를 방문하게 한다.
d. 희생자가 웹 페이지를 브라우징하는 동안 exploit.wmf는 희생자와의 어떤 상호 대화도 없이 희생자의 컴퓨터로 자동 다운로드된다.
e. 만약 안티바이러스의 자동 보호 기능이 켜져 있다면 안티바이러스 엔진은 자동으로 exploit.wmf를 분석하게 되고, 그런 다음 컴퓨터는 즉시 장악될 것이다.
f. 만약 자동 보호 기능이 꺼져 있다고 하면 공격자들에게는 여전히 더 많은 기회가 남아 있다. Exploit.wmf는 웹 브라우저의 캐쉬 디렉토리에 저장되어 있고, 안티바이러스의 예정된 시스템 스캐닝(또는 직접적인 스캐닝)이 실행되면 안티바이러스 엔진은 공격을 당하게 된다.

*** P2P/IM**

P2P/IM을 통한 안티바이러스 엔진의 취약점을 공격하는 것도 역시 가능하다. IM을 통해 친구에 의해 보내진 파일들은 받자마자 자동으로 스캐닝된다. 그래서 안티바이러스 소프트웨어는 스캐닝 즉시 장악된다.

4.4. 관리

대부분의 안티바이러스 소프트웨어는 관리 목적을 위해서 몇 가지 관리 컴포넌트를 가지고 있다. 이 관리 컴포넌트는 보통 C/S 모드로 작동한다. 서버는 몇 가지 TCP/IP 포트를 리스닝하며 연결을 기다리고, 클라이언트는 서버로 아웃바운드 연결을 한다.

* 클라이언트/서버 관리

클라이언트/서버 관리 컴포넌트는 보통 안티바이러스 벤더에 의해 개발된 프로토콜이라서 그들 자신들만이 이해할 수 있다.

아주 좋은 실제 경우는 CVE-2006-2630[42], *Symantec Antivirus Management Remote Stack Buffer Overflow*이다.

Symantec Antivirus 및 Symantec Client Security용 원격 관리 인터페이스는 보통 작동되어 있으며, 디폴트로 TCP 포트 2967을 리스닝한다. 특별히 조작된 COM_FORWARD_LOG 명령을 보냄으로써 공격자는 전형적인 스택 기반의 버퍼 오버플로우 공격을 시행하여 SYSTEM 권한으로 임의의 코드를 실행할 수 있다.

이 취약점은 나중에 악명 높은 Spybot 웜의 변종에 의해 공격 당했다.(W32.Spybot.ACYR, W32.Spybot.AMTE[43]).

* 라이센스 관리

몇몇 안티바이러스 소프트웨어는 라이센스 관리 목적을 위해 라이센스 관리 컴포넌트를 가지고 있다.

CA 라이센스 소프트웨어는 환상적인 예이다. iDefense Labs에 의해 2005년에 보고된 6개의 취약점들[44]이 있다.

* 웹 인터페이스

웹 인터페이스는 또 다른 하나의 관리 목적의 컴포넌트이다.

이 경우, 취약점들은 구현상의 에러 때문에 발생하는 메모리 corruption 문제이거나 디자인 상의 에러에 의해 발생하는 보안 우회 문제이다.

이 두 가지 경우에 대해 Symantec 스캔 엔진을 예로 들면, 메모리 corruption 문제에 대해서는 CVE-2005-2758[45], 디자인 상의 에러에 대해서는 CVE-2006-0230[46]가 좋은 참고자료가 될 것이다.

CVE-2005-2758 ***Symantec Antivirus Scan Engine administrative interface Integer Overflow***

Symantec Antivirus Scan Engine 4.0 및 4.3의 관리 인터페이스의 integer signedness error는 원격 공격자가 음수의 값을 가진 조작된 HTTP 헤더를 통해 임의의 코드를 실행할 수 있게 하며, heap 기반의 버퍼 오버플로우로 이어진다.

