필수 서비스 보호를 위한 자원 재할당

(Resource Reallocation for the Protection of Essential Services)

민 병 준 † 김 성 기 †† 최 중 섭 ††† 김 홍 근 ††††
(Byoung Joon Min) (Sung-ki Kim) (Joong-sup Choi) (Hong Geun Kim)

요 약 새로운 방법의 시스템 공격에 대해서도 시스템의 생존성을 보장하기 위해서는 필수 서비스를 위한 중요 자원을 식별하고 위급 상황에 적절히 대응하는 방안이 필요하다. 본 논문에서는 침입 감내 시스템 구축을 위한 핵심 기술의 하나인 동적 자원 재할당 기법을 제시한다. 이 기법에서는 선택된 필수 서비스에 대해 해당 노드 내에서 자원을 재할당하여 침입이 발생한 후에도 필수 서비스를 보호하여 생존할 수 있도록 한다. 이러한 노드 내에서의 조치에도 불구하고 필수 서비스의 생존성이 확보되지 않으면 준비된 다른 서버 노드로 서비스 제공이 전환될 수 있도록 하는 노드간의 자원 재할당이 이루어지도록 한다. 테스트베드를 구축하여 실험을 실시한 결과 본 자원 재할당 기법의 타당성을 입증할 수 있었다. 향후 이 기법을 침입 탐지 시스템과 접목시키면 매우 효과적인 공격 대응 방안이 될 것이다.

키워드 : 침입 감내, 의존성, 생존성, 동적 자원 재할당, 성능 모니터링

Abstract In order to guarantee system survivability against attacks based on new methodology, we need a solution to recognize important resources for essential services and to adapt the urgent situation properly. In this paper, we present a dynamic resource reallocation scheme which is one of the core technologies for the implementation of intrusion tolerant systems. By means of resource reallocation within a node, this scheme enables the essential services to survive even after the occurrence of a system attack. If the settlement does not work within the node, resource reallocation among nodes takes places, thus the essential services are transferred to another prepared server node. Experimental result obtained on a testbed reveals the validity of the proposed scheme for resource reallocation. This scheme may work together with IDS(Intrusion Detection System) to produce effective responsive mechanism against attacks.

Key words : Intrusion Tolerance, Dependability

1. 서 론

현재의 네트워크는 무제한으로 분산되어 있다. 중앙 집중의 관리 체계가 없어서 지역에서 관리하는 영역 밖의 네트워크 상황을 정확히 볼 수 없게 되어있다. 이러한 상황에서 외부의 공격이나 내부적 결함이 발생하더라도 시스템에 주어진 임무를 완수할 수 있는 능력, 바로 생존성(survivability)이 새로운 시스템 요구 항목으로 등장하게 되었다.

시스템 생존에 위협을 주는 행위나 현상은 다음과 같은 세 가지로 나누어 생각할 수 있다. 외부로부터의 침입이나 서비스 거부와 같은 공격, 시스템이 의존하고 있는 요소의 결함에 의한 고장, 마지막으로 자연 재해와 같은 사고이다. 이 중에서 외부로부터의 시스템 공격, 특히 침입에 대응하기 위한 시스템 설계 기술을 보면 다음의 세 단계로 나눌 수 있다.

- 침입 예방 : 이미 잘 알려진 시스템의 취약점을 막기 위한 조치이다.
- 침입 탐지 : 침입 행위를 검출하고 경보를 발생시키는 조치이다.
- 침입 감내 : 소프트웨어 설계부터 안전성 보장을 위해 예상치 못한 침입에 대응하는 것이다. 침입 탐지에 의해 침입이 검출되면 일정 수준 이상의 서비스를 지속하

· 본 연구는 한국정보보호진흥원 위탁과제 지원과 인천대학교 MRC 지원에 의해 수행되었음
† 종신회원 : 인천대학교 컴퓨터공학과 교수
bjmin@incheon.ac.kr
†† 비 회 원 : 인천대학교 컴퓨터공학과
proteras@incheon.ac.kr
††† 비 회 원 : 한국정보보호진흥원 연구원
jschoi@kisa.or.kr
†††† 비 회 원 : 한국정보보호진흥원 기술단장
hgkim@kisa.or.kr
논문접수 : 2002년 8월 12일
심사완료 : 2003년 8월 22일

기 위해 침입자를 봉쇄하고 시스템 침투에 대한 영향을 최소화하며 시스템 재구성과 복구를 행한다 [1,2].

자동화된 작업이나 지능화된 툴에 의해 실시간으로 침입시도를 탐지하는 침입탐지 시스템(Intrusion Detection System : IDS)에 관해서는 최근 많은 연구 진척이 있었다. 알려진 공격유형을 데이타베이스에 저장시켜 놓고 정해진 규칙에 부합하는 네트워크 트래픽이 탐지되면 침입 시도라고 나타낸다. 이 방식의 가장 큰 결점은 현재 유행하고 있는 방식에 의존해야 하며 지속적인 관리를 요구한다는 것이다.

어떠한 상황에서도 임무를 반드시 완료해야 하는 시스템에서는 침입 감내의 요구가 매우 높다. 일반적인 실시간 결함허용 시스템에 비하여 침입 감내 시스템은 보다 폭 넓은 유형의 결함에 대비할 수 있어야 한다. 하드웨어 결함이나 소프트웨어 결함 뿐 아니라 침입에 의해 발생할 수 있는 임의의 결함 유형에 대한 대비책이 있어야 한다. 시스템의 생존성을 보장하기 위해서는 시스템, 네트워크의 중요 서비스를 제공하는 데 필수적인 중요 자원을 식별하고 위급 상황에 적절히 대응하기 위한 자원 재할당 시스템에 대한 연구가 필요하다. 본 논문에서는 침입에 대응하여 자원을 재할당하기 위한 방법을 제시한다. 우선 필수 서비스를 선택하고 선택된 서비스를 위하여 동적으로 자원을 재할당하여 침입이 발생한 후에도 필수 서비스가 생존할 수 있도록 하는 구체적 방안과 테스트베드를 통한 실험 결과에 대하여 논한다.

