서비스 거부 공격에 대비한 자원 재할당 및 서버 중복 방안

민 병 준[†]·김 성 기[††]·나 용 희[††]·이 호 재[†††]·최 중 섭[††††]·김 홍 근[†††††]

요 약

컴퓨팅 노드가 가지고 있는 자원을 고갈시켜서 본래 의도한 서비스 제공을 방해하려는 서비스 거부 공격에 대응하기 위해서는 어떠한 경우에도 반드시 유지되어야 하는 필수 서비스를 위한 중요 자원을 식별하고 위급 상황에 적절히 시스템을 적응시키고 재구성하는 방안이 필요하다. 본 논문에서는 이를 위하여 두 단계의 대응 방안을 제시한다. 첫 번째 단계로 한 컴퓨팅 노드 내에서 선택된 필수 서비스에 대해 자원을 동적으로 할당하여 공격이 성공한 후에도 필수 서비스가 유지될 수 있도록 한다. 이 조치에도 불구하고 노드 내에서의 충분한 자원 확보가 불가능해지면 두 번째 단계로 미리 준비된 다른 컴퓨팅 노드에서 사용자에게 투명하게 필수 서비스가 제공될 수 있도록 중복성을 적용하는 방안을 제시한다. 테스트베드를 구축하여 실험을 실시한 결과 기법의 타당성을 입증할 수 있었다. 기존의 다른 방법과 성능 및 비용을 분석하여 비교하였다.

A Scheme of Resource Reallocation and Server Replication against DoS Attacks

Byoung Joon Min[†]·Sung-Ki Kim[††]·Yong-Hi Na[††]·Ho Jae Lee[†††]
JoongSup Choi[††††]·Hong Geun Kim[†††††]

ABSTRACT

In order to cope with DoS (Denial of Service) attacks disturbing delivery of intended services by exhausting resources of computing nodes, we need a solution to recognize important resources for the essential services which have to be maintained under any circumstances and to adapt the system to the urgent situation and reconfigure itself properly. In this paper, we present a two-phase scheme to handle the problem. In the first phase, by means of dynamic resource reallocation within a computing node, we try to make the selected essential services survive even after the occurrence of an attack. For the second phase when it becomes impossible to continue the service in spite of the actions taken in the first phase, we apply server replication in order to continue the transparent provision of the essential services with the end users by utilizing redundant computing nodes previously arranged. Experimental result obtained on a testbed reveals the validity of the proposed scheme. A comparison with other proposed schemes has been conducted by analyzing the performance and the cost.

키워드 : 생존성(Survivability), 자원 재할당(Resource reallocation), 서버 복제(Server Replication), 침입 감내(Intrusion Tolerance)

1. 서 론

대부분의 네트워크 정보시스템은 기밀성과 완전성 유지를 위해 중요 정보를 암호화하고 인증 받은 사용자만이 접근할 수 있도록 침입을 예방하고 있다. 그러나 시스템에 내포된 여러 가지 취약점을 이용한 침입이 지속적으로 발생하고 있다. 침입 탐지 시스템은 알려진 또는 정의된 공격을 효과적으로 차단할 수는 있으나 새로운 공격에 대응하기 위해서는 지속적인 관리를 필요로 하는 한계점을 가지고 있다.

한편 최근 새로이 개념이 정립된 침입 감내는 시스템 내의 일부분이 위협받더라도 최소한도의 서비스 수준을 유지할 수 있도록 하기 위한 것이다. 결함 허용 기법의 장점을 활용하여 알려지지 않은 새로운 공격에 대비한다. 문제가 발생하면 이로 인한 피해가 더 이상 확산되지 않도록 해당 영역을 차단하고 경우에 따라서는 일부 성능이 저하된 상태에서 서비스 제공이 지속될 수 있도록 시스템을 재구성하고 복구한다[1, 2].

어떠한 상황에서도 반드시 임무를 완료해야 하는 중요 시스템에서는 침입 감내의 요구가 매우 높다. 일반적인 실시간 결함 허용 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 서비스를 제공하는데 필수적인 중요 자원을 식별하고 위급 상황에 적절히 대응해야 한다. 시스템에서 제공되는 서비스의 종류를 어떠한 경우에도 반드시 유지되어야 하는 필수 서비스와 경우에 따라서 포기할 수 있는 양보 가능한 서비스로 구분하여 예상치 못한 침입

※ 본 연구는 정보통신연구진흥원 지원(과제번호 2002-S-402)과 인천대학교 RRC 지원에 의한 것임.
† 종신회원 : 인천대학교 컴퓨터공학과 교수
†† 준 회 원 : 인천대학교 대학원 컴퓨터공학과
††† 정 회 원 : 한국정보보호진흥원 연구원
†††† 정 회 원 : 한국정보보호진흥원 선임 연구원
††††† 정 회 원 : 한국정보보호진흥원 책임연구원 기술단장
논문접수 : 2002년 8월 7일, 심사완료 : 2002년 12월 3일

에 대해서도 필수 서비스는 반드시 유지할 수 있도록 한다. 컴퓨팅 노드가 DoS(Denial of Service)와 같은 공격을 받으면 그 침입 행위로 인해 노드가 가지고 있던 CPU, 메모리와 같은 컴퓨팅 자원과 통신할 수 있는 네트워크 자원의 손실이 발생하게 된다. 이 손실이 많아지면 정상적으로 수행하던 서비스를 제공할 수 없는 상태에 이르게 된다.

