경량화 프로세스 추적을 통한 중요 데이터 유출 방지

Protective Mechanism for Sensitive Data using Lightweight Process Tracking

강구홍
서원대학교 컴퓨터정보통신공학부 정보통신공학과

Koo-Hong Kang(khkang@seowon.ac.kr)

요약

컴퓨터와 개인 휴대 단말장치 사용의 대중화와 함께 이들 장치들을 이용한 개인정보 혹은 사업상 중요 데이터를 처리하고 저장하는 행위도 따라 증가하게 되었다. 따라서 이들 장치로부터 사용자의 허락 없이 중요 데이터가 자신의 장치로부터 빠져나가는 행위가 일어나서는 안 된다. 본 논문에서는 중요 파일과 관련된 파일 오픈 시스템 콜을 호출하는 프로세스를 일정시간 추적하여 네트워킹 인터페이스를 통해 이들 중요 데이터가 빠져 나가는 것을 방지하는 간단한 방법을 제시하였다. 제안된 방법은 기존 인증 혹은 암호화를 이용한 방법과 비교해 해킹 기술 발전과 하드웨어 및 소프트웨어의 새로운 취약점 발견과 무관하게 안정적으로 동작하는 이점이 있다. 특히, 본 논문에서는 리눅스 환경의 사용자와 커널 영역에서 제안된 알고리즘을 직접 구현하여 타당성을 검증하였다.

■ 중심어 : | 중요 데이터 | 유출 방지 | 보안 |

Abstract

As the usage of computers and mobile handsets is popularized, the processing and storing of private and business data are increased. Hence we note that these sensitive data should never be transferred out of these personal devices without user's permission. In this paper, we propose a simple method to prevent transferring the sensitive data out of personal computing devices through their networking interfaces. The proposed method determines which processes invoke open system call related to the sensitive data, and then traces them within a specific duration. The proposed scheme has advantage over the existing ones using authentication or encryption because it could be still working well independent upon the new attack technologies or the latest vulnerabilities of hardware and software. In order to verify the proposed algorithm, we test it by implementing the necessary codes at the user and kernel spaces of Linux.

■ keyword : | Sensitive Data | Steal Protection | Security |

I. 서 론

과거 특정 개인이나 조직이 독점하고 있던 정보는 인터넷과 모바일 등 정보기술의 발전으로 인해 사회 전반에 걸쳐 공유되고 있는 실정이다. 이러한 정보의 다변화는 정보를 저장하고 유통하는 과정에서 정보에 대한 무단 유출 및 파괴, 변조 등과 같은 부작용이 발생하고 있다. 특히, 정보의 무단 유출은 개인의 사생활 침해뿐

접수번호 : #090220-001
접수일자 : 2009년 02월 20일
심사완료일 : 2009년 03월 26일
교신저자 : 강구홍, e-mail : khkang@seowon.ac.kr

만 아니라 경제적 피해가 상당해 정보화 사회의 가장 큰 화두로 자리 잡고 있다. 본 논문에서는 컴퓨터 혹은 모바일 장치로부터 중요 데이터의 유출을 방지하는 알고리즘을 제안한다. 제안된 알고리즘은 제한적인 하드웨어 및 배터리 소모 등의 어려움을 갖고 있는 모바일 장치에 특히 적합한 특징을 갖는다. 따라서 본 논문은 대부분 모바일 장치를 기준으로 설명을 풀어가도록 한다.

사회가 점점 복잡해져 가고 멀티미디어를 포함한 다양한 서비스가 제공됨에 따라서 사람들은 하나의 시스템으로 여러 가지 기능을 제공하고 제공 받을 수 있는 융합기술에 관심을 기울이고 있다. 통신 시장에서도 이러한 추세에 발맞추어 NGN(Next Generation Network)/BcN(Broadband Convergence Network)[9]이 등장하게 되었다. BcN은 통합 네트워크를 통해서 3가지의 융합(패킷과 음성의 융합, 유선과 무선의 융합, 통신과 방송의 융합) 서비스를 사용자의 통합 단말에 제공하는 기술 추세이다. 이러한 사용자의 통합 단말기는 궁극적으로 기존에 가지고 있던 한계들을 뛰어넘어 "Any Time, Any Where, Any Device" 상황에서도 인간이 원하는 서비스를 제공할 수 있는 유비쿼터스 서비스 환경을 형성할 것이다[10]. 따라서 정보단말기는 유비쿼터스 서비스 환경으로 진압하기 위한 인프라의 중심에 위치하고 있다. 이러한 유비쿼터스 사회에서 개개인의 관심사는 기술적인 관점보다 효용과 편의성 증대의 관점에서 얼마나 안전하고 편리하게 융합 서비스들을 즐길 수 있을 것인가에 있다. 따라서 개인들이 사용할 복합단말기의 안전성과 편리함은 유비쿼터스 사회로 진입하기 위한 선행과제임에 틀림없다.