본 논문의 2장에서는 생존성과 의존성과 관련된 기존의 연구 결과를 정리한다. 3장에서는 이를 토대로 침입 감내 시스템에 대한 개념 정의에 대하여 논하고 침입 감내 시스템 구축을 위한 핵심 기술들에 대하여 설명한다. 4장에서는 침입에 대응하여 자원을 재할당하기 위한 방법을 제시한다. 테스트베드의 구현 및 실험 결과에 대하여 5장에서 논하고 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

이 장에서는 시스템의 생존성(survivability) 보장 기술과 의존할 수 있는(dependable) 시스템 구축 기술 관련 최근 연구 결과에 대하여 논한다.

National Security Agency에서는 미국 국방성을 비롯한 미국의 중대한 정보 인프라가 의존하는 네트워크 정보 시스템의 일반적인 취약점의 원인을 도출하고 취약점에 대응하는 보안 기술들을 다양성, 자원할당, 다중화와 같은 13가지로 분류하고 있다[3]. 이들 중에서 동적 자원 할당은 주로 정보 자원 활용의 우선 순위를 요구한다. 동적 자원 할당에서는 명령권자가 어떤 행위와 자원이 가장 중요한가를 생각해 두어야 한다는 것을 의미한다. 그러다가 응급상황이 발생하면 제한적인 통신 대역과 부족한 연산 능력을 가지고 어떤 활동이 먼저 이루어져야하고 어떤 자원이 높은 우선 순위를 가져야 하는가가 명확하게 제시되어야 한다. 현 시스템의 부하 측정 또는 예견되는 위협을 감안하여 고객과 프로세스의 우선 순위를 두고 비필수적인 기능을 축소하여 시스템을 보호는 경우가 예이다.

EC(European Commission)에서는 IST(Information Security Technologies) 프로그램을 통해서 시스템의 우연히 발생한 고장이나 악의적인 공격을 포함한 부주의한 행동에 대하여 중대한 정보 인프라를 보호하려는 문제를 제기하였다. DEPPY(European Dependability Initiative의 줄임말)는 의존성 지원 기술을 증진시켜서 시스템과 서비스의 신뢰와 확신을 높이는 데 기여하기 위한 활동을 해왔다[4,5].

Brian Randell은 의존성이란 시스템이 제공하는 서비스를 정당하게 신뢰할 수 있음으로 정의하고 있다[6]. John Knight는 이질적인 하드웨어 플랫폼, 운영체제, 응용 소프트웨어, 관리영역상에 존재하는 critical infrastructure 응용의 생존성 보장을 위한 방안을 제시하였다[6]. 침입감내 개념정리가 일부 되어 있는데 우선 침입 감내의 목표는 침입 발생 후에도 정보의 기밀성과 완전성를 유지하기 위한 것이다[7,8]. 단순한 복제 기법만으로는 해결할 수 없음을 강조하고, 결함허용 메커니즘을 적절히 사용하되 결함의 독립성 가정, 즉 하나의 노드를 침해하는 것 보다 여러 개의 노드를 침해하는 것이 훨씬 어렵다는 가정이 만족되도록 고려해야 한다는 것이다. Koilpillai 등은 Recon이란 솔루션의 개념을 소개하였다[3]. QuO(Quality Object)는 미들웨어 레벨에서 생존성 메카니즘을 지원한다. 미들웨어를 이용하면 이기종 호스트의 생존성 보장에는 유리하지만 각 호스트에 plug-in이 필요하게 된다. 이 아이디어의 핵심은 시스템이나 주변상황에 변화가 생기면 응용이 이에 적응하기 위한 어떤 범용 메커니즘을 동원하여 해결한다는 것이다. Mellia-Smith 등은 연성 실시간 분산 시스템을 위한 자원 관리 방안을 논하였다[5]. Hiltunen 등은 customization과 adaptability를 통해서 생존성을 확보하기 위한 연구 결과를 발표하였다[1]. 핵심 개념은 다양성과 적응성이다. 또한 투명하게 생존성을 보강하기 위해서는 통신 네트워크와 OS를 교체하거나 미들웨어를 삽입하는 방법이 유효하다는 것이다. 기존 COTS 미들웨어의 취약점이 제기되었다[9]. CORBA, WAP, XML이 침입 위협에 노출되어 있다는 점을 강조하고 이를 보상하기 위한 ICM(Intelligent Compensating Middleware)를 제시하였다. ICM의 핵심 아이디어는 응용의 호출을 가로채기하고 적절한 보상, 예를 들면, startup, normal, under attack 등을 위한 호출을 하고

보상이 종료되면 COTS 미들웨어를 호출하여 COTS 미들웨어의 취약점을 줄여보자는 것이다.

3. 침입 감내 시스템

침입 감내는 여전히 남아 있는 보안 취약성을 처리하기 위한 것이다. 정보 시스템의 보안에 대한 취약성을 모두 제거하거나 완벽한 조치를 취하지 않는 한 침입 방지나 침입 탐지가 완벽하게 이루어질 수 없다. 따라서 침입 행위를 감내할 수 있는 시스템이 필요하게 된 것이다.

침입 감내의 목표는 침입 감내의 요구 특성에 의해 정해지게 된다. 이 요구 특성은 시스템의 의존성으로부터 부여될 수 있다. 의존성을 나타내는 특성을 그림 1에 나타낸 바와 같이 가용성, 신뢰도, 안전성, 보안성으로 정의하고 있다[9,10].

침입이 발생한 후에도 위에서 언급한 특성들이 유지되게 해주는 것이 침입 감내라고 한다면 침입 감내의 목표를 필수 서비스 유지, QoS trade-off, 비노출 자원, 정보의 기밀성과 완전상태 보존으로 정의할 수 있다. 특히, 침입이 발생한 후에도 사전에 정의된 필수 서비스를 지속할 수 있도록 하는 것이 중요하다. 침입이 발생하지 않은 상태에 비해서 제한된 컴퓨팅 또는 네트워크 자원으로 서비스를 유지해야 하기 때문에 사용자에게 제공하는 정보의 양을 줄이거나 서비스 제공에 보다 많은 시간이 소요될 수 있다. 즉, 보안과 성능간의 trade-off를 행할 수 있어야 한다. 공격자의 손이 닫지 않는 곳에 자원을 숨겨둘 수 있어야 하고, 공격자가 침입하더라도 정보를 부당하게 훔치거나 원래 상태를 변형시킬 수 없도록 해야 한다.

