特輯

컴퓨터 바이러스 이론과 대응 방안

金 世 憲
韓國科學技術院 經營科學科

Ⅰ. 서론

과거에도 유사한 프로그램이 있었으나 자기 자신을 다른 프로그램에 복제 시키고, 때에 따라서는 컴퓨터 시스템을 공격할 수 있는 [컴퓨터 바이러스(Computer virus)]라는 개념은 1983년 11월 3일 한 실험을 통해 최초로 제시되었다.[1,2] 이 실험은 미국 남가주 대학(University of Southern California)의 대학원생이었던 Fred Cohen등에 의해서 행해졌다. 이들은 UNIX하의 VAX 11/750 시스템에서 8시간 작업끝에 성공적으로 컴퓨터 바이러스를 개발하였고, 즉시 이의 전파과정을 분석하는 작업을 이 Multi-User 시스템에서 수행하였다. 시스템에 대한 가상적 공격팀은 컴퓨터 바이러스의 우발적인 외부 유출을 막기 위해서 시스템을 외부와 철저히 단절하고 완벽한 추적기능을 도입한 후 컴퓨터 바이러스를 시스템 내의 한 프로그램에 심어 놓았다. 5번의 실험이 수행되었는데 컴퓨터 바이러스는 한시간 이내에 운영체계을 감염시켜서, 이를 통해 공격팀은 시스템 오퍼레이터가 갖고 있는 최고 권한의 계정을 알아내어 전체 시스템을 마음대로 통제하는데 성공하였다.

이 연구팀은 처음으로 컴퓨터 바이러스라는 용어를 사용하였으며, 위 실험 내용을 84년 컴퓨터 보안 관련 학회 (IFIP/SEC '84)에 발표하면서 바이러스 프로그램의 출현 가능성과 가공할 위험성을 경고하면서 이에 대한 대응책 수립을 주장하였다. 그러나 대부분의 학회 참석자들은 컴퓨터 바이러스가 갖는 위험성에 대해 안이하게 생각하였고, 다만 극소수의 사람들에게 학문적 흥미의 대상이 되었다.[3] 그러던 것이 80년대 후반에 들어서면서 부터 많은 종류의 바이러스들이 만들어지고 수많은 컴퓨터들이 바이러스에 감염되어 피해를 입게 되었다. 이제 컴퓨터 바이러스는 국민학생에서 부터 가정주부들까지도 알고있는 단어가 되어 버렸다. 컴퓨터 바이러스의 종류에 대해서는 그동안 수 많은 기사와 책자에서 다루어져 왔으므로 이글에서는 생략한다. 이에 대해 궁금한 사항은 서점에서 판매하고 있는 서적들을 참고로하기 바란다.

바이러스에 대한 피해현황에 대해서는 지난 90년 6월 月刊 [경영과 컴퓨터]紙에서 실시한 전국의 765명 PC 사용자들의 컴퓨터 바이러스 관련 실태조사를 인용한다.[4] 이에 의하면 응답자의 89.1%가 컴퓨터 바이러스를 직간접으로 경험했음을 밝히고 있다. 그러나 컴퓨터 바이러스를 알고 있다는 사람들중의 절반 정도는 컴퓨터 바이러스에 대한 잘못된 지식을 갖고 있었다. 또한 컴퓨터 바이러스의 소스코드의 공개가 제한적으로 이루어져야 한다고 응답한 사람들이 많았다. 컴퓨터 바이러스의 제작능력을 갖춘 사람의 26.1%가 경우에 따라서는 컴퓨터 바이러스를 만들 생각이 있다고 응답했다. 컴퓨터 바이러스의 제작자를 처벌해야 한다는 대답은 55.0%로 나타나서 아직은 컴퓨터 바이러스 제작에 관대한 입장임을 보이고 있다.