본 논문에서는 컴퓨팅 노드가 DoS 공격을 받더라도 미리 정해진 필수 서비스가 유지되도록 하는것을 목표로 하는 두 단계의 대응 방안을 제시하고자 한다. 각 컴퓨팅 노드 내에서는 침입의 결과로 고갈된 자원 환경에 최대한도로 적응력을 발휘하여 필수 서비스가 유지되도록 하는 것이다. 각 노드 내에서의 필수 서비스 수행을 위한 생존성 평가를 통해서 간접적으로 침입 발생 여부를 판단할 수 있다. 서비스가 제공되는 과정에서 우선 프로세스가 구동되면 CPU, 네트워킹을 포함한 I/O, 메모리 자원을 사용하게 되는데 각각의 자원 사용량에 대한 범위를 설정해 놓고 임계값을 초과하였는가에 따라 정상적인 경우와 그렇지 않은 경우를 판단할 수 있다. 한 노드 내에서 더 이상 필수 서비스 유지가 불가능해지면 다른 중복된 컴퓨팅 노드에서 해당 서비스가 사용자에게 투명하게 계속될 수 있도록 중복성을 적용한다. 이 때, 서버들 간의 공통 취약점을 제거하기 위하여 컴퓨팅 플랫폼과 소프트웨어 설계 및 구현의 다양성을 적용한다. 단순한 복제로 서비스 가용도를 향상시킬 수는 있으나 그렇게 하면 한가지의 성공한 공격에 중복된 서버들이 모두 무방비 상태가 되기 때문이다. 제시된 방안이 기존의 연구와 다른 점은 자원 재할당 기법과 중복된 서버 활용을 단계적으로 연동시킴으로써 비교적 적은 비용을 추가해서 매우 효과적으로 침입에 대응할 수 있다는 것이다.

본 논문의 2장에서는 관련된 기존 연구 결과를 요약하고 제안 기법과의 차별성에 대하여 설명한다. 3장에서는 우선 필수 서비스를 선택하고 컴퓨팅 노드 내에서 자원을 재할당하여 침입이 발생한 후에도 필수 서비스가 유지될 있도록 하는 기법을 제시한다. 4장에서는 노드 내에서의 조치에도 불구하고 필수 서비스의 생존성이 확보되지 않는 경우에 대비하여 서버를 중복시켜서 구성하는 방안을 제시한다. 이러한 기법이 적용된 테스트베드의 구현 및 분석 결과에 대하여 5장에서 논하고 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

침입 감내와 관련된 대표적인 연구 결과로 EC(European Commission)의 IST(Information Security Technologies) 프로그램에서 우연히 발생한 고장이나 악의적인 공격에 대하여 중대한 정보 인프라를 보호하려는 노력을 들 수 있다. DEPPY(European Dependability Initiative의 줄임말)는 의존성 지원 기술을 증진시켜서 시스템과 서비스의 신뢰와 확신을 높이는데 기여하기 위한 활동을 해왔다[2, 3]. 또 다른 연구 결과로 미국 국방성의 DARPA 연구를 들 수 있다[5, 6]. 미국의 중대한 정보 인프라가 의존하는 네트워크 정보 시스템의 일반적인 취약점의 원인을 도출하고 취약점에 대응하는 보안 기술들을 자원 할당, 복제, 재구성, 속임과 같은 기술들로 분류하고 있다.

[5]에서 동적 자원 할당의 기본적인 개념이 소개되었다. 동적 자원 할당은 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. 그러나, 아직까지 기본적인 개념만 제시되었을 뿐이고 공격에 대비한 구체적인 자원 할당 알고리즘이나 구현 방안에 대해서는 소개되지 않았다.

DARPA의 지원을 받아 진행되고 있는 최근 연구 결과 중에는 공격에 대비하여 서버들의 중복을 이용하는 구조에 대한 결과가 있다. 그 예로 UC Davis의 HACQIT(Hierarchical Adaptive Control of Quality of service for Intrusion Tolerance)[7] 프로젝트의 연구 결과를 들 수 있다. 이 프로젝트는 사용자에 대한 종합적인 성능의 25% 이상의 저하를 방지하면서 네 시간 동안의 침입 감내 제공을 목표로 하고 있다. 시스템 내에는 두 대의 서버, 즉, 프라이머리와 백업이 존재한다. 이들을 모니터하고 제어하는 제어기는 결함 진단 소프트웨어를 포함하고 있다. 이 제어기는 서버의 데이터 복제를 가지고 있어서 두 서버의 결과가 일치하지 않는 경우 문제를 파악하는데 이용된다. 여기서 사용하는 복제 및 비교 방법은 전통적인 하드웨어 오류 검출 방식이다. 같은 입력이 서로 같은 컴포넌트에 보내지고 출력을 비교한다. 단지 출력이 서로 다르면 오류가 검출된 것으로 간주하고, 침입 행위가 탐지된 경우 제어기는 해당 서버를 차단한다. 이 구조는 간단하기 때문에 일반 COTS 서버들로 비교적 쉽게 구현할 수 있다. 다만 침입을 탐지하는 기능이 미약하고 확장하는데 한계가 있는 단점이 있다. 또 다른 예로 SITAR(Scalable Intrusion-Tolerance Architecture)[8]를 들 수 있다. MCNC사와 Duke 대학에서 연구를 진행하고 있는데, 분산 서비스, 특히 COTS 서버를 위한 침입 감내의 구조를 제시하고 있다. 이 구조는 플락시 서버, 수용성 검사 모니터, 선출 모니터, 적응 재구성 모듈, 감사 제어기로 구성된다. 플락시 서버는 침입 감내 전략에 의해 규정된 서비스 정책을 결정한다. 서버 결과의 불일치가 발생했을 때 각각의 COTS 서버의 대리인 역할을 하는 것이 선출 모니터이다. 적응 재구성 모듈은 침입 발생 정보를 받아 이를 평가하고 성능 영향을 분석하며, 시스템의 새로운 구성을 생성한다. 감사 제어기는 주기적인 진단을 시행하여 비정상적인 행위를 파악하고 기록한다. 수용성 검사 모니터

는 외부 공격으로 인한 영향을 탐지하고 보고하는 중요한 역할을 담당한다. 수용성 모니터가 침입이나 오류를 검출하면 침입 경보를 발생한다. 이 방안의 장점은 기존 COTS 서버를 바꾸지 않고 그대로 적용 가능하고 사용자에게도 투명하다는 것이다. 그러나 동시에 대량의 COTS 서버 공격에 대응하는데 한계와 COTS 서버 이외의 구성 요소들은 공격의 취약성을 갖고 있지 않아야 한다는 부담이 있다. 침입 대응 과정을 담당하는 기능이 분산되어 있어서 과도한 지연 가능성이 있다. 비교적 구성이 복잡하고 동적 구성에 따른 구현 비용이 많이 드는 단점이 있다.