이동전화 단말기인 모바일 폰 역시 이제 더 이상 단순히 사람의 목소리 통신 용도로 사용되는 것이 아니라 상당한 수준의 프로세싱 능력과 데이터 저장 능력을 갖추게 되었으며, 무선랜, 와이브로, 블루투스, WCDMA, HSDPA 기술을 통한 고속의 데이터 통신을 달성하고 있다. 이와 같이 휴대 단말기의 성능이 향상될수록 개인적이고 사업과 관련된 데이터를 처리하고 저장하는 일이 증가하게 되었다. 따라서 이러한 행위와 관련된 많은 데이터는 상당한 가치를 가지고 있으며 해커들의 주요 공격 목표가 되고 있다. 따라서 휴대 단말기 내에 저장되어 있는 중요 데이터가 휴대 단말기의 네트워킹 인터페이스를 통해 외부로 유출되는 것을 검출하는 것은 휴대 단말기를 사용하는 사용자 입장에서는 가장 중요한 이슈임에 틀림없다[1][2].

본 논문에서는 운영체제의 파일 오픈 시스템 콜을 호출하는 프로세스를 일정시간 추적하여 모바일 장치의 네트워킹 인터페이스를 통해 이들 중요 데이터가 빠져나가는 것을 방지하는 새로운 방법을 제시하였다. 제안된 기법은 기존 인증 혹은 암호화를 이용한 방법과 비교해 해킹 기술 발전과 하드웨어 및 소프트웨어의 새로운 취약점 발견에 무관하게 안정적으로 동작하는 이점이 있다. 특히, 본 논문에서는 리눅스 환경의 사용자와 커널 영역에서 제안된 방법을 직접 구현하여 시험하였다.

서론에 이어 제2장에서는 휴대 단말기에서 중요 데이터 보호가 필요한 이유와 기존 유선 네트워크 장비에서 사용하는 보안 솔루션과는 다른 새로운 방법이 필요한 이유와 함께 본 논문의 연구 동기를 기술한다. 제3장에서는 휴대 단말기에서 중요 데이터 유출을 방지하기 위한 알고리즘을 pseudo 코드를 이용해 기술하고, 제4장에서는 제시된 알고리즘을 리눅스 환경에서 구현하고 시험한 결과를 제시한다. 제5장에서 결론과 향후 연구 방향에 대해 기술한다.

II. 연구 동기

1. 문제 제기

모바일 장치들은 암호화, 데이터 백업, 사용자와 어플리케이션 사이의 인증, 그리고 저장된 데이터와 처리되고 있는 데이터의 일치성 등을 이용한 보안 기법들이 구현될 수 있다. 따라서 이론적으로는 이들 보안 기법들이 적절이 운영될 때 모바일 장치들은 항상 안전성을 보장받을 수 있어야 한다. 그러나 잘 알려진 바와 같이 Cabir 웜 혹은 Skulls 트로이 목마와 같이 모바일 폰의 콘텐츠와 기능을 공격하는 악의적인 소프트웨어들이 대량 유포되고 있어 그 피해의 심각성은 앞으로 더욱 가중될 것으로 예상된다[11][12].

인터넷에 연결된 PC 환경에서 악의적인 공격자로부

터 네트워크와 이들 컴퓨터를 보호하기 위해 방화벽, 침입탐지시스템, 그리고 바이러스 월 및 웜 필터등과 같은 각종 보안 솔루션들이 사용되고 있다. 그러나 이들 보안 솔루션들을 직접 모바일 장치에 이용하기에는 여러 가지 어려움이 있다. 이러한 어려움은 모바일 장치의 하드웨어 측면에서 주로 기인한다. 즉 일반 유선 네트워크 환경에서의 컴퓨터와 비교해 프로세싱 능력 부족, 배터리(전원) 소모, 그리고 메모리 부족 등은 기존 유선 네트워크 보안 솔루션의 사용을 사실상 불가능하게 만든다[1][2].