Ⅱ. 트로이 목마 프로그램과 컴퓨터 바이러스

컴퓨터 바이러스는 이전부터 존재했던 대표적 악성 프로그램인 트로이 목마 프로그램(Trojan Horse Program)의 발전된 형태이다.[5] 모든 컴퓨터는 나름대로의 운영체계(OS)를 갖고 있으며, 이를 통하여 효율적으로 운영되어진다. 트로이 목마 프로그램은

운영체계 내부에 주입되는 일종의 프로그램으로서, 정상적인 운영체계의 변형을 그 목적으로 한다. 트로이 목마 프로그램으로 인하여 변형된 운영체계을 갖게 된 컴퓨터는 특정 작업에 대해서 비 정상적인 결과를 도출하게 되는데, 이 결과는 트로이 목마 프로그램을 만든 사람의 의도에 따라 달리 나타난다.[6]

[그림 1]은 트로이 목마 프로그램이 있는 운영체계을 간단히 나타내고 있다. 여기서는 합법적인 사용자의 Username과 Password를 트로이 목마 프로그램을 통하여 알아내는 내용을 다루고 있다. 즉, 트로이 목마 프로그램에 의해서 가짜 Login 화면이 출력되고, 사용자는 여기에 자신의 Username과 Password를 입력하게된다. 이 때 트로이 목마 프로그램은 입력된 내용을 기록한 후 'Login Faliure !'를 출력하고, 실제 Login 프로그램으로 제어를 넘기게 된다. 결국 사용자는 자신의 Username과 Password가 이미 노출되었는지 모르는 상태에서 다시 정상의 Login 과정을 거치게 된다.

이와 같은 트로이 목마 프로그램의 존재는 운영체계의 복잡성에 기인한다. 운영체계은 복잡한 구조를 가지고 있기 때문에 이를 모두 점검하는 작업은 많은 시간과 인내를 요구하며 또 모든 컴퓨터 사용자들이 복잡한 운영체계의 내용을 반드시 알 필요는 없기 때문에 사용자들은 운영체계 내에 트로이 목마 프로그램이 존재하더라도 그것을 깨닫기 어렵다.

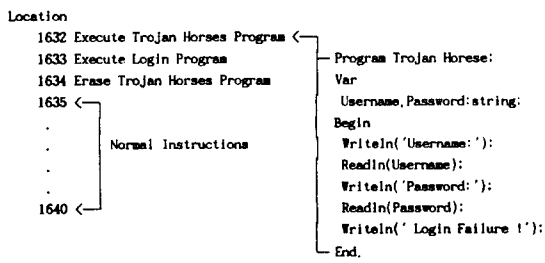


그림 1. 트로이 목마 프로그램

컴퓨터 바이러스는 트로이 목마 프로그램과 같이 정상적인 컴퓨터의 운영체계의 변형을 그 목적으로 하는 프로그램이다. 그러나 컴퓨터 바이러스는, 트로이 목마 프로그램이 특정 컴퓨터를 공격 대상으로 하는 것과는 달리, 동일한 운영체계을 사용하는 모든 정상적인 컴퓨터를 공격 대상으로 한다. 이것은 컴퓨터 바이러스가 갖는 자기복제의 고유특성을 통해서 이루어 진다. 이러한 의미에서 컴퓨터 바이러스는 트로이 목마 프로그램에 복제기능이 추가된 것이라고 할 수 있다. 일단 컴퓨터 바이러스의 기능을 갖게 된 프로그램은 정상적인 컴퓨터의 운영체계을 변형시키게 되는데, 이는 대부분 해당되는 프로그램이 로드될 때 이루어진다.

Ⅲ. 컴퓨터 바이러스의 감염과정

컴퓨터 바이러스의 기능을 갖게 된 프로그램은 다른 정상의 프로그램을 감염시키는데, 이는 각 프로그램을 수행함에 있어서의 의존 관계(Dependency Relation)로 설명할 수 있다. A라는 프로그램을 수행하는데 반드시 B라는 프로그램이 필요하다면, A는 B에 의존(Dependent)한다 라고 하고, 이를 순서쌍들로 모아놓은 것을 의존관계라 한다. 외부로부터 컴퓨터 바이러스가 유입되면, 일단 이것이 의존하는 모든 프로그램들은 감염된다. 또한 새롭게 감염된 프로그램들은 다시 이들이 의존하는 프로그램들을 감염시킨다. 이때 전자의 경우를 1차적 감염, 후자의 경우를 2차적 감염이라 한다. 이와 같은 내용은 다음과 같이 요약 될 수 있다.