본 논문에서 제안하는 방안은 앞서 설명한 자원 재할당 방법과 서버 중복을 이용하는 방법을 단계적으로 적용하여 효과적인 공격 대응이 되도록 하는 것이다. 이 연구 결과는 자원 재할당 방법을 DoS 공격에 대비할 수 있는 구체적인 알고리즘으로 제시하였다는 점과 이와 연계하여 실현 가능한 효과적인 서버 중복 구조를 제시하였다는 점을 강조할 수 있다. 공격으로 인하여 바뀐 환경에 적응할 수 있게 하고 한 가지 공격에 중복된 여러 노드들이 같이 공격당하지 않도록 다양성을 적용하는 개념을 바탕으로 한다.

3. 노드 내의 자원 재할당

이 장에서는 한 컴퓨팅 노드가 침입을 당한 경우에도 필수 서비스를 유지할 수 있도록 하는 자원 재할당 기법에 대하여 논한다.

노드 내에서의 자원 재할당이란 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 서비스가 생존하기 위해서는 일정 수준 이상의 CPU, 메모리, 네트워크 자원이 필요한데, 그 수준이 어느 정도인지 단정적으로 말하기는 힘들다. 여러 태스크들이 동시에 수행되고 있는 서버에서 특정 서비스만을 위해서 모든 자원을 할당해주면 서버의 처리량(throughput)이 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. 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 결정한다. 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 임계값으로 사용한다.

기준선과 기준선 초과 누적 시간은 각각의 필수 서비스 뿐 아니라 전체 서비스의 사용량 합계에 대해서도 설정할 수 있다. 예를 들어, CPU 사용량이 최근 10초 동안 90% 이상을 유지한 시간이 6초 이상을 초과하였다면 CPU가 매우 바쁘게 동작하고 있고 여유 자원이 없다는 것으로 볼 수 있다. 이때, 90%가 전체 서비스에 대한 기준선이고 6초가 기준선 초과 누적시간이다.

본 논문에서 제안하는 노드 내 자원 재할당 알고리즘을 (그림 1)에 나타내었다. 알고리즘을 설명하기 위하여 그림에서 흐름을 나타내는 각 화살표를 번호로 표기하였다. 각 화살표 해당 번호에 대한 알고리즘 수행 내용은 다음과 같다.

① 자원 사용량을 측정한 결과 전체 서비스의 자원 사용량이 설정된 기준선의 초과 누적시간을 넘지 않는 경우에는 아직 충분한 여유 자원이 남아 있어서 필수 서비스를 수행하는데 아무런 문제가 없는 것으로 보고 모니터링을 지속한다. 전체 서비스의 자원 사용량에 대한 설정 기준은 전체에서 필수 서비스를 수행하는데 필요한 최소한의 자원량을 뺀 것이다.

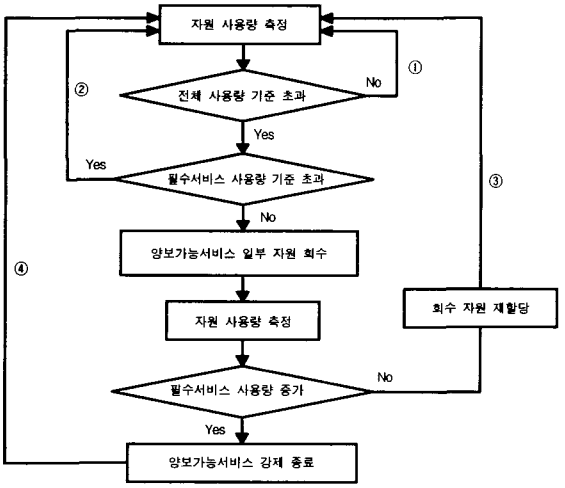
② 전체 사용량이 기준을 초과해서 여유 자원이 얼마 남지 않게 되는 경우는 그 원인을 크게 세 가지로 생각할 수 있다. 첫 번째로, 자원의 대부분을 필수 서비스가 차지하고 있는 것이다. 비록 여분의 자원은 없지만 필수 서비스가 사용하고 있음으로 문제가 없다. 여기서 필수 서비스가 자원을 사용하고 있다는 판단은 필수 서비스의 자원 점유율이 필수 서비스 사용량 기준을 초과한 것으로 할 수 있다.

③ 그렇지 않으면, 두 번째의 경우로, 자원을 양보 가능한 서비스가 대부분 차지하고 있어서 전체 사용량이 기준을 초과, 즉 여유 자원이 얼마 남지 않게 된 것이다. 이를 확인하기 위해서 자원을 차지하고 있는 양보 가능한 서비스의 사용 우선 순위를 인위적으로 낮추어 일부 자원이 돌아오도록 한다. 우선 순위를 낮추어도 필수 서비스 사용량이 크게 증가하지 않으면 필수 서비스는 현재 대기 상태임을 의미하므로 양보 가능한 서비스의 우선 순위를 복귀시키고 경과를 두고 본다.

④ 그러나, 세 번째의 경우로, 양보 가능한 서비스의 자원 점유 우선 순위를 낮추었더니 필수 서비스 점유율이 크게 증가한 경우, 즉, 자원을 양보 가능한 서비스와 필수 서비스가 경쟁적으로 점유하려고 하는 경우에는, 필수 서비스가 자원을 필요로 하고 있음에도 불구하고 그 동안 양보 서비스에 의해 자원을 할당받지 못했다고 판단하여 해당 양보 가능 서비스의 자원을 반납하도록 종료시킨다.

이 알고리즘으로 노드 내에서 필수 서비스가 필요로 하

는 자원을 최대한 확보할 수 있다. 기본적인 목적이 노드에 대한 외부 공격에도 불구하고 사용자에게 지속적인 필수 서비스를 제공할 수 있도록 하는 것이다. 하나의 필수 서비스를 제공하기 위해서는 많은 경우에 그 서비스가 의존하는 다른 서비스들도 보호되도록 하여야 한다. 즉, 따라서 하나의 필수 서비스가 서버 관리자에 의해 선택되면 그 서비스가 의존하는 다른 모든 서비스들도 필수 서비스로 선택되도록 한다.