종래의 휴대 단말기에서 중요 데이터 유출을 방지하기 위한 방법은 크게 두 가지 유형으로 분류할 수 있다. 첫 번째 방법은 인증(authentication)을 이용하는 방법이다. 즉 중요 데이터에 접근하기 위해서는 사용자 인증과정을 거치게 되고 실패할 경우 접근 자체가 불가능하도록 한다. 두 번째 방법은 암호화(encryption) 방법을 이용한다. 즉 할당된 키를 사용하지 않고서는 접근 자체를 불가능하게 만든다[1]. 그러나 해커들의 끊임없는 공격 기술 발전과 휴대 단말기의 각종 하드웨어 및 소프트웨어의 새로운 취약점 발견으로 인해 이들 두 가지 방법으로 완벽하게 중요 데이터 유출을 방지하기란 사실상 불가능하다. 따라서 중요 데이터 유출 방지를 위한 새로운 접근이 필요한 시점이다. 특히 앞에서 언급한 휴대 단말기의 제약 조건들을 고려해 추가적인 배터리 소모와 프로세싱 및 메모리 사용을 최소화할 수 있는 경량화 알고리즘이 요구된다.

2. 새로운 접근 방법

기존 유선 네트워크 환경에서 사용되는 침입 방지 시스템은 두 가지 접근 방법, 즉 알려진 공격 패턴을 조사하는 시그너처(signature) 검출 방법과 시스템의 정상 동작 여부를 조사하여 공격에 의한 비정상적인 행위를 찾아내는 어노멀리(anomaly) 검출 방법이 있다[3]. 대부분의 어노멀리 검출 방법은 사용자 행위에 대해 기준이 되는 프로파일을 작성하고 이 프로파일로부터 벗어나는 사용자 행위를 검출하게 되는데, Forrest[3][4]는 주요 프로그램들이 발생하는 시스템 콜 시퀀스를 프로화일로 작성해 데이터베이스화 하고 현재 동작 중인 프로그램에서 발생되는 시스템 콜 시퀀스를 일정 길이 윈도우만큼 해당 프로화일과 비교해 일정 기준 이상 차이가 발생되면 어노멀리가 검출된 것으로 판단하였다. Forrest에 의해 제안된 이러한 접근 방법은 호스트 기반 침입방지 시스템에 적절히 적용되고 있으며, 적용 결과 역시 훌륭한 것으로 보고되고 있다. 그 이유는 공격자가 악의적인 행위를 하기위해 시스템 콜 자체를 위장하기는 사실상 불가능하기 때문이다.

본 논문에서 제안하는 중요 데이터 유출 방지 알고리즘 역시 Forrest가 사용한 시스템 콜 추적 방법과 유사하나 Forrest가 사용한 방법은 휴대 단말기에서 그대로 사용하기는 불가능하다. 즉 모든 프로세스에서 발생되는 시스템 콜을 계속 추적한다는 것은 휴대 단말기의 프로세싱 능력과 배터리 소모를 고려하면 불가능한 작업이다. 뿐만 아니라, Forrest가 사용하는 대용량의 프로화일 데이터베이스를 메모리 량이 상대적으로 작은 휴대 단말기에서 사용한다는 것은 어려움이 있다. 이러한 문제점을 해결하기 위해 본 논문에서는 특정 이벤트가 발생하는 경우에만 프로세스에서 발생하는 시스템 콜을 추적하기 시작하고 일정 시간이 경과하면 추적 작업을 종료한다. 또한 프로화일 데이터베이스를 최소하기 위해 필요한 시스템 콜들로 구성된 상태머신을 정의하고 이들 상태머신을 추적하여 중요 데이터 유출을 방지하도록 한다.