A set of programs $P = \{p_1, p_2, \ldots, p_n\}$
A dependency relation $d \subset P \times P$
- 1차 감염 (Primary Infection)
  Infected program p_{n+1}
  A set of programs $P' = P \cup \{p_{n+1}\}$
  A dependency relation $d' \subset P' \times P'$
  A set of primary infected programs
  $I = \{ p_i \mid (p_i, p_{n+1}) \in d', p_i \in P\}$
- 2차 감염 (Secondary Infection)
  A set of secondary infected programs
$I' = \{ p_j \mid (p_i, p_j) \in d', p_i \in I, p_j \in P-I\}$

그림 2. 컴퓨터 바이러스의 감염과정

컴퓨터 바이러스가 그 감염 효과를 극대화하기 위해서는, 다른 프로그램들과 상호 의존성이 높은 프로그램을 1차적으로 감염시키는 것이 효과적일 것이다. 컴퓨터의 운영체계는, 컴퓨터내의 대부분의 프로그램들과 의존관계에 있으므로, 1차적 감염 대상이 된다.

결국 운영체계는 그 자체가 컴퓨터 바이러스의 공격 대상일 뿐 아니라 컴퓨터 바이러스를 널리 확산시키는 매개체가 되는 것이다. W. Gleissner는 컴퓨터 바이러스의 감염과정을 확률적인 모델을 이용하여 분석하였는데 다음의 몇가지 가정을 두고 있다. 첫째 시스템 내에는 m개의 프로그램이 있는데 사용자는 이들 프로그램 각각을 1/m의 확률로 호출한다. 둘째 각각의 프로그램들 시스템내의 다른 프로그램과 의존관계가 확률적으로 주어진다. 셋째 시스템내에는 초기에 1개의 바이러스에 감염된 프로그램이 존재한다. 이러한 가정하에서 시뮬레이션해본 결과 380번의 프로그램 호출만에시스템내에 있는 350개의 모든 프로그램이 바이러스에 감염되었다.

Ⅳ. 컴퓨터 바이러스의 일반적 구조

컴퓨터 바이러스 프로그램은 일반적으로 운영체계을 감염시키는 부분(Infect_OS), 정상적인 화일을 감염시키는 부분(Infect_File) 그리고 특정 조건하에서 컴퓨터 시스템내의 일부를 공격하는 부분(Damage)으로 구성 되어 있다. 컴퓨터 바이러스의 일반적인 구조는 다음과 같은 Pseudo-code 형태로 요약된다.

```
Program Computer_Virus:
  Procedure Infect_OS:
  Begin
   If (OS is not infected) then
      OS를 변형 시킨다:
      바이러스 프로그램을 OS에 복제한다:
  End:
  Procedure Infect_File:
   Begin
    File:=dependent_file(Disk):
    If (File is not infected) then
      바이러스 프로그램을 화일이나 실행 가능한
      디스크의 일부분에 복제한다:
  End:
Procedure Damage
  Begin
   If (Certain conditions are satisfied) then
      디스크나 그 속의 화일을 손상시킨다:
  End:
BEGIN {Main Program}
  Infect_OS:
  Infect_File:
  Damage:
END {Main Program}
```