근본적으로 침입 감내 시스템을 구축하기 위해서는 그림 1에 나타낸 바와 같이 우선 침입 감내 요구 특성을 만족할 수 있는 아키텍처를 기술하고 목표로 하는 특성이 나올 수 있는 구체적 아키텍처로 전개되어야 한다. 이를 위해서는 어떤 침입이 발생할 수 있는가? 침입이 생존성에 미치는 영향은 무엇인가? 시스템 아키텍처를 어떻게 바꾸면 이러한 위험을 줄일 수 있는가? 등에 대한 답을 구해야 한다.

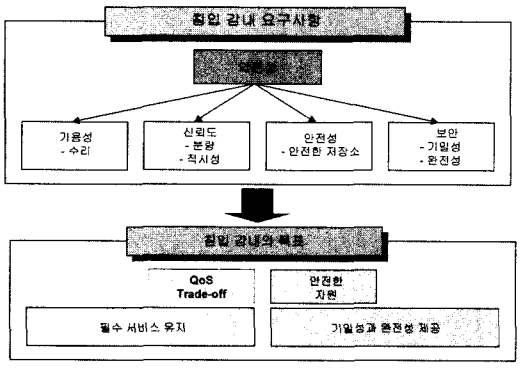

그림 1 침입 감내의 요구 특성 및 목표

본 논문은 침입 감내의 목표 중에서 필수 서비스를 보호, 유지하기 위해 자원을 재할당하는 기법에 대한 것이다.

4. 자원 재할당 기법

이 장에서는 침입을 당한 경우에도 필수 서비스를 제공하고 궁극적으로는 전체 서비스를 복구하는 시스템 기능 구현을 목적으로 하는 자원 재할당 기법에 대하여 논한다.

본 논문에서 제시하는 자원 재할당 시스템의 구조는 외부 공격에도 불구하고 클라이언트에게 지속적인 서비스를 제공할 수 있도록 하는 것이다. 이를 위해서 프로세서, 메모리, 네트워크와 같은 자원을 노드의 상황에 따라서 동적으로 재할당하도록 한다.

4.1 필수 서비스 선택

한 노드가 공격자의 침입을 받으면 그 행위로 인해 노드가 가지고 있던 프로세서, 메모리 같은 컴퓨팅 자원과 통신할 수 있는 네트워크 자원의 손실이 발생하게 된다. 이 손실이 많아지면 정상적으로 수행하던 서비스를 제공할 수 없는 상태에 이르게 된다. 본 연구에서는 이러한 상황에 적응력을 발휘할 수 있도록 하기 위해 서버 관리자가 침입이 발생한 후에도 유지되어야 하는 필수 서비스를 사전에 선택하도록 한다. 선택된 필수 서비스를 실행하기 위한 최소한의 자원을 확보해두면 침입을 당한 후에도 서비스를 유지할 수 있다는 것이다. 물론 선택되지 않은 비필수, 즉 양보할 수 있는 서비스는 포기할 수도 있다.

▷ 필수 서비스와 양보 가능한 서비스의 구분

기본적으로 서버가 제공하려는 추상 수준의 응용 서비스 예를 들어 전자상거래와 같은 웹 응용은 쉽게 정의될 수 있으나 문제는 그러한 응용 서비스를 제공하는데 필요한 여러 기본 서비스들이 존재하게 된다는 것이다. 즉 서비스들간의 의존성이 존재하게 되고 하나의 필수 서비스를 유지하기 위해서는 그 서비스가 의존하는 다른 서비스들도 보호되어야 한다는 것이다. 따라서 하나의 필수 서비스가 서버 관리자에 의해 선택되면 그 서비스가 의존하는 다른 모든 서비스들도 필수 서비스로 선택되도록 한다. 예를 들어 Windows2000에서 FTP Publishing 서비스가 필수 서비스로 선택되면 IIS Admin 서비스와 RPC 서비스도 필수 서비스로 선택된다는 것이다.

▷ 필수 서비스에 필요한 자원 확보 방안

필수 서비스가 수행되는데 필요한 충분한 자원이 확보되어 있는가를 주기적으로 모니터링 한다. 사용 가능한 프로세서, 메모리, 그리고, 네트워크 통신량을 측정하여 이것이 일정 수준 이하로 떨어지면 서비스를 수행하기 위한 환경이 나빠진 것으로 판단한다.

▷ 필수 서비스에 대한 가정

여기서 설계상의 한가지 가정이 필요한데, 침입자가 활동하는 프로세스가 필수 서비스에는 접근하지 못한다는 것이다. 즉 보호하려고 하는 필수 서비스에는 침입이 이루어지지 않는다는 것이다. 본 논문에서는 이 문제가 다루어지지 않았으나 이 가정을 만족하기 위한 한가지 방안은 커널을 드라이버 형태로 구현해 놓고 유저 모드나 커널 모드의 프로세스가 생성되기 위해서 불려지는 커널 호출을 인터셉트해서 정당한 것인가를 판단하는 일종의 소프트웨어 랩핑 방법을 적용할 수 있을 것이다.

4.2 노드 내에서의 자원 재할당

노드 내에서의 자원 재할당이란 앞서 설명한 필수 서비스가 필요로 하는 최소한의 자원이 존재하지 않는 경우 같은 노드 내에서 자원을 많이 보유하고 있는 양보 가능한 서비스로 하여금 자원을 반납하고 정지하게 하는 것이다. 즉, 동적으로 자원이 재할당되게 하여 필수 서비스가 생존하게 하자는 것이다. 이를 위해 해결해야 할 중요한 문제는 필수 서비스가 생존 가능한지 불가능한지를 어떻게 판단하는가 하는 것이다.

▷ 필수 서비스의 생존 가능성 평가

필수 서비스가 생존하기 위해서는 일정 수준 이상의 프로세서, 메모리, 네트워크 자원이 필요한데, 그 수준이 어느 정도인지 단정적으로 말하기는 힘들다. 여러 태스크들이 동시에 수행되고 있는 서버에서 특정 서비스만을 위해서 모든 자원을 할당해주면 서버의 throughput이 저하되기 때문이다. 반대로 침입자가 활동하고 있을 수도 있는 양보 가능한 서비스로 인하여 필수 서비스의 서비스 품질이 지나치게 저하될 수 있다.

이 문제를 해결하기 위해 본 연구에서는 기준선(baseline)과 일정 시간 간격 동안의 기준선 초과 누적 시간을 적절히 설정하도록 한다. 이 개념을 설명하기 위하여 한 노드내의 프로세서 사용량이 변하는 경우를 예로 들어 설명한다.