III. 중요 데이터 유출 방지 알고리즘

본 논문에서 제안하는 중요 데이터 유출 방지를 위한 기본 알고리즘을 pseudo code로 나타내면 [그림 1]과 같다. [그림 1]에서 보듯이 기본 알고리즘은 두 개 모듈로 구성된다. 즉 추적할 프로세스를 결정하는 check_systemcall 모듈과 프로세스에서 발생하는 시스템 콜을 실제적으로 추적해 중요 데이터 유출을 검출해내는 pro_trace 모듈로 구성된다.

먼저, check_systemcall 모듈의 동작 과정은 다음과 같다. 임의의 프로세스가 중요 데이터 sensitive.txt 파일에 접근하기 위해 오픈 시스템 콜을 호출하면, 이 프

로세스가 현재 추적 중인 프로세스인지 확인하기 위해 추적 프로세스 리스트를 조사하고 또한 화이트 리스트에 등록된 프로세스인지 조사하고, 만약 현재 추적 중인 프로세스 그리고 화이트 리스트에 등록된 프로세스가 아니면 이 프로세스를 추적하기 위해 "pro_trace" 프로그램을 시동하여 추적을 시작한다. 여기서, 추적 프로세스 리스트는 현재 추적 중인 프로세스 번호를 유지하고 있으며, 화이트 리스트는 추적하지 않을 프로세스 번호를 유지하고 있다. 예를 들어, 사용자의 요청에 의해 특정 프로세스가 중요 데이터 파일을 자유롭게 처리할 수 있도록 하기 위해 화이트 리스트에 등록하면 된다. 또한 현재 추적 중인 프로세스 개수가 일정 수준 이상이면 경고신호를 발생하게 되며, 경고신호에 대한 처리는 다양하게 이루어질 수 있다. 가장 강력하게는 해당 프로세스를 'kill' 하는 것이며, 가장 단순하게는 사용자에게 알람을 알리는 수준이 될 것이다. 추적 중인 프로세스가 일정 기준치 이상 지나치게 많다는 것은 분명 malware 혹은 외부로 부터의 공격을 의심할 수 있다.

```
    void check_systemcall(pid_t pid)
                    // pid is a process ID which calls a
system call
    {
      while(1){
        do(system call event is occurred){
          if((system call is sys_open)
&&(file_name=sensitive.txt)){
          if((pid != entry of tracing process list) &&
            (pid != entry of white list)){
            if(no. of entry in the tracing process list
> LIMIT)
                  make alert;
            else{
                insert pid into tracing process list;
                invoke a process "pro_trace(pid);
            }
                } // end of inner if-clause
              } // end of outer if-clause
          } // end of do-clause
      } // end of while loop
    }
    void pro_trace(pid_t pid)
                    // pid is a process ID which calls
sys_open
    {
      int againg_no = 100; //set aging number 100
system calls
      int is_socket_called = -1;
      while((system call event is occurred) &&
(againg_no != 0)){
          if(system call is exit) break;
          if(system call is socketcall)
is_socket_called = 0;
          else if((system call is write) &&
(is_socket_called == 1)){
              make alert;
              break;
          else
              aging_no--;
          }
      } // end of while loop
      remove pid from the tracing process list;
      terminate this process;
    }
```

그림 1. Pseudo 코드로 나타낸 알고리즘

[그림 1]의 "pro_trace" 는 해당 프로세스의 시스템 콜을 일정 시간 ([그림 1]에서는 100개 시스템 콜) 추적하고 종료하며, 휴대 단말기의 네트워킹 인터페이스를 통해 중요 데이터를 전송하기 위해 "소켓" 관련 시스템 콜과 "쓰기" 관련 시스템 콜이 차례대로 발생되면 경고를 발생하도록 한다. 그러나 보다 상세한 추적을 할 경우, [그림 2]와 같이 시스템 콜 상태머신을 보다 구체적으로 설계한 후 추적할 수 있으며 추적 시간 역시 확장할 수 있다. 즉 보다 구체적인 일련의 시스템 콜 시퀀스

를 각 상태로 정의하고 이들 상태를 추적함으로서 false positive를 줄일 수 있을 것이다.