그림 3. 컴퓨터 바이러스의 일반적인 구조

Ⅴ. 개인용 컴퓨터에서의 컴퓨터 바이러스

이미 언급한 바와 같이 컴퓨터 바이러스는 동일한 운영체계을 사용하는 모든 컴퓨터를 그 공격 대상으로 한다. 예를 들어 MS-DOS상에서 움직이는 컴퓨터 바이러스는 MS-DOS를 사용하는 모든 컴퓨터가 그 공격대상이 된다. 따라서 대형 컴퓨터에서는 컴퓨터 바이러스에 의한 피해는 별로 없으리라 예상된다. 왜냐하면 대형 컴퓨터에서는 각각의 컴퓨터마다 독특한 운영체계을 채용하고 있으며, 때로는 그 내용을 기술적인 비밀로 하는 경우가 많다. 그러나 가장 널리 보급되어있는 개인용 컴퓨터는 대부분이 동일한 운영체계을 사용하고 있으며, 그것의 내용 또한 공개되어있다. 따라서 컴퓨터 바이러스에 의한 피해는 개인용 컴퓨터에서 대규모적으로 발생한다. 이와 같은 이유로 컴퓨터 바이러스에 관련된 논의는 개인용 컴퓨터에 집중되었으며, 이중에서 DOS를 운영체계으로 사용하는 IBM-PC및 그 호환기종에 특히 많은 관심이 모아지고 있는 것이다. 실제로 IBM-PC 바이러스는 그 종류나 피해 사례 측면에서 여타 기종의 컴퓨터 바이러스보다 월등히 앞서고 있다. 지구상에 보급되어 있는 IBM-PC와 그 호환기종의 숫자를 생각해 보면, 하나의 IBM-PC바이러스에 의한 피해가 얼마나 대규모적일 것인가에 대한 상상을 할 수 있다. 따라서 이후에 언급하는 컴퓨터 바이러스는 DOS를 사용하는 IBM-PC 바이러스에 중점을 둘 것이다. 물론 이를 통해서 여타기종의 컴퓨터 바이러스의 분석 또한 용이하게 이루어지리라 판단된다.

이제 IBM-PC 바이러스의 일반적인 구조에 대해서 언급하도록 하겠다. 부트 바이러스나 화일 바이러스는 각각 컴퓨터가 부팅이 될 때, 또는 실행화일의 실

행이 이루어질 때 컴퓨터의 램에 로드 된다. 램에 로드된 컴퓨터 바이러스는 인터럽트 벡터 테이블을 변경하여서 특정 인터럽트 루틴을 차지하게 된다. 또한 이 때 대부분의 IBM-PC 바이러스들은 램상주 형태로 자신의 바이러스 프로그램을 보호한다. 일단 이와 같이 램을 감염시킨 후 컴퓨터 바이러스는 다음의 작용을 한다.

특정 실행화일이 바이러스 자신이 가로채고 있는 인터럽트 루틴을 호출했을 때, 컴퓨터 바이러스는 PC의 특정 부분(부트 섹터, 실행화일, 로드된 실행화일)이 자신의 바이러스 프로그램에 감염되었는지 여부를 체크하고 만약 감염되지 않았다면 새롭게 바이러스 프로그램을 감염 시킨다.

그러나 이 때 특정 조건이 (예, 13일의 금요일, 3월 21일) 만족되면 IBM-PC바이러스는 더 이상의 감염을 중단하고 시스템의 파괴 작용을 한다. 여기서 시스템의 파과작용이란 DOS에서 제공하는 기능을 통해 디스크를 포맷하는등의 소프웨어적인 현상을 의미한다.

다음은 여기서 언급한 IBM-PC 바이러스를 psuedo-code 형태로 정리한 것이다.

```
Program IBM_PC_VIRUS;
  Procedure Infect_System;
   Begin
    If (System is not Infected) then
      Begin
        특정 인터럽트 루틴의 주소를 가로챈다;
        바이러스 프로그램을 해당되는 주소의 램
        에 복제한다;
      End;
  End;
Procedure Infect_File;
  Begin
   File:=dependent_file(Disk);
   If (search_string(File) .NE. 1234567) then
     바이러스 프로그램을 화일이나 실행 가능한
     디스크의 일부분에 복제한다;
  End;
Procedure Damage
  Begin
    If (Certain conditions are satisfied) then
      디스크나 그 속의 화일을 손상시킨다;
  End;
BEGIN{Main Program}
    Infect_System;
    Infect_File;
    Damage;
END{Main Program}
```

그림 4. IBM-PC 바이러스의 일반적 구조

Ⅵ. 컴퓨터 바이러스의 발견과 치유

컴퓨터 바이러스에 관련한 대응방안은 컴퓨터 바이러스의 발견, 치유, 예방등으로 분류할 수 있다.[8,9,10,11,12] 이중 먼저 발견과 치유에 대하여 알아보자.