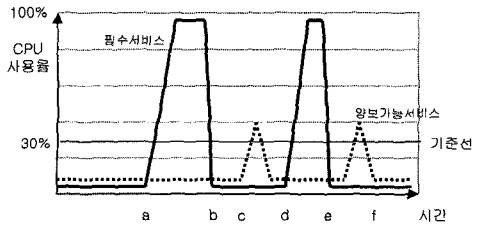

그림 2 프로세서 사용 그래프

그림 2에서 가로축은 시간을 나타내고 세로축은 프로세서 사용량을 나타낸다. 진한 실선으로 표시된 필수 서비스의 프로세서 사용량을 보면 시간 a까지는 거의 유휴 상태로 있다가 전체 프로세서 용량의 90%이상을 점유하여 서비스를 제공하고 시간 b 지점에서 종료한 것으로 설명된다. 그러다 다시 시간 d e 사이에서 또 한 번 서비스 제공을 위해 프로세서를 사용한 것이다. 점선으로 표현된 것은 침입에 의한 것일 수도 있는 어떤 양보 가능한 서비스가 프로세서를 사용한 것을 나타낸 것이다. 이 경우는 필수 서비스 수행이 양보 가능 서비스 수행에 의해 별 영향을 받지 않은 경우이다. 그림에서 CPU 사용률 30% 지점에 기준선이라고 표시된 것은 그림의 필수 서비스가 최저 품질의 서비스를 제공하는데 필요한 최소한의 프로세서 점유 비율이다.

▷ 기준선 및 기준선 초과 누적 시간

기준선은 각 필수 서비스 별로 설정이 가능한데 기준선 설정 값은 해당 필수 서비스가 허용할 수 있는 가장 낮은 품질로 서비스를 제공한다고 할 때 필요한 자원의 요구량으로 생각할 수 있다. 즉, 정상적인 경우에 필수 서비스를 제공하기 위해서는 자원 사용량이 기준선을 초과하게 되어 있다는 것이다.

기준선 초과 누적시간은 순간적인 자원 사용량의 변화에 대응하기 위한 것으로 일정 시간 간격 동안 기준선을 초과한 누적 시간인데 이를 기준선을 초과한 것을 판단하기 위한 임계값이다.

기준선과 기준선 초과 누적 시간은 각각의 필수 서비스 뿐 아니라 전체 서비스의 사용량 합계에 대해서도 설정할 수 있다. 예를 들어, 프로세서 사용량이 최근 1분 동안 90% 이상을 유지한 시간이 6초 이상을 초과하였다면 프로세서가 매우 바쁘게 동작하고 있고 여유 자원이 없다는 것으로 볼 수 있다. 이때, 90%가 전체 서비스에 대한 기준선이고 6초가 기준선초과 누적시간이다. 본 논문에서 제안하는 노드 내 자원 재할당 알고리즘은 다음과 같다.

▷ 노드 내 자원 재할당 알고리즘

(1) 우선 전체 서비스의 자원 사용량이 설정된 기준선의 초과 누적시간을 넘지 않는 경우에는 충분한 여유 자원이 있어서 필수 서비스를 수행하는데 아무런 문제가 없는 것으로 보고 모니터링을 지속한다. 만일 기준선을 초과한 시간이 길어지게 되면 여유 자원이 거의 남아 있지 않다는 의미이므로 다음과 같은 조치를 취한다.

(2) 자원이 남지 않게 되는 경우는 그림 3에 나타낸 것과 같이 크게 3가지의 경우를 생각할 수 있다. 그림의 (a) 경우는 자원의 대부분을 필수 서비스가 차지하고 있는 것이다. 비록 여분의 자원이 없지만 필수 서비스가

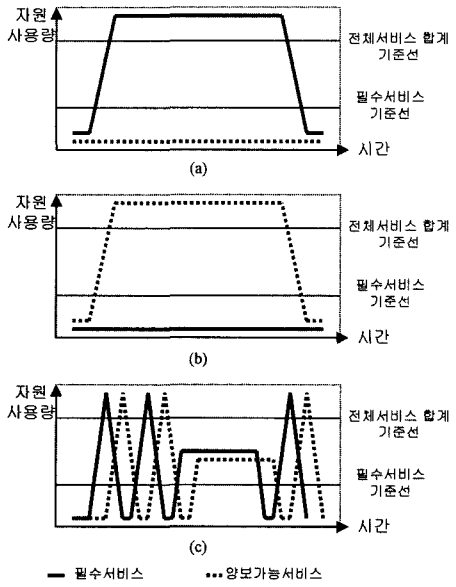

그림 3 필수 서비스의 생존성 평가

사용하고 있음으로 문제가 없다. 여기서 필수 서비스가 자원을 사용하고 있다는 판단은 필수 서비스의 자원 점유율이 필수 서비스에 대한 기준선을 초과한 것에 기인한다.

(3) 그림의 (b)의 경우는 자원을 양보 가능한 서비스가 대부분 차지하고 있는데 이 때에는 자원을 차지하고 있는 양보 가능한 서비스의 사용 우선 순위를 인위적으로 낮추어 일부 자원이 돌아오도록 한다. 우선 순위를 낮추어도 그래프의 모양이 크게 변하지 않는다면 필수 서비스는 현재 대기 상태임을 의미하므로 그대로 경과를 두고 본다.

(4) 그러나 만일 그래프의 모양이 그림의 (c)와 같이 또는 (a)와 같이 바뀐다면 필수 서비스가 자원을 필요로 하고 있음에도 불구하고 그 동안 양보 서비스에 의해 자원을 할당받지 못했다고 판단하여 해당 양보 가능 서비스의 자원을 반납하도록 종료시킨다.

그림 4는 노드 내에서의 자원 재할당 알고리즘을 나타낸 것이다. m개의 필수 서비스를 위한 프로세스와 n개의 비필수 서비스만을 위한 프로세스가 한 노드 내에 존재하는 경우의 알고리즘이다. CheckTotalResourceUsage 함수는 앞서 설명한 (1)의 조건을 확인하기 위한 것으로 전체 자원 사용량(TotalResourceUsage)이 정해진 기준선(BaselineTotal)을 초과한 누적량(excessCount)이 임계치(threshold)를 초과하면 True를 반환한다. CheckEssentialServiceResourceUsage 함수는 앞서 설명한 (2)의 경우 중 필수 서비스에게 충분한 자원이 확보되어 있는 가를 확인하기 위한 것이다. 필수 서비스를 위한 프로세스의 자원 사용량(ResourceUsage[EP[i]])이 사전에 정해진 기준선(Baseline[EP[i]])을 초과한 누적량(excessCount)이 임계치(threshold[i])를 초과한 프로세스의 개수를 반환한다. 모든 필수 서비스가 자원을 충분하게 확보하고 있지 않다면 비필수 서비스만을 위한 프로세스를 자원 사용량에 따라 내림차순으로 정렬하고 자원 사용량이 많은 프로세스의 자원 일부를 회수한다. 이렇게 했을 때 필수 프로세스의 자원 점유가 늘어나면 해당 비필수 프로세스를 강제 종료시켜서 자원을 반환하게 하고 그렇지 않으면 원상 복구하는 내용이 Resource_Reallocation 프로세저에 포함되어 있다.