CVE-2006-0230 ***Symantec Scan Engine Authentication Fundamental Design Error***

---

42 http://www.securityfocus.com/bid/18107/references

43 http://www.securityfocus.com/bid/18107/references

44 http://secunia.com/advisories/14438/

45 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2758

46 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-0230

Symantec Scan Engine 5.0.0.24와 아마도 5.1.0.7 이전의 다른 버전들도 패스워드를 확인하기 위해 클라이언트 측 점검을 사용하는데, 이는 원격 공격자가 어떤 XML 요청을 보내는 수정된 클라이언트를 통해 관리자 권한을 획득할 수 있도록 한다.

웹 인터페이스 기반의 취약점을 공격하는 것은 다른 웹 서버(Apache, IIS)의 취약점을 공격하는 것과 같은데, 왜냐하면 그들 대부분이 안티바이러스 벤더 자신들이 개발한 가벼운 웹 서버이기 때문이다.

5. 결론

이 글에서 우리는 안티바이러스 소프트웨어의 취약점을 발견하는 테크닉뿐만 아니라 공격 테크닉도 살펴보았다.

그렇다고 해서 안티바이러스가 쓸모 없다는 것을 의미하지 않는다. 우리는 대체제품을 제안하는 것도 아니다. 우리는 단지 안티바이러스 소프트웨어의 취약점들이 실제적 위협이라는 사실에 주의를 기울이기를 원할 뿐이다.

다음은 안티바이러스 벤더와 최종 사용자에게 하고 싶은 말이다.

벤더에게

안티바이러스 소프트웨어는 입력 파일(스캐닝되는 파일)에 대해 너무 많은 신뢰를 한다. 안티바이러스 소프트웨어는 보안을 염두에 두고 개발되고 점검되어야 한다.

* SDL(Security Development Lifecycle)을 따른다.
* 자신의 제품부터 먼저 점검해라.
* fuzzing은 아주 효과적이다.
  - 릴리즈하기 전에 퍼징을 해라. 제품을 공개하기 전에 퍼징을 통해 취약점을 발견하고 버그를 수정하라.
  - 릴리즈 후에 퍼징을 해라. 퍼징 테크닉은 아주 빨리 발전한다.
* Microsoft, Mozilla 등의 보안 관련 정보를 따른다.
  - Bulletin. 정기 게시판에 보안 취약점을 발표하고, 사용자들에게 가능한 빨리 업데이트 하도록 해라.
  - Credit. 연구자들의 노력과 공로를 인정해라.

최종 사용자들에게

앞에서 언급한 것처럼, 최종 사용자들은 안티바이러스 솔루션에 너무 많은 믿음을 주고 있으며, 안티바이러스 소프트웨어 그 자체도 공격될 수 있다는 사실을 무시해왔다.

사람들은 과거에 의심스러운 것이라면 모든 것(어플리케이션, 아카이브, 문서)을 스캐닝했다. 하지만 지금은 그렇게 하기 전에 좀더 생각해보는 것이 좋을 것이다.

6. 미래 작업

안티바이러스 소프트웨어의 보안은 방화벽, IPS, IDS, 그리고 기타 다른 제품들과 같은 보안 제품에 우리가 주의를 기울이도록 한다. 보안 제품들은 사용자들을 보호하는 것으로 간주되지만, 만약 실패한다면 어떻게 될 것인가? 만약 안티바이러스 소프트웨어가 당신의 시스템에 공격자들을 위한 새로운 문을 연다면 어떻게 할 것인가?

미래의 작업은 '보안' 제품들의 이런 측면에 초점을 맞출 것이다.

7. 감사의 말

필자는 다음 사람들에게 그들의 통찰력과 피드백, 그리고 기술적 의견을 준 것에 대해 감사한다: Chi Zhang, Becky, Sanjay Pendse, Derin Mellor, Khushboo Shah, LinLin Zhao, Neeraj Thakar, Gary Kinghorn, Xin Ouyang, 그리고 Zheng Ren.

8. 참고문헌[47]

[47] 원문의 참고문헌은 이 편역 글의 각 각주에 반영하였다.