컴퓨터 바이러스의 발견이란 정상적인 화일과 바이러스에 감염된 화일의 구분을 의미한다. 이와 같은 작용은 컴퓨터 바이러스 프로그램에 이미 내재되어 있다고 할 수 있다.

바이러스 프로그램은 자신의 바이러스 프로그램을 특정 화일에 복제를 할 때, 이 화일이 자신의 바이러스 프로그램에 감염되었는지를 체크하게 되는데 이는 대부분 바이러스 프로그램 내부에 있는 특정 문자열을 체크함으써 이루어지게 된다. 예를 들어 예루살렘 바이러스는 바이러스 프로그램 내부에 "MsDos"라는 문자열을 갖고 있다.

따라서 바이러스 프로그램이 갖는 특정 문자열을 파악하는 것에 초점을 맞추어야 한다. 그러나 누군가가 이 문자열만 변형해서 새로운 변종 바이러스의 제작을 할 수도 있고, 이는 기존의 컴퓨터 바이러스 발견법으로는 발견이 불가능할 수 있다.

컴퓨터 바이러스의 치유는 바이러스 프로그램에 감염된 화일을 정상의 화일로 복구시키는 과정을 의미한다. 그러나 바이러스 프로그램에 감염된 화일의 복구가 완벽하게 이루어지지 않는 경우가 있다.

컴퓨터 바이러스의 발견과 치유는 거의 동시에 이루어 지고 있으며, 이를 위한 상업화된 소프트웨어가 존재한다. 그러나, 이는 이미 존재하는 컴퓨터 바이러스에 대해서만 정상적인 작동을 하기 때문에 신종 바이러스 프로그램에 대해서는 무기력하다고 할 수 있다.

Ⅶ. 컴퓨터 바이러스의 예방

컴퓨터 바이러스의 예방이란 정상적인 컴퓨터를 이미 존재하는 컴퓨터 바이러스 혹은 신종 컴퓨터 바이러스에 감염되는 것을 방지하는 과정을 의미한다. 이하에서는 이와 같은 컴퓨터 바이러스 예방기술에 대해서 서술하고자 한다.

1. Checksum 방식에 의한 컴퓨터 바이러스 예방

컴퓨터 바이러스에 대한 유일하고도 완벽한 예방대책이란, 정상적인 컴퓨터 시스템을 외부로부터 차단시키는 것이다. 그러나 이와 같은 방법은 극단적인 방법이지 현실적이지 못하다.

따라서 여기서는 정상적인 컴퓨터 시스템 내의 프로그램들이 컴퓨터 바이러스에 의해 감염될 가능성을 최소화하는 방법을 생각해 보기로 한다. 이는 컴퓨터 시스템 내의 일부 프로그램에 checksum값을 대응시킨 다음, 컴퓨터 바이러스에 의한 프로그램의 변형을 사후적으로 알 수 있도록 하여, 감염의 가능성을 최소화하겠다는 뜻이다. 여기서는 checksum이 부가되는 프로그램의 선택이 중요한데, 운영체계 내의 여러 유틸리티 프로그램들 중에서 선택하는 것이 바람직할 것이다. 이와 같은 내용은 다음과 같이 요약된다.

```
Set of programs in the system  P ={ p1,p2,...,pn}
Set of seclected programs      P'c P

Step 0.  At time t, generate values V = {v1,v2,...,vn},
         where  vi= C(pi,kt), pi ∈ P' and  kt = f(t),
At time tk:  tk > t
Step 1. Get a program "pi" to be interpreted.
Step 2. If (pi ∈ P') or ( pi ∉ P' and C(pi,kt) = vi)
        then interprete pi, set t=tk and  goto step 0.
        Otherwise goto step 3.
Step 3. Ask user for a move "m"
    m=m1   interprete pi,  set t=tk, goto step 0.
    m=m2   search for computer viruses.
```

그림 5. checksum 방식에 의한 바이러스 예방 프로그램

Step 0는 적절한 시간 t에서 key인 kt를 도출하고 이를 이용하여 선택된 프로그램 pi에 checksum값 vi를 대응시키는 단계를 나타낸다. 일정한 시간이 지난 후 이 checksum값이 변해 있으면 컴퓨터 바이러스의 침입이 있었음을 예측하게 된다.

