情報保護學會論文誌
第 21 卷 第 1 號, 2011. 2

모바일 단말에서 외부 저장 매체로의 불법 데이터 유출 방지 기법*

정 보 흥,† 김 정 녀‡
한국전자통신연구원

Prohibiting internal data leakage to mass storage device in mobile device*

Bo-heung Chung,† Jung-nyu Kim‡
Electronics and Telecommunications Research Institute

요 약

최근 들어, 모바일 단말의 폭발적인 보급 더불어 단말 내의 중요정보가 외부 저장 매체로 불법적으로 유출되는 보안 위협이 증가되고 있다. 따라서, 본 논문에서는 단말 내 중요정보의 외부 저장 매체로의 유출 방지 기법을 설계 및 구현한다. 이 기법은 파일의 임의위치에서 시그니처를 샘플링하고 이를 이용하여 유출 탐지, 차단 기능을 수행한다. 시그니처 샘플링 과정은 대상 파일을 일정 크기의 추출 윈도우로 구분한 후 이 영역 내에서 임의의 위치에서 1개 이상의 시그니처를 추출한다. 그리고, 가장 효과적인 샘플링을 수행하기 위하여 전체 샘플링, 이항분포 샘플링, 동적 샘플링의 다양한 추출 방식을 구현 및 시뮬레이션을 수행 한다. 제안된 기법은 파일의 임의 위치에서 시그니처를 샘플링하여 공격자의 시그니처 예측성을 낮출 수 있고 원본 데이터에 대한 변형 없이 유출 방지기능을 효과적으로 구현할 수 있다는 장점을 가진다. 따라서, 사용자 편의성이 중시되고 비교적 저 사양의 시스템인 모바일 단말에서 효과적으로 유출방지 기능을 구현할 수 있는 기법이다.

ABSTRACT

According to proliferation of mobile devices, security threats have been continuously increased such as illegal or unintentional file transmission of important data to an external mass-storage device. Therefore, we propose a protection method to prohibit an illegal outflow to this device and implement this method. This method extracts signatures from random locations of important file and uses them to detect and block illegal file transmission. To get signatures, a target file is divided by extracting window size and more than one signatures are extracted in this area. To effective signature sampling, various extraction ways such as full, binomial distribution-based and dynamic sampling are implemented and evaluated. The proposed method has some advantages. The one is that an attacker cannot easily predict the signature and its extraction location. The other is that it doesn't need to modify original data to protect it. With the help of these advantages, we can say that this method can increase efficiency of easy-to-use and it is a proper way leakage prevention in a mobile device.

Keywords: Data leakage protection, Mobile Security, Signature Sampling

접수일(2010년 11월 23일), 게재확정일(2010년 1월 27일)
* 본 연구는 지식경제부 및 정보통신연구진흥원의 산업원천기술개발사업의 일환으로 수행하였음.[10035708, "고신뢰 자율제어 SW를 위한 CPS 핵심기술 개발"].
† 주저자, bhjung@etri.re.kr
‡ 교신저자, jnkim@etri.re.kr

I. 서 론

최근 들어 MP3, PMP, 스마트 패드, 스마트폰 등과 같은 모바일 단말이 일상생활에서 매우 빠른 속도로 보급 되고 있고, 생활의 필수품으로 여겨지고 있다. 또한, 사용자들의 다양한 요구를 충족시키기 위해 E-mail, 오피스 프로그램, 금융거래, 일정관리, 게임 등과 같은 다양한 서비스를 단말에서 제공하고 있으며 더 많은 서비스에 대한 필요성이 증가되고 있다. 그리고, 사용자는 모바일 단말 내부 저장 공간에 주소록, 인증서, 개인적 자료 등의 중요한 정보가 보관하게 된다. 이를 위해서 모바일 단말은 이러한 다양한 서비스를 수용하고 내부 메모리의 부족과 사용자 편의성 제공을 위하여 USB 메모리, SD/CMD, CF 카드 등의 외부 저장 매체(mass storage device)와 WiFi, 블루투스(Bluetooth)와 같은 네트워크 인터페이스를 사용할 수 있도록 허용하고 있다. 이러한 환경에서는 일반적인 PC환경과 유사한 바이러스, 웜 등의 악성프로그램, 사용자 부주의 또는 단말의 보안성 부족 등의 원인으로 단말 내의 중요한 정보가 외부 저장 매체로 사용자가 모르는 사이 불법적으로 유출되는 위협이 존재하게 된다.

이러한 위협으로부터 사용자 데이터 보호를 위한 기존 연구로는 데이터 암호화 기법, 워터마킹 또는 스테가노그라피와 같은 정보은닉 기법, MD5 또는 CRC와 같은 해쉬 값을 이용한 무결성 검사 기법 등이 있다[2, 4, 5, 9, 12, 13]. 암호화 기법과 정보은닉 기법은 암호화와 숨겨진 식별자(Hidden Mark)를 삽입하기 위해서 원본 데이터에 대한 변경이 필수적으로 필요하게 된다. 이러한 변경을 위해서는 시스템에 부가적인 부하가 발생하고 이들 데이터에 접근하는 서비스 프로그램에 대한 변경이 필요하게 된다. 즉, 암호화되고 정보 은닉된 데이터를 접근하기 위한 전용 뷰어 프로그램이 필요하게 된다는 의미이다. 또한, 무결성 검사 기법은 대상 파일의 변경 여부는 해쉬 값 비교를 통하여 쉽게 알 수 있으나 전체 데이터를 모두 가진 상태에서 이를 스캔하여 계산하여야 하기 때문에 데이터가 모두 전송된 후가 아니면 데이터 유출을 차단할 수 없다.