4.3 노드간의 자원 재할당

노드 내에서의 자원 재할당으로도 필수 서비스의 품질을 유지하는데 필요한 자원이 확보되지 않으면 다른 서버로 필수 서비스 제공이 유지되도록 노드간의 자원 재할당을 수행한다. 본 논문에서 적용한 네트워크 로드 밸런싱 메카니즘[16,17]은 서비스 수행 노드의 생존 능력이 결여되었을 때, 클라이언트의 접근을 생존하고 있는 다른 노드의 컴퓨팅 자원을 이용할 수 있도록 분산함으로써 지속적인 서비스 수행을 보장한다.

5. 테스트베드의 구현 및 실험 결과

자원 재할당 기법의 타당성을 검증하기 위하여 테스트베드를 구현하였다. 이 장에서는 테스트베드 구현 방법과 실험 결과에 대하여 논한다.

5.1 테스트베드 구현

테스트베드의 전체 구조는 그림 5와 같다. 여러 대의 멤버 서버 노드들과 프론트엔드 노드가 이더넷으로 연결되어 있다. 4장에서 설명한 노드 내의 자원 재할당 알고리즘의 주요 임무는 LSM(Local Survivability Manager)에 의해 다루어진다. 또한 노드 간의 자원 재할당은 프론트엔드 노드의 GSM(Global Survivability Manager)이 관리하는데 노드 간의 전환을 구현하기 위하여 Windows 2000 Advanced 서버의 OS에서 제공하는 네트워크 로드 밸런싱 서비스를 이용하였다.

응용 객체로 구현한 LSM은 감시 대상이 되는 필수 서비스 객체와 필수 서비스 객체를 위해 가용 자원을 양보할 수 있는 양보 서비스 객체들을 구분할 수 있도록 사용자 인터페이스를 제공한다. 또한, 필수 서비스의 생존성 평가를 적용할 수 있는 기준선과 기준선 초과 누적 시간을 설정할 수 있는 사용자 인터페이스도 제공한다. 여기서 설정된 감시 대상 객체의 성능 카운터 값들은 미들웨어 레벨의 HMT(Health Monitor Thread)로부터 실시간으로 수집되어 SET(Survivability Evaluation Thread)로 전달된다. SET는 수집된 정보를 이

```
{ m개의 필수 서비스를 위한 프로세스와 n개의 비필수 서비스만을 위한 프로세스가 존재 }
var
   EP[m] : array of essential process id's;
   DP[n] : array of dispensable process id's;
   TotalResourceUsage, BaselineTotal, ResourceUsage[], Baseline[] : integer;

procedure Resource_Reallocation(m, n : integer);
var
   i : integer;
begin
   if CheckTotalResourceUsage(mon_slot, threshold) then
   begin
      if (CheckEssentialServiceResourceUsage(m, mon_slot, threshold) < > m) then
      begin
         sort dispensable processes stored in DP[n] in descending order of resource usage;
         for  i := 1 to n do
         begin
            withdraw resource of DP[i];
            if EssentialServiceResourceUsage has increased then
               terminate DP[i];
            else
               restore the withdrawn resource to DP[i]
         end;
      end;
   end;
end;

function CheckTotalResourceUsage(mon_slot, threshold : integer) : boolean;
var
   i, excessCount : integer;
begin
   excessCount := 0;
   for i := 1 to mon_slot do
      if TotalResourceUsage > BaselineTotal then
         excessCount := excessCount + 1;
   if excessCount > threshold then
      CheckTotalResourceUsage := true;
   else
      CheckTotalResourceUsage := false
end;

function CheckEssentialServiceResourceUsage(m, mon_slot, threshold[m] : integer) : integer;
var
   i, j, k, excessCount : integer;
begin
   k := 0;
   for i := 1 to m do
   begin
      excessCount := 0;
      for j := 1 to mon_slot do
         if ResourceUsage[EP[i]] > Baseline[EP[i]] then
            excessCount := excessCount + 1;
      if excessCount > threshold[i] then
         k := k + 1
   end;
   CheckEssentialServiceResourceUsage := k
end;
```

그림 4 노드 내 자원 재할당 알고리즘

용하여 주기적으로 특정 서비스 및 노드 차원에 대한 생존성 평가를 앞서 설명한 알고리즘에 따라 수행한다. 이를 위해, HMT는 감시 대상 객체들의 성능 카운터 값을 저장하는 데이타 블록을 가지고 있으며, SET는 감시 대상 객체의 성능 카운터 값과 평가 기준을 저장하는 데이타 블록을 가지고 있다. HMT는 자신의 쓰레드 수행 시간 내에 Registry API를 이용하여 자신 데이타 블록과 SET의 데이타 블록으로 수집한 성능 카운