2. KAIST-ANTI-VIRUS

위에서 제시한 방법을 MS-DOS운영체계를 갖는 IBM-PC에 적용한 경우를 생각해보자.[13] 이미 언급한 바와 같이 IBM-PC 바이러스는 메모리상에 있는 인터럽트 루틴을 그 1차 감염 대상으로 한다. 이는 메모리의 0번 블록(Block)에 존재하는 인터럽트 벡터 테이블의 변경을 통해 이루어진다. 따라서 인터럽트 벡터 테이블에 checksum을 부가하여 그것의 변경됨을 수시로 파악해야 된다. 그러나 이와 같은 검사를 컴퓨터 사용자가 직접해야 한다면 번거로울 뿐만 아니라, 컴퓨터 바이러스의 침입시기를 정확히 감지하기 어렵다. 따라서 이와 같은 checksum 검사를 자동적으로 그리고 수시로 이루어지게 하는 방법이 필요하다.

일반적으로 IBM-PC는 그것의 사용중에 메모리내의 램(램)을 일정시간 간격으로 재충전 한다. 만약 재충전의 과정이 없으면 램에 기록되어 있는 정보를 잃게 되고 컴퓨터의 비정상적인 작동이 이루어진다. 이와 같은 재충전은 초당 대략 18.2 번 발생한다. 그런데 이 재충전 과정에서 인터럽트 8h번과 인터럽트 1Ch번이 반드시 호출된다. 즉, IBM-PC는 정상적인 동작중에도, 사용자가 모르게 그 내부적으로 초당 18.2번의 인터럽트 8h와 1Ch를 수행한다. 만약 인터럽트 8h 혹은 1Ch를 일부 변경하여 위와 같은 checksum을 검사하는 부분을 첨가하게 된다면 우리는 자동적으로 초당 18.2번 정도의 checksum 검사를 할 수 있게 된다. 물론 이는 컴퓨터 사용중에 자동적으로 이루어지는 것이다. 이와 같은 방법은 IBM-PC 바이러스가 유입되더라도 더 이상의 감염을 막아주는 역할을 수행한다. 이러한 방법으로 필자가 전자통신연구소의 지원을 받아 개발한 바이러스 감염 예방 프로그램인 KAIST-ANTI-VIRUS 는 다음과 같이 요약된다.

Program "KAIST-ANTI-VIRUS"

Step.0 롬 바이오스(ROM-BIOS) 데이터 영역으로 부터 사용가능한 메모리 크기를 구하고 부트 바이러스의 존재 여부를 체크한다.(0040:0013h)

Step.1 주요한 롬 바이오스 인터럽트 루틴과 DOS 인터럽트 루틴들의 주소를 저장하고 이들과 관련된 checksum을 구한다.(checksum1)

Step.2 인터럽트 루틴 1Ch의 주소를 변경하여 다

음과 같은 인터럽트 루틴을 가리 키도록 한다.

```
Procedure MY_Handler: Interrupt;
 Begin
    IF ( Clock-Counter = n) Then
      Begin
       주요한 ROM-BIOS 인터럽트 루틴과 DOS인터럽트 루틴들
       의 주소와 관련된checksum을 구한다 (checksum2);
       IF (checksum1 〈〉 checksum2) Then
          Begin
             현재 메모리에 로드 되어 실행중인 프로그램이 등
          록된 프로그램인지 여부를 확인한다 (vector.dat);
             만약 등록되지 않은 프로그램이라면 Step 1에서 저
          장된 인터럽트 루틴의 주소를 인터럽트 벡터 테이
       블에 복사한다;
          End;
      End;
 End;
```

Step 3. Step 2의 My_Handler를 램상주(TSR) 시킨다.