본 논문에서는 원본 데이터에 대한 변경 없이 적용할 수 있으며 일부 전송 데이터만 가지고도 중요 정보의 외부 저장 매체로의 불법 유출을 효과적으로 차단하기 위한 방법을 제안한다. 이 기법은 파일의 임의위치에서 샘플링 방식을 사용하여 시그니처를 추출하고 이를 이용하여 유출 탐지, 차단 기능을 수행한다. 여기서 임의위치 샘플링은 사용될 시그니처를 파일의 임의의 위치에서 일정 길이의 바이트 문자열을 추출하는 것을 의미한다. 시그니처 추출을 위해서는 일정한 크기의 추출 윈도우 크기로 대상 파일을 구분하고 이 영역들 내에서 최소 1개 이상의 시그니처를 추출한다. 이 과정에서 시그니처 중복성 검사를 수행하여 중복이 발생한 경우에는 다른 임의위치에서 시그니처를 추출한다. 또한 제안되는 기법에서는 이렇게 구분된 각각의 추출 윈도우에서의 시그니처의 추출여부에 따라서 전체 샘플링, 이항분포 샘플링, 동적 샘플링의 다양한 추출 방식을 제시한다. 유출 탐지 과정에서는 외부 저장매체로 복사되는 데이터 블록과 시그니처에 대한 패턴매칭을 수행하여 매칭되는 경우 이를 차단한다. 또한, USB 디바이스 드라이버 및 파일 시스템 필터 드라이버에 이 기능을 구현하여 차단되기 전에 전송된 데이터 블록도 자동으로 삭제되도록 하였다. 제안하는 기법은 임의위치 샘플링을 통해 유출 탐지, 차단 기능을 수행하기 때문에 원본 데이터에 대한 변경이 필요 없고 이 데이터를 접근하기 위한 전용 프로그램이 추가적으로 필요하지 않게 된다. 다양한 시그니처 추출 방법을 통하여 시그니처에 대한 예측을 어렵게 하고 시그니처 추출 위치의 일정 영역으로의 편중성을 줄이게 된다. 또한, 제시된 각각의 기법에 대한 시뮬레이션을 수행하여 가장 효과적인 추출 방식을 제시하고 시그니처 추출 및 탐지부하에 대한 시뮬레이션도 수행하여 최적의 추출 윈도우 크기를 제시한다.

논문의 구성은 다음과 같다. 2장에서 정보은닉 또는 암호화를 통한 데이터 보호 방법에 대해서 알아보고 3장에서는 제안기법의 시그니처 추출을 위한 고려사항에 대하여 설명한다. 4장에서는 제안된 기법의 모바일 단말에 구현에 대해 설명하고 5장에서는 시그니처 추출을 위해 제시된 각각의 샘플링 방법에 대한 시뮬레이션에 대해서 설명한다. 마지막으로 결론을 기술한다.

II. 관련연구

다양한 서비스와 기능을 가진 모바일 단말의 폭발적인 성장과 더불어서 보안에 대한 중요성도 증가하고 있다[8, 9]. PDA, 스마트 폰, 태블릿 등과 같은 모바일 단말 들은 기존의 PC와 같은 컴퓨팅 장비에 비해서 컴퓨팅 능력, 메모리 등과 같은 요소에 있어서 한계를 가지게 된다[2, 3].

모바일 단말 분야에서의 불법적인 정보의 외부 유출을 차단하기 위한 보안기술에 대한 다양한 형태의 연구가 진행되어 왔다[7, 9, 10, 11]. 첫 번째로는 DRM(Digital Right Management)[15], 스테가노그라피(steganography), 워터마킹(watermarking), 핑커프린팅(fingerprinting)[9, 12, 13] 등과 같이 보호할 정보에 암호화를 수행하거나 은닉 표시자(hidden marker)를 보호할 정보에 삽입하는 방법이 있다. 이러한 방법들은 기본적으로 데이터를 보호하기 위한 암/복호화 비용 또는 은닉 표시자를 원본 데이터에 사용자가 알아차릴 수 없게 삽입하고 검출하기 위한 추가적인 비용이 발생하게 된다. 또한, 위 과정을 통해서 원본 데이터에 대한 변형이 이루어지게 되며, 변형된 데이터에 대한 접근을 위해서는 추가적인 응용 프로그램이 필요하다. 예를 들면, DRM 기법이 적용된 문서파일을 읽기 위해서는 DRM을 제거한 후 메모장으로 문서를 읽거나 DRM이 적용된 문서를 바로 읽을 수 있는 별도의 메모장 프로그램을 개발하여야 한다. 일반적으로 이러한 작업을 고비용의 수학적 연산과 높은 컴퓨팅 능력을 필요로 하기 때문에 ASIC, FPGA와 같은 하드웨어 형태로 제작되는 것이 일반적이고, 소프트웨어적으로 모바일 단말에 적용하는 경우에는 단말에 상당한 부하를 초래하게 된다.