(그림 1) 노드 내 자원 재할당 알고리즘

기준선과 기준선 초과 누적 시간은 서비스 유형에 따라 다르게 결정되어야 한다. 정상적인 수행 시간 동안의 모니터 자료를 토대로 도출이 가능하다. 이 자원 재할당 기법의 설계에는 한 가지 가정이 필요한데, 그것은 침입자가 활동하는 프로세스가 필수 서비스에는 접근하지 못하도록 해야 한다는 것이다. 즉 보호하려고 하는 필수 서비스에는 침입이 이루어지지 않는다는 것이다. 이것이 충족되지 않으면 결국 자원 재할당에도 불구하고 침입 당한 노드는 필수 서비스를 제대로 유지할 수 없게 될 것이다. 이 때에는 필수 서비스가 다른 노드에 의해서 사용자에게는 투명하게 계속해서 제공될 수 있도록 노드간의 협조가 필요하게 된다. 즉, 서버를 다른 노드에 중복해 둠으로써 대비할 수 있다.

4. 서버 중복 적용 방안

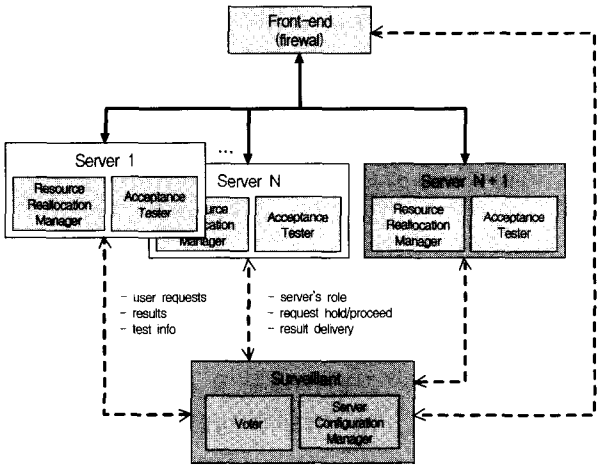
노드 내에서의 자원 재할당으로도 필수 서비스의 품질을 유지하는데 필요한 자원이 확보되지 않으면 다른 노드에서 필수 서비스 제공이 유지되도록 한다. 이 장에서는 이를 위하여 서버 중복을 이용하는 방안에 대하여 논한다.

일반적인 중복 방법은 중복 노드의 상태에 따라 크게 두 가지로 나눌 수 있다. 하나는 능동 중복이고 다른 하나는 수동 중복이다. 능동 중복에서는 항상 능동 상태에 있는 노드 집단이 동시에 사용자의 서비스 요청을 처리한다. 그리고 일정 비율 이상의 같은 응답이 오면 이를 유효한 것으로 선출한다. 얼마나 많은 중복이 필요한가는 결함의 유형과 일치성 보장 조건에 달렸다. 선출된 응답과 다른 결과를 산출한 노드들의 복구를 위해서 서비스 요청을 시작하기 전에 저장해 두었던 안전한 상태로 돌아가 재시도를 하도록 할 수도 있다. 수동 중복에서는 한 노드만 능동 상태이고 중복된 노드들은 모두 수동 상태에 있다. 능동 상태에 있는 노드는 프라이머리가 되고 나머지 노드는 백업이다. 각 노드는 수행 결과에 대해서 자체적으로 수용성 시험을 한다. 프라이머리가 수용성 시험을 실패하면 백업 중에서 선택된 새로운 프라이머리가 재시도 한다. 각 노드가 상태값을 유지하는(stateful) 서버라면 성공한 프라이머리는 백업들에게 결과 메시지를 보내서 백업들이 이 사실을 인지하고 자신들의 복제된 데이터를 갱신하게 한다. 이렇게 하면 재시도에 필요한 시간을 줄일 수 있다. 수동 중복은 백업의 상태에 따라 항상 재시도가 가능한(hot standby) 상태 또는 백업이 재시도를 위해서 별도의 준비 과정이 요구되는(cold standby) 상태로 구분할 수 있다.

본 논문에서는 수용성 시험과 선출 방법을 같이 이용하는 N+1 중복 구조를 제시한다. 제시된 중복 방안은 N개의 능동 노드와 항상 재시도가 가능한(hot standby) 상태의 1개의 백업 노드로 구성된다. 일반 사용자는 접근할 수 없는 별도의 네트워크를 통해 연결된 감독자(Surveillant)가 이들을 관장한다. 백업 노드가 침입 발생시 재구성에 대비하고 있어서 신속한 재구성이 가능할 뿐 아니라 재구성 후에도 계속되는 다른 공격에 대응할 수 있다. 따라서 첫 번째 침입이 발생한 후에는 침입 감내 기능을 유지하지 못하는 문제를 극복할 수 있다. 중복된 노드들의 효율을 제고하기 위하여 능동 노드들이 동기화되어 같이 동작하는 것은 필수 서비스로 제한한다. 즉, 능동 노드들이 동시에 서로 다른 양보 가능한 서비스를 제공할 수 있도록 하여 비용 대비 성능을 높일 수 있다. 수용성 시험 방법과 선출 방법을 같이 사용하여 두 방법의 단점을 상호 보완한다. 기존의 수용성 시험은 크게 두 가지로 이루어진다. 하나는 결과에 대한 논리적 분석을 통한 시험이고 다른 하나는 수행 시간을 측정하여 시간이 경과한 경우 실패로 간주하는 것이다. 논리적 시험에서는 응용의 규격에 적합한지, 결과 값이 타당한 범위 안에 들어가 있는지, 산출된 데이터의 크기가 일치하는지를 시험한다. 그러나 대부분의 경우 이러한 논리 시험의 검출 비율을 높이기 위해서는 응용에 의존할 수밖에 없다. 따라서 좋은 수용성 시험을 개발하는 일반적인 방법을 찾는 것이 큰 문제로 남아 있는 실정이다. 시간을 정하는 문제도 응용과 플랫폼에 따라 달라지기 때문에 어떤 원칙을 따르기 보다는 경험적으로 결정되는 경우가 대부분이다. 결과를 비교 선출하는 경우에는 모든 노드로부터 결과값을 받아서 단순 비교하기 때문에 노드 중에 가장 느린 노드에 의해 처리 시간이 지연되는 문제가 있다. 이러한 문제들을 해결하기 위해 수용성 시험에서 각 노드는 일반적인 또는 시스템적인 방법으로 자체 수용 시험을 하고 능동 노드들의 결과를 선출하는 제

어기는 일부 결과가 도달하지 않아도 일치된 결과를 사용자에게 전달하도록 하여 응답 시간을 줄이도록 한다.