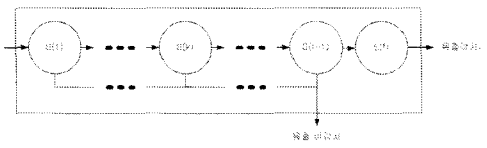

그림 2. 시스템 콜 상태 머신

VI. 구현 및 실험

1. 구현

본 논문에서는 리눅스 fedora 4 (리눅스 커널 2.6.11-1.1369_FC4)[13]가 인스톨된 데스크-톱 PC에서 3장에서 설명한 알고리즘을 [그림 3]과 같이 구현하였다. 본 논문의 성격상 휴대 단말기에서 직접 구현하는 것이 프로세서 부하 및 메모리 사용정도, 그리고 배터리 수명 등 성능 분석을 위해 가장 이상적이나, 본 논문에서는 제시된 알고리즘의 타당성 검증 수준에서 구현하였음을 미리 밝힌다.

구현은 [그림 3]에서 보듯이 커널 공간에서 동작할 두 개의 커널 모듈을 제작하고 사용자 공간에서 동작할 두 개의 프로그램을 제작하였다. 사용자 영역에서 데몬 프로세스([그림 3]의 9번 표시)는 커널 영역의 문자 디바이스 드라이버([그림 3]의 6번 표시)에 'read'로 접근하여 추적할 프로세스가 발생할 때까지 잠들게 된다. 이 때 문자 디바이스 드라이버는 데몬 프로세스의 접근을 관리하기 위해 대기 큐(Wait Queue)[7]를 사용하고 interruptible_sleep_on() 함수를 사용하여 데몬 프로세스를 재운다. 한편 사용자 영역에서 임의의 프로세스([그림 3]의 1번 표시)가 오픈 시스템 콜을 호출([그림 3]의 2번 표시)하는 것을 시스템 호출 래퍼(wrapper [5]) sys_my_open 모듈([그림 3]의 3번 표시)이 후킹하고 오픈할 파일이 'A.txt'이면 대기 큐에서 자고 있는 데몬 프로세스를 wake_up_interruptible() 함수를 이용해 깨우고 추적할 프로세스 번호를 디바이스 드라이버로 넘김으로서 데몬 프로세스가 디바이스 드라이버로부터 추적할 프로세스 번호를 읽어간다.