다른 방법으로는 원본 데이터에 대한 변형 없이 유출 차단 기능을 제공하는 방법이며, 시그니처(signature) 또는 해쉬 값(hash value)을 이용한다[2, 4, 5]. 예를 들면, CRC 또는 MD5와 같은 해쉬 값을 사용하며, 이 값 들은 선택된 영역 또는 전체 메시지/파일을 스캔한 후 생성되는 바이트 값들 이다. 검출 과정에서는 유출이 시도되는 선택 영역 또는 전체 데이터에 대한 바이트 값들을 모니터링 한 후 해쉬 값을 생성하여 이 값과 보관된 해쉬 값이 일치하는지를 검사하여 동일한 경우 탐지/차단하는 방법이다. 시그니처를 이용하는 방법은 원본 데이터의 일부분에서 바이트 값들을 추출하여 보관하고 이 값을 이용하여 위 검사를 수행하는 방법이다. 예를 들면, 바이러스 백신 프로그램들은 바이러스를 분석하여 바이러스를 유일하게 식별할 수 있는 바이트 값들을 추출한다. 이후에 이 값들을 파일 또는 메모리의 값과 비교하여 바이러스를 검출하는 방법을 사용한다. 이 방식은 원본 데이터에 대한 변형이 필요 없으며 추가적인 비용이 적게 발생한다는 장점이 있지만 유일하게 식별할 수 있는 바이트 값들을 추출하는 것이 상당히 어렵다는 단점을 가진다.

III. 모바일 단말 데이터 유출 방지 기법

이 장에서는 모바일 단말 데이터 유출 방지를 위한 시그니처 샘플링에 대해 설명한다. 이를 위해, 먼저 유출방지를 위해 샘플링을 적용하는 이유에 대해 설명하고, 다음으로 효과적인 샘플링을 위해 본 논문에서 사용하는 구체적 샘플링 방식에 대해 설명한다.

3.1 샘플링을 통한 탐지 시그니처 추출

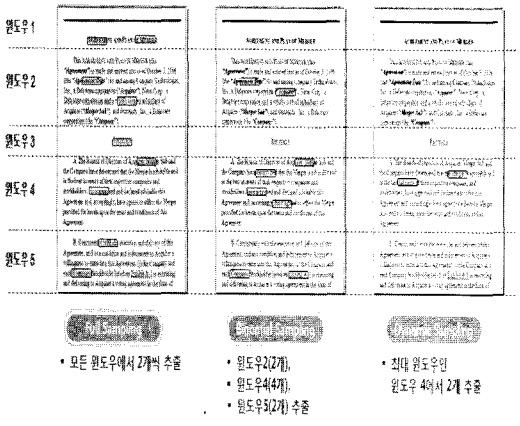
본 논문에서 제안하는 기법은 기본적으로는 시그니처 방식에 근간을 두고 있다. 그러나, 시그니처 방식을 그대로 사용하는 데는 몇 가지 어려움이 존재한다. 예를 들면, 바이러스 백신 프로그램의 예에서 보듯이 시그니처 생성을 위해서는 바이러스에 대한 선행지식 및 분석과정이 필요하다는 점이다. 다른 말로 하면, 유출 방지를 위해서는 보호할 데이터가 포함된 파일 등을 정확하게 규정해야 한다는 것이다. 이는 사용자가 파일의 내용을 일일이 수작업으로 분석하여 보호할 데이터가 위치한 파일의 특정 역역을 표시하여야 한다는 것을 의미한다. 이 과정을 효과적이면서도 사용자 편의성을 높여주기 위해서는 내용인식(content recognition)과정을 제공해야 한다. 즉, 사용자는 중요정보에 대한 키워드 정도만 제시하고 이 정보를 바탕으로 파일을 스캔하여 이 키워드가 포함된 파일들을 자동으로 검출하면 된다. 그러나, 이러한 방법에도 어려움이 존재한다. 일반적으로 내용인식 과정은 텍스트 형태의 파일을 대상으로 하고 있으며 많은 연구가 진행되어 왔지만 효과적인 검출을 위해서는 문제점들이 존재한다. 예를 들면, "비밀문서"라는 키워드가 포함된 파일을 검출하고자 하는 경우, "비.밀.문.서", "비-밀$문^서" 등과 같이 악의적인 목적으로 원본파일에 대한 변경을 수행하게 되면 검출하지 못하는 한계를 가지게 된다. 또한, 계좌번호와 같은 개인정보를 그림파일의 형태로 보관하고 있는 경우, 계좌번호를 키워드로 지정해 놓은 경우라고 하더라도 이를 검출하기 어렵다는 것이다. 따라서, 본 논문에서는 사용자 개입을 최소화하고 파일에 대한 선행지식 또는 분석과정을 최소화하기 위해 시그니처를 샘플링 방식을 사용하여 추출한다.

3.2 샘플선정을 위한 고려사항

샘플링 방식에 기반한 유출 방지 기법이 효과적으

로 동작되기 위해서는 먼저 추출된 샘플의 위치를 쉽게 예측할 수 없어야 하며 추출된 시그니처의 효율적인 관리를 위해서 추출되는 샘플의 크기는 가능한 한 최소로 유지되어야 한다. 추출한 시그니처에 대한 예측을 어렵게 하기 위해서는 기본적으로 샘플링은 랜덤 위치에서 이루어져야 하며 대상 영역 또는 파일 전체의 다수 위치에서 추출되어야 한다. 이를 위해, 제안하는 기법에서는 전체파일을 다수의 윈도우로 구분하고 이 윈도우내에서 랜덤 샘플링을 수행한다. 즉, 전체 윈도우들 중에서 샘플링 대상 윈도우들을 선정하고 이 윈도우에서 다수의 시그니처를 추출한다. 이들 각 영역을 샘플링 윈도우라고 하고 이 윈도우의 크기는 최소값으로는 시그니처의 크기에서부터 최대값으로 전체파일의 크기 사이로 결정된다. 따라서, 어떤 윈도우를 선정할 것인지와 그 윈도우에서 몇 개의 시그니처를 추출할 것인지가 핵심적인 요소이다. 그러므로, 본 논문에서는 가장 효과적인 샘플 선정을 위해서 전체 샘플링(FS:Full Sampling), 이항분포 샘플링(BS:Binomial Sampling), 동적 샘플링(DS:Dynamic Sampling) 세부적인 샘플 선정과정을 수행하고 이들 각 방법 들을 비교함으로써 가장 효과적인 방법을 제안한다.