제시된 중복 방안을 적용한 시스템의 구조는 (그림 2)와 같다.



(그림 2) 중복 방안이 적용된 시스템 구조

중복의 개수 N을 정하는 문제는 동시에 몇 대의 노드에 문제가 발생하는 것을 감내할 것인가에 달려있다. Byzantine 일치 알고리즘[9]에 의하면, N이 중복 노드의 개수이고 최대 t개의 노드가 결함 또는 공격에 의해(정상 서버가 동작하는 것을 방해하는 것을 제외한) 임의의 비정상적인 행위를 할 수 있다고 할 때, $3t < N$이다.

N의 2/3 이상이 같은 결과를 가지고 자체 수용 시험을 통과하였다면 나머지 결과에 관계없이 먼저 도래한 결과를 사용자에게 전달할 수 있다. 감독자는 일정 시간이 경과해도 이미 사용자에게 전달된 값과 같은 결과를 출력하지 못하는 노드는 문제가 발생한 것으로 판단하고 시스템에서 차단시킨다.

그림에서 Server 1부터 Server N까지는 능동 상태에 있고 Server N+1은(hot standby) 백업이다. 즉, 실제로 요청을 수행을 하지는 않지만 성공한 다른 서버들의 결과 값을 갖고 있어서 언제든지 능동 상태로 손쉽게 전환될 수 있다. 중요한 역할을 담당하는 것이 감독자이다. 이 노드는 물리적으로 별도의 네트워크로 서버와 연결되어 있어서 일반 사용자들에게는 노출되지 않는다. 동작하는 순서는 다음과 같다.

① 사용자 요청은 프론트엔드를 통해서 서버들에게 전달된다. 이 요청은 서버에 의해 감독자도 전달되고 능동 서버들은 요청을 수행한다.

② 결과에 대하여 각 서버들은 수용성 시험(Acceptance Test)을 실시한 후, 시험 결과를 수행 결과와 함께 감독자의 선출기(Voter)에게 전달한다.

③ 선출기는 결과 비교를 통하여 침입 여부를 판단한다. 침입이 발생하지 않은 경우에는 결과값을 백업 서버에 전달하여 상태를 갱신토록 한다. 침입이 발생한 경우, 해당 서버를 차단하고 수리에 들어간다. 재구성을 위하여 백업 역할을 하였던 서버가 이를 대신한다. 침입이 있었던 서버의 문제가 해결될 때까지 백업이 없는 상태로 서비스가 제공된다. 수리가 완료된 후 백업 서버로 다시 결합한다.

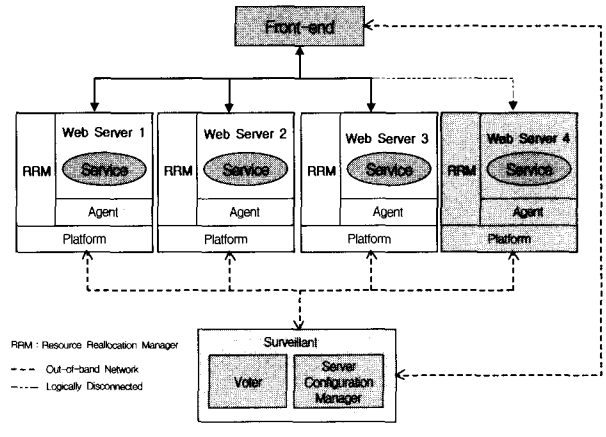
5. 테스트베드의 구현 및 결과

제시된 자원 재할당 및 중복 방안의 타당성을 검증하기 위하여 테스트베드를 구현하였다. 이 장에서는 테스트베드 구현 방법과 분석 결과에 대하여 논한다.

5.1 테스트베드 구현

테스트베드는 인터넷에서 가장 많은 이용하는 HTTP 기반의 웹 응용을 대상으로 하였다. 서버 노드의 시스템 환경은 윈도우와 리눅스 플랫폼에 IIS, Apache, Tomcat를 혼용하여 웹 서비스를 할 수 있도록 구성하고 각 웹 서버가 동일한 결과를 내도록 한다. 이는 서버 또는 프로세스들 간의 공통 취약점을 제거하기 위한 것이다. 서버 측에는 단순히 웹 컨텐츠의 표현을 담당하는 HTML 코드와 기능적 서비스를 제공하는 스크립트 코드로 이루어져 있다. 스크립트 코드가 서버 측에서 수행된 후 클라이언트에 대한 응답 결과는 HTML 형태로 전달된다. 이 때, 기능적인 서비스 제공을 위해 서버 측에서 처리되는 스크립트 코드는 클라이언트 세션당 하나의 프로세스 또는 쓰레드 단위로 수행되는데 이러한 스크립트 코드가 서버 자원에 영향을 미칠 수 있다.

테스트베드는 (그림 3)과 같이 4대의 서버 노드와 프론트엔드, 그리고 감독자 노드로 구성된다.