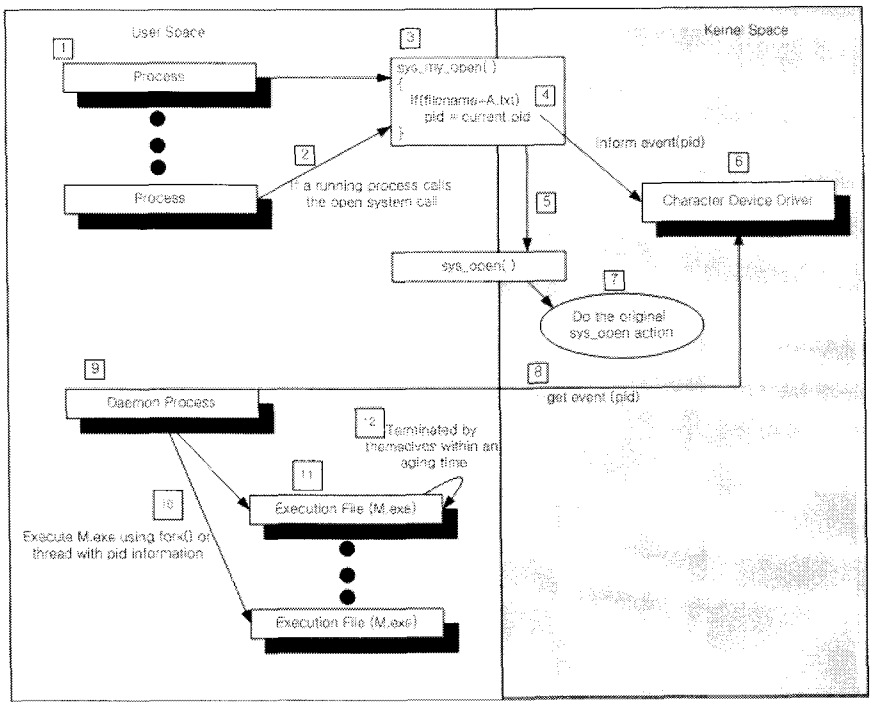

그림 3. 알고리즘 구현 블록도

디바이스 드라이버에서 선언한 대기 큐를 래퍼 모듈이 사용할 수 있도록 대기 큐를 EXPORT_SYMBOL 매크로[8]를 사용하여 공개하고 래퍼에서는 extern을 이용해 대기 큐를 사용하게 된다. 데몬 프로세스는 추적할 프로세스 번호가 발생되면 fork() 함수를 이용한 멀티태스킹 혹은 스레드를 기반으로 한 멀티스레딩 방식([그림 3]의 10번 표시)으로 프로세스 추적 프로그램([그림 3]의 11번 표시)을 실행한다. 추적 프로그램은 ptrace 시스템 콜[6]을 사용해 제작되었으며 일정 시간 수행된 후 자동 종료되도록 설계되었다. 한편, 래퍼는 사용자 영역에서 임의의 프로세스([그림 3]의 1번 표시)에게 정상적인 sys_open([그림 3]의 5번과 7번 표시)을 제공하도록 설계되었다.

2. 실험

본 논문에서 제안한 중요 데이터 유출방지 기법은 외부로부터의 공격 자체를 검출하는 것은 아니다. 즉 해커의 공격을 받거나 혹은 임의의 불법적인 방법으로 컴퓨터 혹은 휴대 단말기에 저장되어 있는 주요 정보가 네트워크를 통해 유출되는 행위를 방지하기 위함이다. 물론 위기 상황을 그대로 재현할 수 있는 공격용 툴을 제작해서 시험하는 것이 바람직하나 이러한 종류의 공격용 툴은 대부분 루트 권한의 프로세스를 획득하고 이후 자신의 다양한 행위를 구현하게 되는데[14], 이러한 행위 중 하나가 네트워크를 통해 중요 데이터를 유출하는 행위가 될 것이다. 궁극적으로 본 논문에서 제안한 기법의 타당성을 검증하기 위한 실험이 공격 자체를 검출하는 것이 아니라 특정 정보의 유출을 방지하는 것임으로 본 절에서는 단순히 ftp를 이용한 데이터 유출에 따른 검출 결과만을 확인하였다.

[그림 4]의 결과 화면에서 보듯이, 데몬 프로세스는 블록되어 있다가 디바이스 드라이버로부터 추적할 프로세스 번호 (본 시험에서는 5946)를 넘겨받고 추적 프로그램을 실행한다. 그림에서와 같이 socketcall과 write 시스템 콜이 일어남을 확인하고 경보를 발행하게 된다. 3장에서 설명한 바와 같이 이 경보에 대한 처리는 가장 강력하게는 해당 프로세스를 'kill' 하는 것이며, 가장 단순하게는 사용자에게 알람을 알리는 수준이 될 것이다.

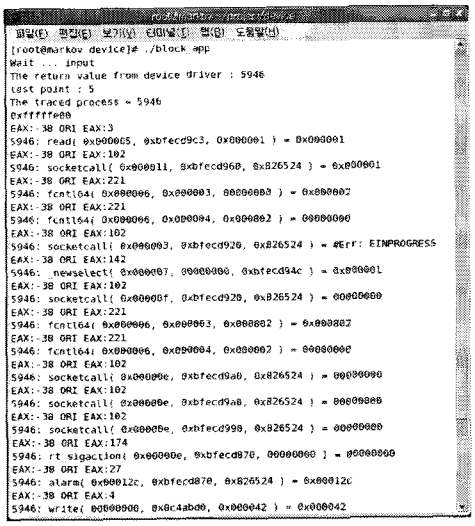

그림 4. ftp를 이용한 실험 결과 화면

V. 결론 및 향후 연구방향

기존의 호스트기반 침입방지 시스템은 상당한 기술 수준에 올라 와 있으나 뛰어난 프로세싱 능력과 풍부한 하드웨어 사양을 요구한다. 그러나 휴대 단말기는 일반 유선 네트워크 환경에서의 컴퓨터와 비교해 프로세싱 능력 부족, 배터리(전원) 소모, 그리고 메모리 부족 등의 문제로 인해 기존 유선 네트워크 보안 솔루션을 그대로 사용하는 것은 사실상 불가능하다. 기하급수적으로 늘어나는 휴대 단말기의 사용 추세에도 불구하고 휴대 단말기에 대한 보안 연구는 이제 막 시작 단계에 불과하다. 특히, 사용자 입장에서는 휴대 단말기 내에 저장되어 있는 중요 데이터가 외부로 유출되는 것을 검출하는 것은 가장 중요한 이슈임에 틀림없다.