시그니처는 추출 윈도우 내에 완전히 포함되도록 추출되어야 한다. 즉, 임의로 선정된 추출 윈도우내의 추출 위치 값에서 시그니처 크기만큼 추출을 할 수 없게 되면 정상적인 결과를 기대할 수 없기 때문이다. 예를 들면, 추출 윈도우는 100 바이트이고 시그니처 크기가 24 바이트라고 할 때, 추출 위치가 랜덤함수를 통하여 91 바이트로 나왔다면 이 위치에서부터 시그니처를 추출하게 되면 24 바이트가 아니라 10 바이트가 되기 때문이다. 또한, 단편화로 인하 탐지 불능 상황을 방지하기 위해서 시그니처는 파일 전송 블록 경계영역에 걸치게 추출되지 않도록 하여야 한다. 예를 들면, 대상 파일이 총 10개 전송 블록으로 구분되고 파일 전송 블록 단위가 8K 바이트이고 시그니처 크기가 24바이트라고 한다면 $1 < n < 10$ 일때 $8196 \times n - 24 <$ 추출 위치 $< 8196 \times (n+1) + 24$ 범위 내에서는 시그니처를 추출하지 않도록 하여야 한다. 만일, 이 영역에서 시그니처를 추출한 경우에는 시그니처와 비교할 대상이 단편화 되어 있기 때문에 오탐이 발생할 수 있게 되기 때문이다.



〔그림 1〕 전체샘플링, 이항분포 샘플링, 동적 샘플링 방식의 시그니처 샘플링 방법

3.2.1 전체 샘플링(FS)

전체 샘플링 방식은 가장 단순하고 적용하기 쉬운 샘플링 방식으로서 모든 윈도우에서 시그니처 샘플링을 수행하는 방법이다. 예를 들면, 〔그림 1〕에서 보는 바와 같이 각각의 윈도우에서 2개의 시그니처를 추출하게 되며 이들 시그니처들은 각각의 윈도우 내에서 임의의 위치에서 추출된다. 또한, 동일한 위치에서 추출하지 않도록 추출된 시그니처들간의 중복성 검사를 수행한다.

3.2.2 이항분포 샘플링(BS)

이항분포 샘플링은 전체 샘플링과는 다르게 모든 윈도우가 아닌 일부 윈도우에서만 시그니처를 샘플링하는 방식이다. 확률이론과 통계 분야에서 이항분포는 연속된 n번의 독립시행 결과의 확률적 분포를 예측할 수 있도록 도와준다. 따라서, 본 논문에서는 파일 내에서의 중요정보 또는 보호할 데이터에 대한 위치 분포를 예측하기 위해서 이항분포의 개념을 도입하였다. 즉, 중요정보가 확률적으로 파일 내에서 이항분포에 따라 분포되어 있다고 가정하고 이에 따라 시그니처를 샘플링 하는 방식이다.

$$P_{(k\ out\ of\ n)} = \frac{n!}{k!(n-k)!}(p^k)(q^{n-k}) \quad (1)$$

이항분포는 위 식(1)〔14〕과 같이 표시 할 수 있으며 n은 총 윈도우 개수, k는 현재 윈도우, p는 사건이 발생할 확률, q는 1 - p 값이다. 여기서 p값은 중요정보가 가장 많이 분포할 가능성이 높은 위치에 대한 확

률 값을 의미한다. 예를 들어, 일반적으로 p 값이 0.5라고 한다면 파일의 중앙부근에서 중요정보가 위치할 확률이 높다는 의미이다. 그러나, 본 기법에서는 대상 파일의 외부 유출을 탐지할 샘플 문자열을 임의의 위치에서 추출하기 위해서 p값을 사용하기 때문에 고정 값을 사용하지 않는다. 이는 p 값이 고정되게 되면 항상 같은 위치에서 샘플링이 이루어지게 될 가능성이 높고, 이는 유출방지 탐지과정을 회피하기 위한 시그니쳐에 대한 예측성을 높일 수 있기 때문이다.

3.2.3 동적 샘플링(DS)

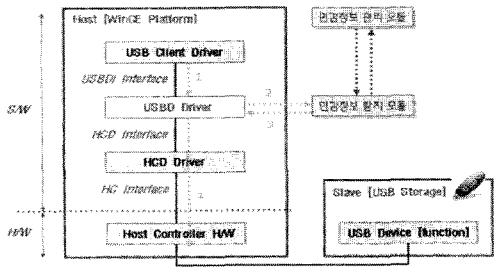
동적 샘플링은 모든 과정이 이항분포 샘플링과 동일하고 시그니처를 샘플링 할 윈도우의 선정과정과 추출할 시그니처의 개수 결정 방식만 상이하다. 간단히 말하자면, 시그니처를 추출하기 위해 선정된 윈도우들 중에서 가장 많은 시그니처를 추출할 윈도우에서만 샘플링을 수행하는 방법이다. 예를 들면, 이항분포에서 선정되는 2, 4, 5 윈도우가 선정되었고, 각각의 윈도우에서 추출할 시그니처가 2, 4, 2이라고 한다면, 4번 윈도우에서만 시그니처를 추출한다. 또한, 추출할 시그니처의 개수도 이항분포 샘플링 보다는 적은 수로 샘플링을 수행한다. 이는 시그니처 추출과정의 부하와 추출된 시그니처들에 대한 관리 부담을 줄이기 위함이다.