(그림 3) 테스트베드 시스템 구성

5.1.1 자원 재할당 구현

3장에서 설명한 노드내의 자원 재할당은 (그림 3)의 RRM (Resource Reallocaton Manager)에 의해 다루어진다. 응용 객체로 구현한 RRM은 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 값들이 수집되고 수집된 정보를 이용하여 주기적으로 특정 서비스 및 노드 차원에 대한 생존성 평가를 앞서 설명한 알고리즘에 따라 수행한다. 생존성 평가 결과는 RRM이 자원 재할당을 위한 코드 수행 여부를 결정하는 기준이 되며, 생존성 평가 결과에 따라 RRM의 자원 재할당 수행 여부를 결정하는 플래그 값을 설정한다. 생존성 평가 결과로 자원 재할당 수행이 결정되면, RRM은 Registry API를 이용하여 사용자로부터 미리 설정된 양보 가능한 서비스 객체들의 우선순위를 낮추거나 종료시켜 이들 객체들의 가용 자원을 반환하도록 제어한다. 이때 양보 가능한 서비스 객체들의 종료 우선 순위는 자원의 이용도가 높은 순서를 따른다.

5.1.2 서버 중복 구조 구현

4장에서 설명한 중복 구조는 (그림 3)의 프론트엔드(Front-end), 수용성 시험을 포함한 에이전트(Agent), 그리고 감독자(Surveillant)로 이루어진다.

프론트엔드는 클라이언트로부터 요청 메시지를 받아 고유 번호와 도착시간을 부여하여 각 능동 서버 노드에 있는 에이전트에게 보낸다. 동시에 들어오는 많은 요청 메시지를 처리하기 위하여 복수 개의 연결 객체를 풀로 만들어 쓰레드 처리한다. 연결이 이루어진 후 클라이언트로부터 들어오는 HTTP 요구 메시지를 자신의 쓰레드 수행 시간 내에 객체 데이터의 복사본을 각 능동 서버 에이전트로 보낸다. 프론트엔드는 "wrong request repository"라는 일종의 데이터베이스를 관리하는데 이는 정의되지 않은 URL 요구나 감독자로부터 보고되는 평가에서 실패한 요구 메시지 정보를 저장하고 있다. 프론트엔드의 연결 객체는 클라이언트의 요구 메시지를 에이전트로 보내기 전에 이 정보를 근거로 요구 메시지의 전달 여부를 결정한다. 이런 기능은 불필요한 연결 유지를 위한 시스템 전체의 부하를 덜어주는데 기여한다.

에이전트는 프론트엔드로부터 들어오는 요청 메시지를 수신하고, 이를 자신의 웹 서버에게 전달하며, 웹 서버의 서비스 결과에 대한 수용성 시험을 행하는 구성 요소이다. 이를 위해서 에이전트와 웹 서버는 각각 다른 포트 번호를 가지고 있으며 에이전트도 요청 메시지를 웹 서버에게 전달하는 순간 후속으로 도착하는 요청 메시지를 처리하기 위해 복수 개의 연결 객체를 풀로 구성하여 사용한다.

에이전트는 자신이 관리하는 수용성 시험 쓰레드 객체로 각 웹 서버 노드의 수행 결과를 평가한다. 웹 응용이기 때문에 요청의 규격에 따라 미리 예상되는 결과값을 테이블로 만들어 놓고 실제 결과가 이에 해당하는지를 시험하는 방법을 사용한다. 또한, 에이전트는 감독자와의 요구 메시지를 중계하는 중요한 역할을 수행한다. 특히, 클라이언트의 요구 메시지를 웹 서버에 넘길 때, 해당 웹 서버의 동적 웹 서비스 구현을 위해 사용한 스크립트 기술의 차이를 극복한다. 이는 각 웹 서버 노드가 플랫폼, 웹 서버, 서비스 구현 기술에 있어서 다양성을 가지고 있기 때문에, 각 웹 서버 노드에는 같은 컨텐츠가 구성되었더라도 그 URL 확장자가 각각 다르다. 즉, Win32-아파치-JSP, 리눅스-아파치-PHP, Win32-IIS-ASP 등의 구성에 따라 URL 확장자가 다르기 때문에 에이전트가 자신의 웹 서버에게 클라이언트 요구 메시지를 넘길 때는 URL 확장자를 일치시켜주는 작업이 필요하다. 이러한 처리가 끝나면, 에이전트의 연결 객체는 웹 서버에게 객체 데이터의 요청 정보를 전달하고 웹 서버로부터의 응답을 기다린다. 수신된 응답은 다시 감독자에게 전달된다.

감독자는 각 웹 서버의 에이전트가 보내준 결과 데이터를 비교 선출한다. 모든 능동 서버의 결과를 기다리지 않고 일치된 결과가 2개 이상 발견되면 그 값을 자신이 가지고 있는 데이터베이스에 저장하고 프론트엔드에 전달한다. 변경된 데이터 값은 백업 노드에도 전달되어 백업 데이터 갱신이 사용된다.

각 웹 서버 노드의 에이전트는 웹 서버의 응답 결과를 감독자의 지정된 포트로 전송한다. 이 때 감독자의 각 포트는 수신된 응답 결과 데이터를 해쉬 테이블에 저장한다. 감독자의 선출기(Voter) 쓰레드는 각 해쉬 테이블에서 키 값을 참조하여 각 웹 서버별 응답 결과에 대해비교 평가를 수행한다.

SCM(Server Configuration Manager)은 노드간 자원 재할당 관리자로서 노드 차원의 자원 재할당 요구가 발생했을 때 서버 노드의 재구성을 관리하는 역할을 수행한다. 이 요구는 각 서버 노드의 RRM과 선출기에 의해 작동된다. 각 능동 노드 RRM에 의해 주기적으로 보고가 전달되지 않을 때, 그리고 특정 능동 노드의 결과값이 일치하지 않거나 일정 시간이 경과할 때까지 결과값이 도달하지 않을 때에 프로세스가 구동된다. 재구성에 들어간 노드는 N+1 노드(백업 노드)로 물러나며, 원래의 N+1 노드는 재구성에 들어간 노드를 대체하는 새로운 멤버 노드가 된다. SCM에 의해 재구성에 들어간 노드가 N+1 노드로 재부팅 되는 과정에서 한 가지 해결해야 문제는 다양성을 갖는 각 웹 서버 노드의 에이전트를 로딩하고 초기화하는 일이다. 이 문제를 해결하기 위해 본 구현에서 리눅스의 경우는 /etc/rc.d/ rc.local 파일에 실행 파일을 추가하여 재구성(rebooting) 과정에서 OS상에 로딩하도록 하고, Win2K의 경우는 에이전트를 로딩할 수 있는 서비스 프로그램을 등록하도록 하였다. 보호되지 않은 프로세스에 의해 생성된 파일을 삭제하고 변경된 파일은 제어기가 보관하고 있던 원래의 것으로 바꾸어 복구한다.