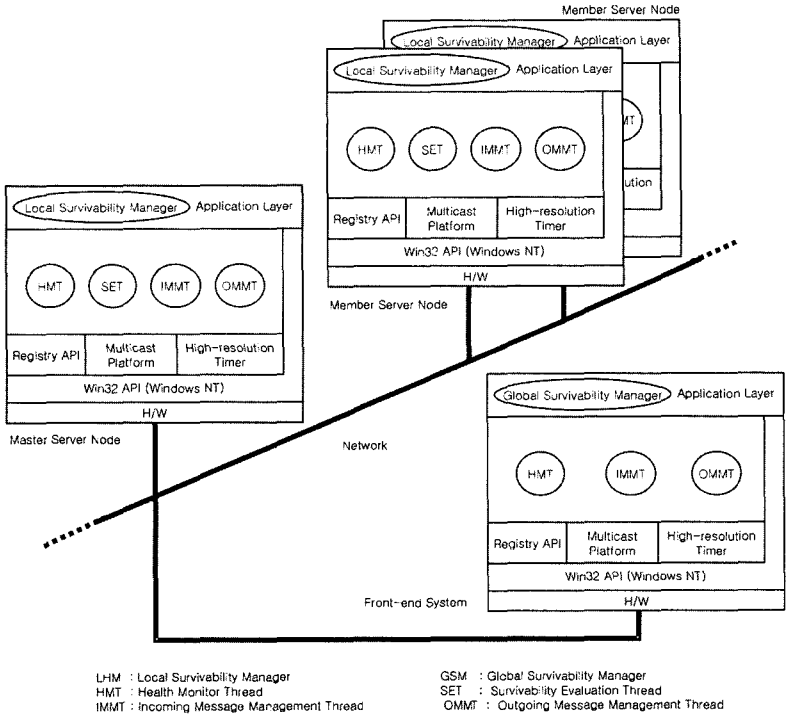

그림 5 자원 재할당 테스트베드 구조

터 값을 저장한다. SET의 데이타 블록은 생존성 평가에 이용되며, HMT의 데이타 블록은 LSM의 사용자 인터페이스를 통한 화면 표시에 이용된다.

생존성 평가 결과는 LSM이 자원 재할당을 위한 코드 수행 여부를 결정하는 기준이 되며, SET가 생존성 평가 결과에 따라 LSM의 자원 재할당 수행 여부를 결정하는 플래그 값을 설정한다. 생존성 평가 결과로 자원 재할당 수행이 결정되면, LSM은 Registry API를 이용하여 사용자로부터 미리 설정된 양보 가능한 서비스 객체들을 종료시켜 이들 객체들의 가용 자원을 반환하도록 제어한다. 이때 양보 가능한 서비스 객체들의 종료 우선 순위는 자원의 이용도가 높은 순서를 따른다. SET의 다음 주기에서 자원 재할당 수행 이후의 생존성 평가가 다시 이루어진다. 이때 생존성 평가에 실패했을 경우에는 LSM이 OMMT(Outgoing Message Management Thread)를 통하여 프론트엔드 시스템의 HMT에 이 사실을 통보한다.

프론트엔드 시스템의 HMT는 각 지역 서버 노드의 SET로부터 주기적으로 생존성 평가 결과를 보고 받아 자신의 데이타 블럭에 각 노드별 생존성 여부 및 상태를 감시한다. 이때 특정 멤버 노드의 LSM 으로부터 자원 재할당 실패 메세지를 수신하면 해당 노드의 이상을 사용자에게 경보하고 해당 노드를 종료하여 재구성에 들어가게 한다. 이때 클라이언트로부터의 지속적인 요청 메시지는 네트워크 차원에서 서비스되는 네트워크 로드 밸런서를 통해, 현재 생존중인 다른 노드로 메시지를 분산시켜, 노드 단위의 자원 재할당을 수행한다.

분산된 컴퓨팅 노드 중에 서비스 수행 멤버 노드로 참가하는 컴퓨팅 노드에게 네트워크 로드 밸런싱 환경을 구성하기 위해서는 멤버 노드의 IP 주소는 모두 동일한 서브넷에 속하고, 멤버 노드 모두가 공통된 가상 IP 주소를 가져야 한다.

실험에서 생존성 평가 기준으로 노드 차원의 총 프로세서/메모리 이용도, TCP/IP 연결 수와 같은 자원 이용도에 대한 기준선과 이 기준선을 초과한 분당 횟수를 적용하였으며, 보호 대상 서비스 수행 객체에 대해서는 프로세서 이용도에 대한 생존성 평가 기준만을 적용하였다. 노드 차원의 총 프로세서 이용도란 현재의 순간에서 CPU가 쉬지 않고 일하고 있는 시간에 대한 백분율을 의미한다. 노드 차원의 총 메모리 이용도는 최대 메모리 용량에 대해서 현재 모든 서비스 수행 객체가 사용하고 있는 메모리 사용량에 대한 비율이다. TCP/IP 연결 수는 현재의 원격으로부터 연결이 이루어진 수를 의미한다.

5.2 실험 결과

실험을 위하여 이더넷 환경에서 각 서버(Win2K AS)

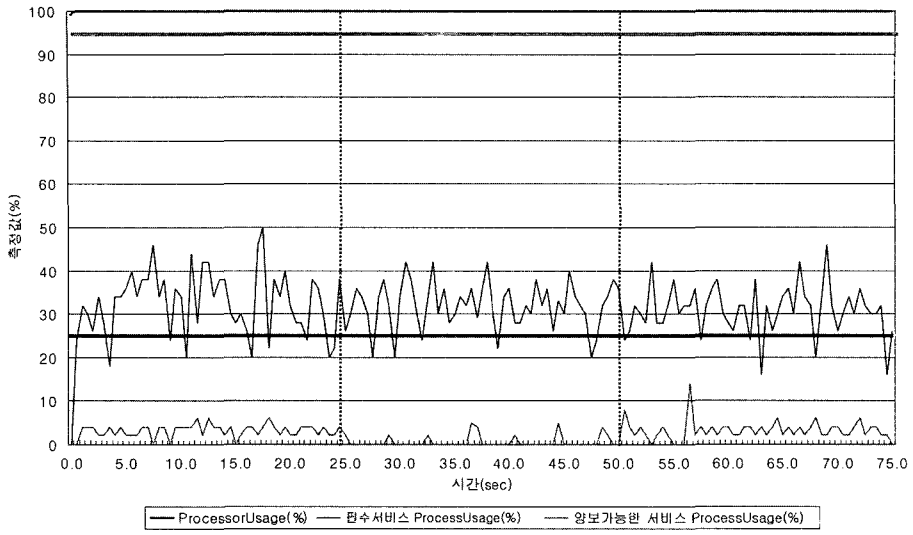

그림 6 실험 결과 1(필수 서비스 사용량이 기준선을 초과한 경우)

에 대한 다음과 같은 자원 이용도 측정 항목에 대한 측정값을 1초 단위로 저장하였다.