본 논문에서는 운영체제의 파일 오픈 시스템 콜을 호출하는 프로세스를 일정시간 추적하여 모바일 장치의 네트워킹 인터페이스를 통해 이들 중요 데이터가 빠져나가는 것을 방지하는 새로운 알고리즘을 제시하였다. 제안된 알고리즘은 휴대 단말기의 특성을 반영하여 경량화 구조를 채택하고 가능한 간단하게 구현될 수 있도록 하였다. 한편 리눅스 fedora 4 PC 환경에서 제안된

알고리즘을 직접 구현하고 ftp를 통해 중요 데이터가 유출되는 것은 검출해 내는 것을 확인함으로서 제시된 알고리즘의 타당성을 검증하였다. 본 논문에서 제시된 알고리즘은 시스템 콜을 직접 추적하여 검출하는 방식을 채택함에 따라 새로운 하드웨어 혹은 소프트웨어 취약점 발견 혹은 새로운 해킹 기술의 발전에 따른 보안 솔루션의 패치 부담이 상당 부분 감소할 것으로 예상된다.

4장에서 언급한 바와 같이 본 논문에서 구현된 시험 환경은 휴대 단말기가 아닌 일반 PC 환경에서 이루어졌다. 따라서 제시된 알고리즘의 동작에 대한 타당성 검증 수준에 불과하다. 알고리즘 구현에 따른 휴대 단말기에서의 CPU 및 메모리의 사용 부하와 추가적인 배터리 소모량 등과 같은 성능 분석은 현재 진행 중에 있으며 추후 발표할 예정이다.

참 고 문 헌

[1] M. Miettinen, P. Halonen, and K. Hätönen, "Host-Based Intrusion Detection for Advanced Mobile Devices," Proc. AINA'06, pp.72-76, 2006.

[2] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges," ACM Transactions on Embedded Computing Systems, Vol.3, No.3, pp.461-491, 2004(8).

[3] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes," Proc. of the 1996 IEEE Symposium on Security and Privacy," pp.120-128, 1996.

[4] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," Proc of the 1999 IEEE Symposium on Security and Privacy, pp.133-145, 1999(5).

[5] 조유근, 최종무, 홍지만, *리눅스 매니아를 위한 커널 프로그래밍*, 교학사, 2001.

[6] W. Mark and B. Dan, *Self-Service Linux Mastering the Art of Problem Determination*, Prentice-Hall, 2005.

[7] 유영창, *리눅스 디바이스 드라이버*, 한빛미디어, 2004.

[8] 한동훈, *리눅스 커널 프로그래밍*, 한빛미디어, 2007.

[9] 장승원, 조일권, 최영준, 박석천, "BcN 서비스 동향 및 발전 방향", 정보처리학회지, 제13권 제4호, pp.62-72, 2006.

[10] 설원희, "차세대 무선인터넷 기술 동향 및 발전 방향", 정보처리학회지, 제12권, 제1호, pp.25-27, 2005.

[11] J. Jamaluddin, "Mobile phone vulnerabilities: a new genration of malware," Proc. of the 2004 IEEE International Symposium on Consumer Electronics, pp.199-202, 2004.

[12] S. Ravi, A. Raghunathan, and S. Charkradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems," Proc. of the 17th International Conference on VLSI Design, pp.605-611, 2004.

[13] http://fedoraproject.org/, Fedora User Guide

[14] A. Purohit, V. Navda, and T. Chiueh, "Tracing the root of "rootable" processes," Proc. Computer Security Applications Conference, pp.284-293, 2004.

저 자 소 개

강 구 홍 (Koo-Hong Kang) 정회원

- 1985년 8월 : 경북대학교 전자공학과(공학사)
- 1990년 2월 : 충남대학교 전자공학과(공학석사)
- 1998년 2월 : 포항공과대학교 전자계산학과(공학박사)
- 2000년 9월 ~ 현재 : 서원대학교 정보통신공학과 부교수

<관심분야> : 컴퓨터 네트워크, 네트워크 프로그래밍