5.2 실험 및 분석 결과

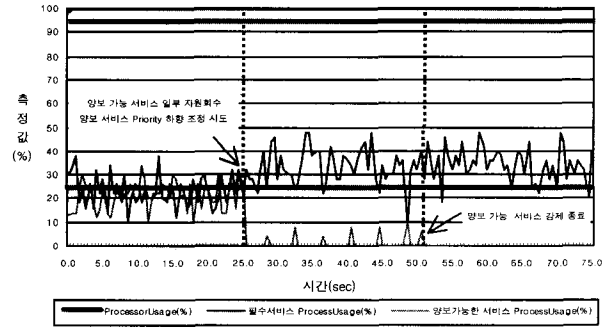
실험에서 생존성 평가 기준으로 노드 차원의 총 CPU 및 메모리 이용도, TCP/IP 연결수와 같은 자원 이용도에 대한 기준선과 이 기준선을 초과한 분당 횟수를 적용하였으며, 보호 대상 서비스 수행 객체에 대해서는 CPU 이용도에 대한 생존성 평가 기준만을 적용하였다. 노드 차원의 총 CPU 이용도란 현재의 순간에서 CPU가 쉬지 않고 일하고 있는 시간에 대한 백분율을 의미한다. 노드 차원의 총 메모리 이용도는

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.

실험을 위하여 이더넷 환경에서 각 서버에 대한 CPU 사용량과 네트워크 인터페이스 및 TCP 설정 수와 같은 자원 이용도 측정 항목에 대한 측정을 하였다.

공격도구로는 인터넷상에서 수십 가지의 도구가 있지만 윈도우 환경에서 사용하는 Divine, PacketBuilder, Nuke, Port Fuck, Elite, ExploitGenerator 등과 같은 도구를 사용하였다. TCP SYN 범람에 대해서는 [10]의 Random Drop과 같은 간단한 방법을 적용하였다.

(그림 4)는 양보 가능 서비스를 강제 종료한 경우의 CPU 사용량의 예를 나타낸 것이다. 필수 서비스의 기준선이 25%로 설정되어 있는데 전반부에 진한 선으로 나타낸 필수 서비스의 CPU 사용량이 이에 미치지 못하게 되어 (그림 1)에 나타낸 알고리즘에 따라 양보 가능한 서비스의 우선 순위를 낮추어 CPU 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 4) RRM 실험 결과

중복성 적용 실험에서는 인위적으로 결함 또는 공격을 삽입한 후 감독자에 의해 재구성이 이루어지는 과정을 살펴보았다. 결함 삽입 방법으로는 특정 서버의 스크립트 파일을 미리 엉뚱한 내용으로 바꾸어 놓고 해당 파일이 요청되게 하였다. 공격 삽입은 과도하게 많은 자원을 차지하는 비정상적인 영상처리 응용을 의도적으로 필수 서비스로 설정해두어 임의의 시간에 응용이 동작하게 하였다.

이와 같은 방법으로 실험을 실시한 결과 수용성 검사와 선출기에 의해 검출이 되었고 이에 따라 노드 재구성이 이루어졌다. 검출 지연 시간은 선출기가 허용하는 응답 지연 시간 파라미터에 의해 크게 좌우되는데, 전체 처리량을 높일 목적으로 각 서버가 필수 서비스 외에 서로 다른 비 필수서비스를 동시에 수행하게 하는 것이 불리하게 작용하였다. 즉, 전체 처리량을 높이기 위하여 비 필수 서비스를 수행하는 데에 따른 정상적인 필수 서비스 응답 지연 시간에 비례하여 검출이 지연된다. 재구성에 소요되는 시간은 백업 노드의 부팅 시간과 데이터 복제 시간에 좌우되었다.

이 실험으로 본 논문에서 제시한 자원 재할당 알고리즘과 서버 중복 구조가 타당한 것으로 판단할 수 있다. 중복 구조를 기존의 다른 연구 결과와 비교하면 (그림 5)와 같다.

구 분	제시된 방법	HACQIT	SITAR
기본 복제 접근 방법	N + 1	primary & backup	multiple replica (single point of failure 배제)
침입 탐지 기법	수용성 검사 및 비교	단순 비교	수용성 검사 및 비교
시스템 재구성으로 연쇄적 침입 대응	가 능	불가능	가 능
제어 장치 연결	별도 네트워크 사용	별도 네트워크 사용	대칭적인 내부 네트워크
구현 비용	중 가	저 가	고 가
처리 복잡도	간 단	간 단	복 잡

(그림 5) 서버 중복 구현 결과 비교

타 연구 결과에 대한 수치 자료가 없어서 정확한 비교는 불가능하지만 본 연구에서 제시하는 방안이 HACQIT의 연구 결과에 비하여 정확한 탐지가 가능하여 탐지 실패율이나 허위 탐지 비율을 줄일 수 있다. 서버의 개수가 추가되고 수용성 시험을 하는데 따른 추가 비용이 들어간다. SITAR는 모든 구성 요소들을 매트릭스의 형태로 중복하여 실제 구현하기에는 매우 복잡하고 많은 지연 시간이 예상된다. 다만 본 연구 결과에서 드러난 프론트엔드의 취약성을 보완하는데 도움이 될 것으로 보인다.

본 논문에서 제시된 자원 재할당 기법과 서버 중복 기법이 실제 시스템에 적용되기 위해서는 앞으로 해결해야 할 문제점들이 남아 있다. 노드 내 자원 재할당 알고리즘에 있어서 각 필수 서비스에 대한 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. 또한 전체 처리량을 높이기 위한 방안들과 검출 지연 및 재구성 소요시간을 줄이기 위한 방안들 사이의 균형(trade-off)이 필요하다.