- 프로세서 사용량 : % Processor Time
- 네트워크 인터페이스 : Byte Received / Sec
- TCP : Connection Established / Sec

공격도구로는 인터넷상에서 수 십 가지의 도구가 있지만 윈도우 환경에서 사용하는 Divine, PacketBuilder, Nuke, PortFuck, Elite, ExploitGenerator 등과 같은 도구를 사용하였다. 공격도구들의 공통적인 입력 값은 표적 IP 주소, 포트 번호, 패킷량 등이다. FTP 최초 Configuration에서 설정한 최대 연결 수가 공격에 의해 Full되었을 때 더 이상의 서비스 이용이 불가능해졌다. 그런데 이와 같은 공격을 받으면 연결 수 뿐만 아니라 해당 서비스에 의해 사용되는 프로세서 요구량도 따라서 증가하는 것으로 나타났다. 따라서 보다 구체적인 실험 결과 설명을 위해 프로세서 사용량을 중심으로 설명하기로 한다.

그림 6은 필수 서비스의 프로세서 사용량이 기준선을 초과한 경우이다. 그림의 가로 축은 시간이고 세로 축은 프로세서 점유율을 나타낸다. 앞서 그림 3과 그림 4에서 설명한 전체 서비스 합계 기준선은 95%로, 진한 선으로' 사용량이 표현된 필수 서비스의 기준선은 25%로 되어 있다. 이 경우는 필수 서비스의 사용량이 기준선을 초과하고 있으므로 별다른 조치를 취하지 않고 성능 감시를 지속한 것이다. 그림에서 흐린 선으로 표현된 양보 가능한 서비스의 사용량은 5% 정도에 머무르고 있으나 전체 프로세서 사용량이 100%에 이르는 것은 그림에 나타낸 두 서비스 외에 다른 서비스가 같이 수행되고 있음을 말한다.

그림 7은 양보 가능 서비스를 강제 종료한 경우의 프로세서 사용량을 나타낸 것이다. 그림 6에서와 마찬가지로 필수 서비스의 기준선이 25%로 설정되어 있는데 전반부에 진한 선으로 나타낸 필수 서비스의 프로세서 사용량이 이에 미치지 못하게 되어 그림 4에 나타낸 알고리즘에 따라 양보 가능한 서비스의 우선 순위를 낮추어 프로세서 자원을 일부 양보하게 한 것이다. 그러자 양보 가능한 서비스의 점유율이 5% 이하로 크게 줄고 상대적으로 필수 서비스의 점유율이 높아졌다. 양보 가능한 서비스에 의해 필수 서비스의 품질이 저하되고 있었음으로 판단하여 양보 가능한 서비스를 강제 종료시켜서 필수 서비스의 품질이 유지되도록 한 것이다.

그림 8은 그림 7에서와 마찬가지로 필수 서비스의 프로세서 사용량이 기준선을 초과하지 못한 경우이다. 알고리즘에 따라서 양보 가능한 서비스의 우선순위를 낮추었으나 그림 7의 경우와는 달리 필수 서비스의 사용량이 증가하지 않아서 원상 복구한 경우이다. 이 때는 필수 서비스가 자원을 필요로 하지 않는 것으로 볼 수 있다.

이 실험으로 본 논문에서 제시한 자원 재할당 알고리즘이 타당한 것으로 판단할 수 있다. 네트워크 로드 밸런싱을 활용한 노드 간의 자원 재할당도 제대로 실행되었다. 그러나 이 자원 재할당 기법이 실제 시스템에 적용되기 위해서는 다음과 같은 문제점들이 앞으로 해결되어야 할 것이다.

- 알고리즘에 필요한 파라메터 값 설정 : 노드 내 자원 재할당 알고리즘에 있어서 각 필수 서비스에 대한

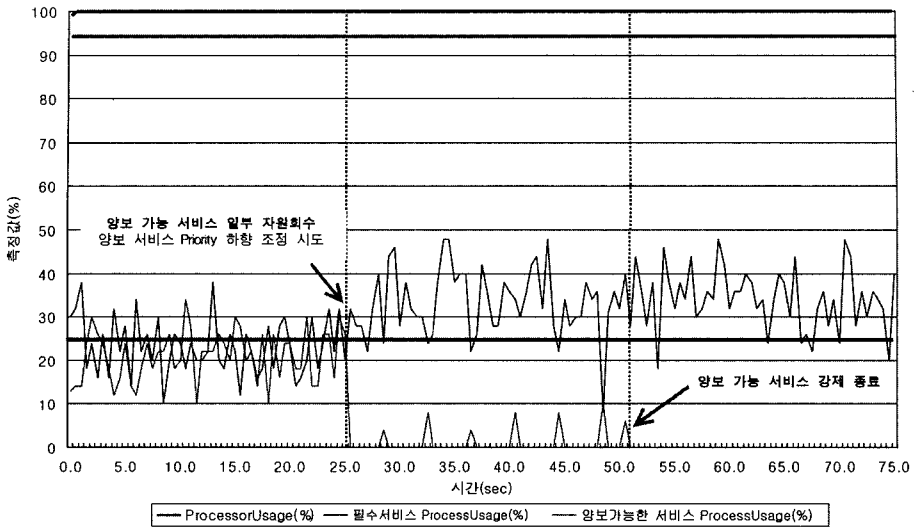

그림 7 실험 결과 2 (양보 가능 서비스 강제 종료한 경우)

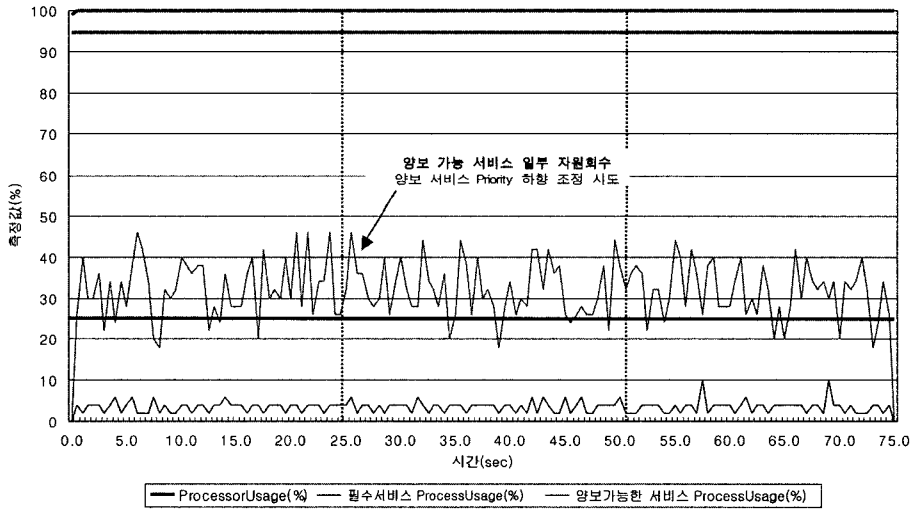

그림 8 실험 결과 3 (필수 서비스가 자원을 필요로 하지 않는 경우)

기준선, 측정 시간 간격, 그리고 양보 가능 서비스 종료를 판단하는 사용량 증가 비율을 정하는 문제이다. 이를 위해서는 정상적인 경우 각 서비스가 필요로 하는 자원 사용 패턴을 알고 있어야 한다. 따라서 이러한 유형 분석하고 알고리즘에 정확히 반영하는 방안이 강구되어야 한다.