IV. 모바일 단말 데이터 유출방지기법 구현

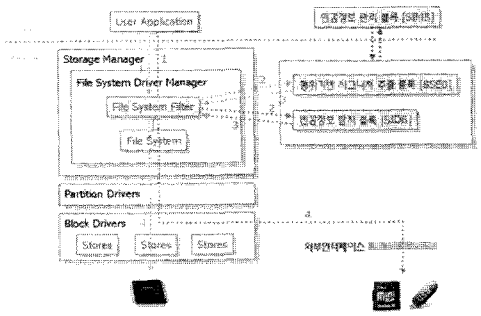
이 장에서는 모바일 단말 데이터 유출 방지 기법의 구현에 대하여 설명한다. 이를 위해, 먼저 유출방지 기법이 운용될 시스템 구조에 대해 설명하고, 각각의 샘플링 방법의 구체적인 구현 방식에 대해 설명한다.

4.1 단말의 유출방지 기능 적용 위치

모바일 단말에서 외부 저장장치로의 데이터 유출 방지 기능을 구현하기 위해서는 실제 파일 복사 등의 연산이 수행되는 시점에 구현되어져야 한다. 이 기능들은 기본적으로 운영체제와 밀접하게 연관되어 있기 때문에 운영체제를 수정하여 구현하는 접근방식이 가장 좋다. 그러나, 확장성과 가용성 측면에서 이 방식은 비효율적이고 상용의 운영체제가 소스를 공개하고 있지 않기 때문에, 제안하는 기법은 운영체제에서 지원하는 시스템 레벨의 접근 포인트에 적용하는 방식을



[그림 2] 디바이스 드라이버 형태의 구현방식



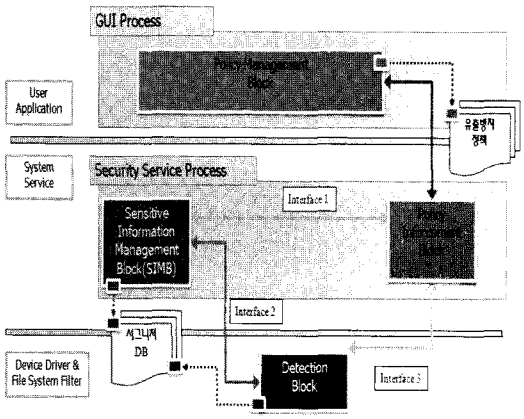
[그림 3] 파일시스템 필터 형태의 구현방식

사용하였다.

즉, 디바이스 드라이버, 파일 시스템 필터 등의 운영체제가 지원하는 접근 포인트에 제안 기법을 추가 구현하여 적용하였으며 [그림 3]에서 보는 바와 같다. 디바이스 드라이버에 적용하는 방법은 [그림 3]에서 좌측그림에서 보는 바와 같이 USB Client Driver와 HCD Driver 사이에서 동작하는 USBD Driver에 민감정보 탐지 모듈의 기능을 추가하는 방식으로 구현하였다. 또한, 파일 시스템 필터에 적용하는 방법은 윈도우 운영체제의 Storage Manager내의 File System Driver Manager와 연계하여 동작할 수 있도록 구현하였다.

4.2 유출방지 기능 구현을 위한 시스템 구조

제안된 기법은 [그림 4]와 같이 사용자 응용 프로그램 수준, 시스템 서비스 수준, 디바이스 드라이버 또는 파일시스템 필터 수준 등과 같이 크게 3개의 부분으로 구성되며 [그림 4]와 같은 구조를 가진다. 응용 프로그램 수준에서는 데이터 유출 방지를 위한 보안 정책의 지정, 변경 등의 관리 기능과 유출 차단 경보 출력 및 관리 등과 같은 사용자 인터페이스적인 측면을 담당한다. 시스템 서비스 수준에서는 시그니처



〔그림 4〕 단말 데이터 유출 방지를 위한 시스템 구조

추출, 관리 등의 작업을 수행하는 SIMB 블록과 사용자 GUI로부터 보안정책을 전달받아 SIMB 전달하거나 하위의 탐지 블록에서 전달된 유출 경보 및 차단 경보를 사용자 GUI로 전달하는 정책 적용 블록(PIB:Policy Enforcement Block)이 있다. 디바이스 드라이버 또는 파일 시스템 필터 수준에서는 탐지 블록이 추출된 시그니처를 바탕으로 외부 메모리로의 파일 전송시 데이터 유출 여부를 판단하여 통과 또는 차단 기능을 수행한다.

4.3 시그니처 추출 구현

본 논문에서 제안된 전체 샘플링, 이항분포 샘플링, 동적 샘플링은 모두 기본적으로는 시그니처 추출과정에서 〔표 1〕과 같은 알고리즘을 사용한다. 이 알고리즘은 전체 윈도우에 대해서 설정된 샘플링 방식에 따른 추출 시그니처의 개수를 계산한 후에 해당 윈도우에서 중복을 피하여 임의의 위치에서 시그니처를 추출하는 알고리즘이다. 여기서 이항분포 샘플링을 통한 샘플링 개수 계산 과정은 앞에서 설명한 식(1)을 사용하여 계산한다.