6. 결 론

본 논문에서는 새로운 연구 이슈가 되고 있는 침입 감내 시스템 구축을 위한 중요한 기술인 자원 재할당 알고리즘을 구체적으로 제시하였다. 또한 생존성을 보다 향상시키기 위하여 노드간의 자원 재할당 및 시스템 재구성이 이루어지도록 하는 서버 중복 구조를 제시하였다.

자원 재할당을 통하여 침입 감내 기능을 갖고 궁극적으

로 시스템의 생존성을 보강할 수 있는 시스템 개발을 목표로 하였다. 우선 필수 서비스를 선택하고 노드 내에서 자원을 재할당하여 침입이 발생한 후에도 필수 서비스가 생존할 수 있도록 하였다. 이러한 노드 내에서의 조치에도 불구하고 필수 서비스의 생존성이 확보되지 않으면 다른 서버 노드로 서비스 제공이 전환될 수 있도록 하는 노드간의 자원 재할당이 이루어지도록 설계하였다. 이 기법의 타당성을 검토하기 위하여 테스트베드를 구축하여 CPU와 네트워크 사용량에 대하여 재할당 기법을 실험하였다. 플락시를 이용하여 클러스터 형태로 연결되어 있는 아키텍처에서 중복성과 복제의 다양성을 적용한 사례 연구에 중점을 두고 장단점을 논하였다. 실험 결과 본 논문에서 제시한 기법은 기존의 다른 방안에 비하여 비교적 적은 추가 비용으로 실패율이나 허위 탐지율을 줄일 수 있어서 효과적인 침입 감내 시스템 구축 기술의 하나가 될 수 있다는 가능성을 보였다.

인터넷 응용에서 흔히 볼 수 있는 자원을 고갈시키는 DoS 공격, 파일이나 주소를 노출시키는 기밀성 공격, 데이터 내용을 임의로 변경시키는 완전성 공격 중에서 DoS 공격과 완전성 공격에 매우 유력할 것이다. 이 기법이 실제 시스템에 적용되기 위해서는 서비스의 자원 이용 유형을 분석 적용하고, 구현상의 문제 등에 대한 보완 연구가 진행되어야 할 것이다.

참 고 문 헌

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

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

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

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

[5] 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.

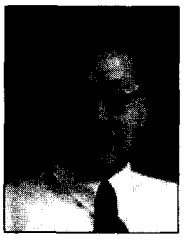
[6] Matti A. Hiltunen, et. al, "Survivability through Customization and Adaptability : The Cactus Approach," DARPA Information Survivability Conference & EXposition, 2000.

[7] Reynolds, J. et. al, "The Design and Implementation of an Intrusion Tolerant System," Proc. of Int'l Conference on Dependable Systems and Networks, 2002.

[8] Wang F., et. al, "SITAR : A Scalable Intrusion-Tolerant Architecture for Distributed Services," DARPA Information Survivability Conference & EXposition, 2001.

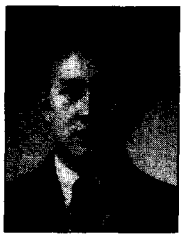
[9] Marshall Pease, Robert Shostak, Leslie Lamport, "Reaching Agreement in the Presence of Faults," Journal of the ACM 27/2, pp.228-234, 1980.

[10] Sun Microsystems, Sun's tcp syn flooding solutions, http://ciac.llnl.gov/ciac/bulletins/h-02.html.



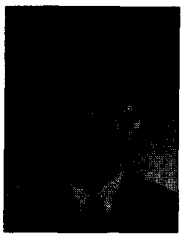
민 병 준

e-mail : bjmin@incheon.ac.kr
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
관심분야 : 분산시스템, 통신망관리, 보안



김 성 기

e-mail : proteras@incheon.ac.kr
1996년 인천대학교 컴퓨터공학과(학사)
1998년 인천대학교 컴퓨터공학과(석사)
1998년~1999년 인천대학교 멀티미디어 연구센터 연구원
2000년~현재 인천대학교 컴퓨터공학과 박사과정
관심분야 : 분산시스템, 실시간시스템, 결함허용, 침입감내

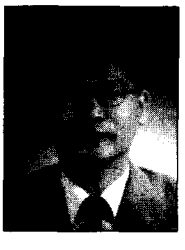


나 용 희

e-mail : ramen@incheon.ac.kr
2002년 인천대학교 컴퓨터공학과(학사)
2000년~현재 인천대학교 컴퓨터공학과 석사과정
관심분야 : 분산시스템, 침입감내

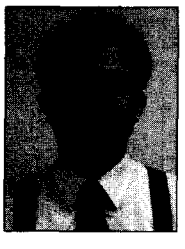
이 호 재

e-mail : leehj@kisa.or.kr
1999년 성균관대학교 정보공학과(학사)
2002년 성균관대학교 전기전자컴퓨터공학과(석사)
2002년~현재 한국정보보호진흥원 연구원
관심분야 : 컴퓨터보안, 침입감내 시스템, 무선보안



최 중 섭

e-mail : jschoi@kisa.or.kr
1993년 인천대학교 전자계산학과(학사)
1995년 숭실대학교 컴퓨터학과(석사)
2000년 숭실대학교 컴퓨터학과(박사)
1995년~1996년 한국전산원 연구원
2000년~현재 한국정보보호진흥원 선임연구원
관심분야 : 컴퓨터시스템보안, 실시간시스템, 분산시스템



김 홍 근

e-mail : hgkim@kisa.or.kr
1985년 서울대학교 컴퓨터공학과(학사)
1987년 서울대학교 컴퓨터공학과(석사)
1994년 서울대학교 컴퓨터공학과(박사)
1994년~1996년 한국전산원 선임연구원 전산망안전보안센터장
1996년~현재 한국정보보호진흥원 책임연구원 기술단장
관심분야 : 컴퓨터보안, 병렬 알고리즘