그림 5. KALST-ANTI-VIRUS 의 구조

위에서 step 0는 이미 부팅과정에 감염되었을지도 모르는 부트 바이러스를 체크하는 단계이다. 또한 위의 프로그램은 컴퓨터 바이러스가 아니더라도 인터럽트 벡터 테이블을 변경하는 실행 화일의 리스트를 사용자 임의로 ' vctor.dat '라는 화일에 저장하도록 한다. 이예방 프로그램을 현재 소유하고 있는 IBM-PC 바이러스에 적용하여 보았고, 그 결과 100%의 바이러스 예방능력을 갖고 있음을 알 수 있었다. 여기서 제시한 바이러스 예방 프로그램은 신종 바이러스를 예방하는 능력을 갖추고 있다.

Ⅷ. 맺는 말

컴퓨터 바이러스는 앞으로도 근절되기는 어려울 것으로 보인다. 지속적으로 새로운 바이러스가 제조될 것이며 이에 대응하는 백신프로그램은 새 바이러스를 막기위해 계속적으로 수정보완 되어야 한다. 이 작업은 현재 안철수씨등 몇몇 개인의 희생적인 봉사로 수행되고 있으나 앞으로는 우리의 정보화사회가 이러한 개인의 희생적인 봉사에 기대어 발전해서는 안될 것이다. 이러한 백신 프로그램을 정당한 가격을 치르고 구입하는 의식의 변화가 있거나 아니면 정부의 체계적인 지원이 있어야 할 것으로 생각 된다.

參 考 文 獻

[1] F. Cohen, "Computer Virus: Theory and Experiment", Computers & Security 6(1), pp. 22-35, 1987.

[2] F. Cohen, Computer Virus, Ph.D. Thesis, University of Southern California, 1985.

[3] P. J. Denning, Computers under Attack: Intruders, Worms and Viruses, Addison-Wesley, 1990.

[4] 안철수, "컴퓨터 바이러스 실태조사", 경영과 컴퓨터, 1990년 6월.

[5] 김세헌, 김성륜, "트로이 목마 프로그램을 이용한 컴퓨터 바이러스", 화랑대 국제 학술대회, 1991년 11월.

[6] D. B. Parker, Crime by Computer, Scribners, 1976.

[7] 김세헌, 김성륜, "컴퓨터 바이러스 구조분석과 대응방안", 한국경영과학회 '91 춘계학술대회, 1991년 4월.

[8] F. Cohen, "A Note on High Integrity PC Bootstrapping", Computers & Security 10(6), pp. 535-539, 1991.

[9] F. Cohen, "Models of Practical Defenses against Computer Viruses", Computers & Security 8(2), pp. 140-160, 1988.

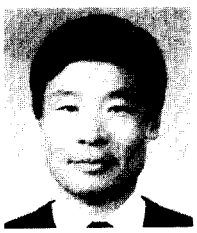
[10] F. Cohen, "Crytographic Checksum for Integrity Protection", Computers & Security 6(3), pp. 505-510, 1987.

[11] S. R. White, "An Overview of Computer Viruses and How to Cope with The m", Computer Security Journal 2(2), pp. 37-56, 1990.

[12] B. P. Zajac, Jr., "Computer Viral Risks

- How Bad is the Threat ?", Computers & Security 11(1), pp. 29-34, 1992.
[13] 김세헌, 김성륜, "컴퓨터 바이러스의 일반적 예방 프로그램 개발에 관한 연 구", 데이타 보호 기반 기술 Workshop, 1992년 8월.

筆 者 紹 介



金 世 憲
1950年 1月 17日生
1972年 2月 서울대학교 물리학과 졸업(학사)
1981年 3月 미 Stanford 대 Operations Research (박사)
1981年 ~ 1982年 미 System control사 선임연구원
1982年 ~ 현재 한국과학기술원 경영과학과 교수

주관심 분야 :통신시스템 최적화, 정보통신 보안