알고리즘은 크게 초기화 단계, 샘플선정 단계, 추출 단계로 나뉘어 진다. 초기화 단계는 라인 002에서 004까지에서 보는 것 처럼 샘플링 알고리즘에서 사용할 자료구조에 대한 초기화를 수행한다. 샘플 선정 단계는 라인 005에서 018까지에서 보는 것처럼 기본적인 자료구조를 초기화한 후에 추출할 샘플링 방식과 샘플 개수를 계산한다. 이 과정을 좀 더 자세히 설명하면, 대상 파일에 대한 전체 추출 윈도우 n개에 대해서 각 윈도우에 대한 추출 시그니처의 개수를 계산하

〔표 1〕 시그니처 추출 알고리즘

```
ALGORITHM SampleSignature
INPUT
    - SamplingMethod:
      . TYPE_FS:전체 샘플링
      . TYPE_BS:이항분포 샘플링
      . TYPE_DS: 동적샘플링
    - n : 전체 샘플링 방식에서 추출 윈도우에서 추출할 샘플 개수
OUTPUT
    - ExtractedSignatures :
      추출된 시그니처를 보관하는 자료구조
=============================================================
001: {
002:    샘플링을 위한 자료구조 초기화;
003:    sc = 0;          // 동적 샘플링 개수
004:   index = -1;   // 동적 샘플링을 수행한 윈도우 인덱스
005:    for( I = 0; I < 윈도우개수; I++ ) {
006:       n = 0;
          // 추출할 샘플 개수 선정후 n에 설정;
007:       if(SampleMethod == TYPE_FS)
        {   // 전체 샘플링
008:          n = 전체 샘플링 개수;
009:       }
010:     else if((SampleMethod==TYPE_BS)
           || (SampleMethod==TYPE_DS) )
         {
011:        n = 이항분포 샘플링을 통한 샘플링 개수 계산;
012:        if(SampleMethod==TYPE_DS )
         {
013:           if( nc > n ) {
014:              sc = n;
015:              index = i;
016:           }
017:        }
018:     }

019:     if( n > 0 ) {
020:       do {
021:          collision = 0;
022:          locations = 샘플 추출 위치 램덤 함수 수행;
023:          bad_position = location 값들에 대한 유효성 검사;
024:          if( bad_position != 1 ) {
025:          I번째 윈도우에서 n개의 샘플을 임의 위치에서 추출;
026:          추출된 샘플이 이전 샘플과 중복되면 collision 을 1로 설정;
027:          }
028:       } while( collision != 1 );
029:    }
030:  }
      // 동적 샘플링 수행
031:   index 윈도우에서 sc 개수의 시그니처를 임의 위치에서 추출;
032: }
```

여 이 개수가 1개 이상인 경우에만 추출 단계를 수행한다. 추출 단계는 라인 019에서 031까지에서 보는 것처럼 선정된 방식에 따라 임의위치에서 샘플링을 수행한다. 동적 샘플링은 이항분포 샘플링은 최대 시그니처 개수를 가지는 추출 윈도우에서만 시그니처를 샘플링 하는 방법이기 때문에 라인 031에서 보는 것처럼 라인 013에서 016까지 과정을 통해 설정된 index 값에 해당하는 추출 윈도우에서 샘플링을 수행한다. 이상의 과정에서 추출된 시그니처는 ExtractedSignature에 저장 되기 때문에 025라인에서 샘플을 추출한 후 ExtractedSignature에 보관된 값과 비교하여 충돌이 발생한 경우에는 다시 라인 019로 돌아가 시그니처 추출과정을 계속한다.

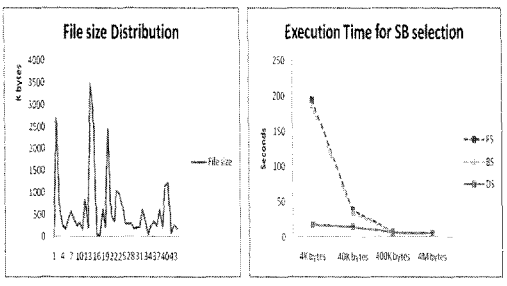
V. 추출 알고리즘 시뮬레이션

각각의 샘플링 알고리즘에 대한 결과와 효과를 검증하기 위해서 다음과 같은 환경에서 시뮬레이션을 수행하였다. 먼저, 샘플링을 수행할 중요정보가 포함된 파일은 4K byte에서 3.5M bytes 사이의 43개 파일을 선정하였다. 시뮬레이션에 사용한 시스템은 3GB RAM을 가진 2.67 GHz Intel PC이다. 또한, 파일 복사 시 시스템의 기본 전송 블록 크기는 일반적으로 블록 크기로 사용되는 8192(8K) bytes로 가정하였다. 본 성능 평가는 제안된 각각의 알고리즘의 시그니처 추출 부하를 비교하는 것이 주 목적이므로 시뮬레이션의 편의와 효율성을 위해 일반 PC시스템을 사용하였다. 시뮬레이션 방식은 윈도우 크기를 4K에서 4M까지 변화시키면서 각각의 방식의 추출시간을 비교하는 방식을 사용하였다. 또한, 제안된 방법을 적용하지 않고 파일복사를 한 경우와 적용된 후의 파일복사 시간을 비교하였다. 이 과정에서 최대 시그니처 비교를 수행하는 최악의 상황을 가정하기 위해서 추출된 시그니처를 포함하고 있지 않은 파일을 대상으로 시험을 하였다.