• 노드 실패 도미노 현상 : 한 노드가 노드 내 자원 재할당에 실패해서 다른 노드로 서비스가 전환되어도 같은 공격이 계속되면 결국 연쇄적으로 노드가 실패하는 도미노 현상을 겪게 될 것이다. 이를 막기 위해서는 IDS(Intrusion Detection System)와의 연계가 필요하다. 즉, 한 노드가 자원 재할당에 실패하면 이를 IDS에 알려서 같은 공격을 차단하도록 해야 한다.

• 구현상의 문제 : 테스트베드 구현을 위해 많은 기능들이 윈도우에서 제공하는 서비스를 이용하고 있으나 실제로 많은 교묘한 공격들이 이를 피해갈 수 있다. 예를 들어 어떤 바이러스들은 단순한 프로세스 종료 명령이 제대로 적용되지 않는 것을 볼 수 있다. 그러나 이것은 구현 상의 문제이며 실제 시스템에 적용하기 위해서는 구현 기술을 보완해야 할 것이다.

6. 결 론

본 논문에서는 새로운 연구 이슈가 되고 있는 침입 감내 시스템의 개념을 정립하고 침입에 대응하여 미리

선택된 서비스를 보호하기 위하여 동적으로 자원을 재할당하는 구체적 방안과 테스트베드를 통한 실험 결과에 대하여 논하였다.

한 노드가 공격자의 침입을 받으면 그 행위로 인해 자원의 손실이 발생하게 되고 이 손실이 많아지면 정상적으로 수행하던 서비스를 제공할 수 없는 상태에 이르게 된다. 이러한 상황에 적응력을 발휘할 수 있도록 하기 위해 서버 관리자가 침입이 발생한 후에도 유지되어야 하는 필수 서비스를 사전에 선택하도록 하고 선택된 필수 서비스를 실행하기 위한 최소한의 자원을 확보해 두면 침입을 당한 후에도 서비스를 유지할 수 있다는 것이다.

제안된 자원 재할당 방법의 타당성을 검토하기 위하여 테스트베드를 구축하여 프로세서와 네트워크 사용량에 대해서 재할당 기법을 실험하였다. 실험 결과 본 논문에서 제시한 자원 재할당 방법이 매우 유력하고도 효과적인 침입 감내 시스템 구축 기술의 하나가 될 수 있다는 가능성을 보였다.

앞으로 이 기법이 실제 시스템에 적용되기 위해서는 서비스의 자원 이용 유형을 분석 적용하고, IDS와의 연계하는 방안, 구현상의 문제 등에 대한 보완 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] V. Stavridou, "Intrusion Tolerant Software Architectures," DARPA Information Survivability Conference & EXposition, 2001.

[2] DARPA의 Information survivability program(http://www.darpa.mil/ito)

[3] National Security Agency, Defence Advanced Research Projects Agency, Office of the Assistant Secretary of Defence, "Securing the U.S Defence Information Infrastructures: A Proposed Approach," 1998.

[4] Marc Wilikens, et. al., "An Agenda for a Dependability Initiative," Jan. 1998.

[5] Working Paper "European Dependability Initiative: Inventory of EC Funded Projects in the area of Dependability," Nov. 2000.

[6] John C. Knight, Matthew C. Elder, Xing Du, "Error Recovery in Critical Infrastructure Systems," Computer Security, Dependability & Assurance: From Needs to Solutions, 1998.

[7] C. Meadows, "Security and Dependability: Then and Now," Computer Security, Dependability & Assurance: From Needs to Solutions, 1998.

[8] V. Stavridou and B. Dutertre, "From Security to Safety and Back," Computer Security, Dependability & Assurance: From Needs to Solutions, 1998.

[9] Brian Randell, "Dependability - Unifying Concept," Computer Security, Dependability & Assurance: From Needs to Solutions, 1998.

[10] Marc Wilikens, et. al., "Defining the European Dependability Initiative," May 1998.

[11] Working Paper, "The European Dependability Initiative," Dec. 2000.

[9] Amjad Umar, et. al., "Intrusion Tolerant Middleware," DARPA Information Survivability Conference & EXposition, 2001.

민 병 준

1983년 연세대학교 전자공학과(학사) 1985년 연세대학교 전자공학과(석사) 1991년 미국캘리포니아주립대학교(UCI) 전기및컴퓨터공학과(박사). 1884년~1986년 삼성전자 연구원. 1992년~1994년 한국통신 선임연구원. 1995년~현재 인천대학교 컴퓨터공학과 부교수. 관심분야는 분산시스템, 통신망 관리, 보안

김 성 기

1996년 인천대학교 컴퓨터공학과(학사) 1998년 인천대학교 컴퓨터공학과(석사) 1998년~1999년 인천대학교 멀티미디어 연구센터 연구원. 2000년~현재 인천대학교 컴퓨터공학과 박사과정. 관심분야는 분산시스템, 실시간시스템, 결함허용, 침입감내

최 중 섭

1993년 인천대학교 전자계산학과(학사) 1995년 숭실대학교 컴퓨터학과(석사) 2000년 숭실대학교 컴퓨터학과(박사) 1995년~1996년 한국전산원 연구원 2000년~현재 한국정보보호진흥원 선임연구원. 관심분야는 컴퓨터시스템보안, 실시간시스템, 분산시스템

김 홍 근

1985년 서울대학교 컴퓨터공학과(학사) 1987년 서울대학교 컴퓨터공학과(석사) 1994년 서울대학교 컴퓨터공학과(박사) 1994년~1996년 한국전산원 선임연구원 전산망안전보안센터장. 1996년~2003년 한국정보보호진흥원 기술단장. 관심분야는 컴퓨터보안, 병렬 알고리즘