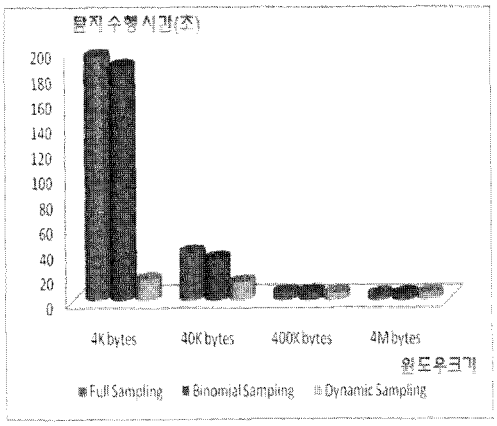
[그림 5]에서 왼쪽 그래프는 시뮬레이션에 사용된 파일들의 크기 분포를 표시한 것이고, 오른쪽 그래프는 이들 파일에 대한 시그니처 추출 시간을 보인 것이다. 사용자의 중요정보가 포함된 파일들은 다양한 크기 분포를 가질 수 있으므로 이를 최대한 반영하기 위해서 4K byte에서 3.5M bytes 사이의 파일들을 대상으로 하였고 이에 따라 윈도우 사이즈를 변화시키며 시험하여 최적의 윈도우 사이즈를 선정하기 위함이다. 시그니처 추출 시간은 시그니처 개수에 비례하게 되기 때문에 이러한 부하를 줄이기 위해서는 윈도우를 최대한 크게 선정하는 것이 좋다. 그러나, 한 파일에서 추출되는 시그니처의 개수가 줄어들게 되면 시그니처에 대한 예측성도 높아질 수 있기 때문에 이에 대한 적절한 조정이 필요하다. [그림 5]의 오른쪽 그래프는 추출 윈도우 크기를 변화시켜가면서 각 추출 방식에 따른 추출시간을 보인 것인데 최대 파일 사이즈에 가깝게 윈도우 사이즈를 선정한 경우에는 각각의 방식이 비교적 유사한 시간을 보이는데 비해 40K bytes 보다 작아지는 경우에는 급작스럽게 시간이 증가하는 것을 볼 수 있다.

[그림 6]은 각각의 샘플링을 적용한 경우 유출이 발생하는지 탐지하는 수행시간을 표시한 것이고, [그림 7]은 추출 윈도우 사이즈가 400K bytes인 경우 동적 샘플링 방법을 적용한 경우 탐지를 수행하지 않고 파일을 복사한 경우와 탐지를 수행한 경우 사이의 수행시간 비교를 수행한 것이다.

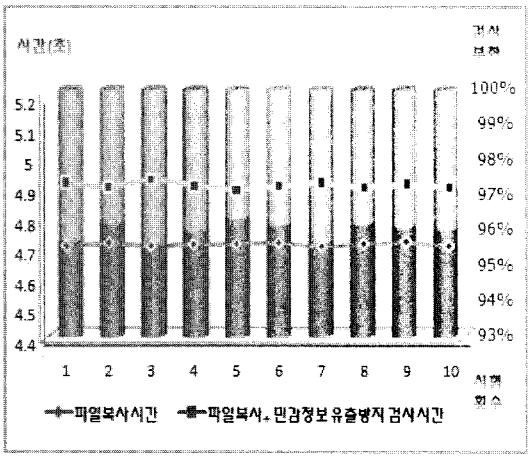
[그림 6]에서 보는 바와 같이 작은 추출 윈도우를 선택하는 경우 전체 샘플링과 이항분포 샘플링 방식의



[그림 5] 파일크기 분포 비교 및 시그니처 샘플링 부하 비교



[그림 6] 유출방지 수행 시간



〔그림 7〕 유출방지 기능 탐지부하

경우 추출된 시그니처 비교에 많은 시간이 소요되어서 시스템에 상당한 부하를 초래하게 된다. 이에 비해 동적 샘플링은 추출 윈도우 크기에 상관없이 거의 일정한 시간이 소요됨을 볼 수 있다. 따라서, 각각의 추출 방식을 효과적으로 적용하기 위해서는 추출 윈도우 사이즈를 평균 파일크기의 약 7% 정도에 해당하는 크기로 선택하는 것이 가장 효율적이다. 또한, 〔그림 7〕은 유출 방지 검사시간을 포함한 파일 복사시간을 100%로 보고, 이 검사를 수행하지 않은 경우의 수행시간을 비율로 표시한 것이다. 또한 각각의 과정을 수행하기 위한 시간을 초단위로 표시하였다. 그림에서 보듯이 10번의 시도에서 검사시간을 포함한 경우는 평균적으로 약 5초의 시간이 걸리고 파일 복사는 평균적으로 약 4.7초가 걸린 것을 알 수 있다. 즉, 탐지 부하가 평균적으로 4.2% 이하임을 알 수 있다.

VI. 결 론

본 논문에서는 단말 내의 중요정보가 불법적으로 외부 저장 매체로 유출되는 경우를 효과적으로 탐지, 차단할 수 있는 모바일 단말 중요정보 유출 방지 기법을 제안하였다. 이 기법은 유출 방지를 위해 파일의 임의의 위치에서 탐지 시 사용할 시그니처 바이트 문자열을 추출하고 추후 단말에서 외부로의 파일 복사 시도 시 이 시그니처들과 복사되는 파일을 비교하여 데이터의 유출 여부를 판별하는 방법이다. 이 과정에서 가장 중요한 요소는 시그니처의 추출 방식이므로 전체 샘플링, 이항분포 샘플링, 동적 샘플링 등의 다양한 시그니처 추출 방법을 제시하였고 이들 중 최적의 추출 방법을 찾기 위해 시뮬레이션을 수행하였다. 그리고, 제안된 방법의 실제 운용을 위해서 윈도우 모바일 운영체제에서 USB 디바이스 드라이버, 파일 시스템 필터 드라이버를 수정하여 적용하였다.

제안된 기법은 시뮬레이션을 통하여 추출 윈도우의 크기가 작아질수록 시그니처 추출, 유출 탐지 비용이 증가하게 됨을 알 수 있었으며 특히 전체 샘플링과 이항분포 샘플링의 경우는 추출 윈도우의 크기가 특정 값 이하로 작아지면 부하가 기하급수적으로 커지게 됨을 알 수 있었다. 따라서, 추출 윈도우의 크기는 중요정보 파일로 지정한 대상 파일들의 평균 파일크기의 약 7%정도로 선정하는 것이 가장 적합하며, 이러한 최적의 추출 윈도우로 설정한 경우 탐지 부하가 전체 파일 복사시간의 4.2% 이하로 적게 유지될 수 있었다. 이와 같이 제안된 기법은 모바일 단말에서 단말 내 중요한 정보가 외부 저장 매체로 불법 유출되는 경우를 효과적으로 차단할 수 있는 방법이고, 대상 파일의 크기를 고려하여 추출 윈도우를 설정하여 운영하면 최소한의 부가 비용으로 효과적인 유출 차단을 수행할 수 있는 방법이다. 향후 연구로는 시그니처의 개수를 최소화하면서도 오탐율을 최소화 할 수 있도록 추출 알고리즘의 개선하는 연구와 단말 내 시그니처 저장소에 대한 불법적인 접근을 통제할 수 있는 기법에 대한 연구가 필요하다.

참고문헌

〔1〕 Smith, T.F., Waterman, M.s., Identification of Common Molecular Subsequences, J. Mol. Biol. 147, pp. 195-197, July 1981.

〔2〕 Shi, Z., Ji, Z., Hu, M., A Novel Distributed Intrusion Detection Model Based on Mobile Agent, ACM InfoSecu04, pp. 155-159, November 2006.

〔3〕 Young-guang, Z., Wenke, L., Yi-an, H., Intrusion Detection Technique for Mobile Wireless Networks, ACM MONET, pp. 545-556, November. 2004.

〔4〕 Deepak, V., An Efficient Signature Representation and Matching Method for Mobile Devices, Proceedings of the 2nd annual international workshop on Wireless internet, Vol 220, August 2006.

[5] Geetha, R., Delbert, H., A P2P Intrusion Detection System based on Mobile Agents, ACM ACME'04, pp. 185-195, April 2004.
[6] National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov
[7] Narayanan Shivakuma, Hector Garcia-Molina, "Building a scalable and accurate copy detection mechanism", DL'96 Proceedings of the first ACM international conference on Digital libraries, pp. 160-168, March 1996.
[8] Yogesh Prem, S., Hannes, T., Protecting Mobile Devices from TCP Flooding Attacks, ACM mobiarch'06, pp. 63-68, December 2006.
[9] Benjamin, H., Mobile Device Security, ACM InfoSecCD Conference'04, pp. 99-101, September 2004.
[10] Ingemar, J. Ton, K., Georg, P., Information Transmission and Steganography, IWDW 2005, LNCS 3710, pp. 15-29, 2005.
[11] David, C., Sebastian, H., Pasquale, M., Quantitative Analysis of the Leakage of Confidential Data, Electronic Notes in Theoretical Computer Science 59 No. 3., November 2003.
[12] Christian, C., An Information Theoretic Model for Steganography, Information Hiding 1998, LNCS 1525, pp. 306-318, 1998.
[13] Dan, B., James, S., Collusion-Secure Fingerprinting for Digital Data, IEEE Transactions on Information Theory, Vol. 44, No. 5, September 1998.
[14] Binomial Distribution, http://en.wikipedia.org/wiki/Binomial_distribution
[15] Digital Rights Management, http://en.wikipedia.org/wiki/Digital_rights_management

〈著者紹介〉

정 보 흥 (Bo-heung Chung) 정회원
1996년 2월: 인하대학교 컴퓨터공학과 졸업
1998년 2월: 인하대학교 컴퓨터공학과 석사
2002년 2월: 한국대학교 컴퓨터공학과 박사
2002년 ~ 현재: 한국전자통신연구원 선임연구원
〈관심분야〉 정보보호, 네트워크 보안, 침입탐지

김 정 녀 (Jeong Nyeo Kim) 종신회원
1987년: 전남대학교 전산통계학과 졸업
1996년: OSF/RI 공동연구 파견(미국)
2000년: 충남대학교 컴퓨터공학과 석사
2004년: 충남대학교 컴퓨터공학과 박사
2005년: Univ. of California, Irvine Post-Doc.
현재: 한국전자통신연구원 휴먼인식기술연구팀장 책임연구원
〈관심분야〉 시스템·네트워크보안, 보안 OS, 바이오보안 등