2003년 대한전자공학회 하계종합학술대회 제26권 제1호

다양성과 중복성을 이용한 웹 서비스 침입감내 시스템 구현

김 성 기*, 나 용 희, 민 병 준
인천대학교 컴퓨터공학과
e-mail : proteras@incheon.ac.kr

Implementation of a Web Service Intrusion Tolerance System based on Diversity and Redundancy

Sung-Ki Kim*, Yong-Hee Na, Byoung-Joon Min
Dept. of Computer Science & Engineering, University of Incheon
e-mail : proteras@@incheon.ac.kr

Abstract

The intrusions appears continuously by new unknown attacks exploiting vulnerabilities of systems or components but there are no perfect solutions to protect this unknown attacks. To overcome this problem, in this paper, we have proposed and implemented a Web service intrusion tolerant system that provides continuous Web services to the end users transparently even after the occurrence of an attack against the Web services, and prevents the disclosure of system's configuration data from server. Our system has an N+1 node architecture which is to minimize the number of redundant server nodes and to tolerate the intrusion effectively, and it also supports diversity in its design.

Experimental result obtained on an implemented system show that our system can cope with intrusion such as DoS, file modification, confidentiality compromise of system properly.

I. 서론

전자상거래 및 인터넷을 이용한 정보 활용 측면에서 웹 서비스는 오늘날 가장 많이 이용하는 정보 서비스이며, 기존의 다른 정보 서비스도 웹으로 통합되어가고 있는 추세이다. 따라서 웹 서비스에 대한 새로운 침입 시도가 끊이지 않고 있다는 점을 감안할 때, 웹 서비스의 생존성과 기밀성, 데이터의 무결성을 보호하는 대응 방안이 요구된다. 이를 위해 본 논문에서는 다음과 같은 침입 감내를 위한 일반적인 대응 과정을 체계적으로 실현시킬 수 있도록 시스템 구조를 대표적인 결함 허용 접근 방법의 하나인 다양성과 중복성을 적용하여 설계한다.

첫 째, 서버의 자원을 고갈 시키는 DoS(Denial of Service) 공격에 대비하기 위해서, 웹 서비스 시스템이 DoS 공격을 받더라도 정상적인 사용자에게는 투명하게 웹 서비스가 유지되도록 두 단계의 대응 방안을 제시 한다. 각 컴퓨팅 노드 내에서는 침입의 결과로 고갈된 자원 환경에 최대한도로 적응력을 발휘하여 웹 서비스가 유지되도록 하는 것이다. 각 노드 내에서의 웹 서비스 수행을 위한 생존성 평가를 통해서 간접적으로 침입 발생 여부를 판단할 수 있다. 서비스가 제공되는 과정에서 먼저 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 웹 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. 이러한 단계적인 노드

내의 자원 재할당과 서버 노드의 중복 활용은 비교적 저렴한 비용으로 매우 효과적으로 침입에 대응할 수 있게 한다.

두 번째, 기밀성 유지를 위해서, 다양성을 적용한 중복 서버의 결과를 선출하고 사용자에게 응답하는 과정에서 사전에 저장된 루트 권한 정보와 서버의 구성 정보를 응답 결과와 비교 검사하여 중요한 정보의 외부 노출을 막도록 한다.

세 번째, HTML, 이미지, 스크립트 코드 파일, 데이터베이스 내용 정보와 같은 서버가 유지하는 데이터의 무결성을 유지하기 위해, 중복 서버의 결과를 비교 선출하는 과정에서 이들 데이터의 무결성을 검사하고 복구 한다.

본 논문에서는 중복을 최소화 하면서 이상의 설계 목표를 충족시키기 위해 N+1 노드 중복구조를 제시하고 설계와 구현을 하였다.

II. 웹 서비스 침입 형태 및 증세

기존의 웹 서버 제품 및 시스템에 대한 취약성 보고는 제품의 종류 및 컴포넌트 수만큼 다양하지만, 대표적인 웹 서버의 취약성을 이용한 침입 증세를 분류하면 다음과 같다.

① path revealing, directory listing, file disclosure, IP address disclosure 과 같이 서버의 응답 결과가 클라이언트에게 노출 되는 기밀성 위협
② 웹 서버의 자원을 고갈시키는 DoS 공격 의 결과
③ file creation, file deletion, file modification 등과 같은 서버 시스템의 데이터 무결성 위협

III. 설계

3.1 서버 중복 방법

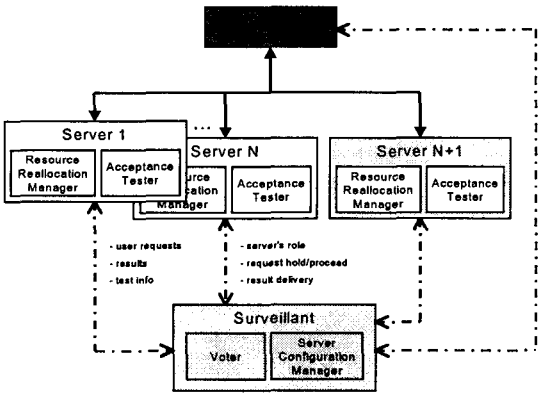
침입에 대비해 웹 서비스 수행에 필요한 자원은 서버 노드 내에서 사전에 설정하여 식별 될 수 있도록 하고, 침입에 의해 웹 서비스의 생존성이 위협 받을 때에는 노드 내에서 가용 자원을 재할당하도록 제어한다. 이를 위해서, 사전에 반드시 보호 되어야 할 필수 서비스와 이들을 위해 점유한 자원을 양보할 수 있는 서비스를 구분한다. 그리고 이들 각 서비스에 대한 자원 사용량에 대한 범위를 설정해 놓고 임계값을 초과하였는가에 따라 정상적인 경우와 그렇지 않은 경우를 판단한다. 만일, 침입에 의해 서비스를 수행중인 서버 노드 내에서 CPU, 메모리, 통신 자원이 고갈하고, 그 결과 필수 서비스의 생존성을 위협할 때는 노드 내에서 자원을 재할당하는 제어를 수행한다. 그럼에도 불구하고, 한 노드 내에서 더 이상 웹 서비스의 품질을 유지하는데 필요한 자원이 확보되지 않으면 다른 중복된 컴퓨팅 노드에서 해당 서비스가 유지되도록 중복성을 적용한다. 이러한 기능의 시스템 구조가 되기 위해서는 이에 효과적인 중복 방안이 연구되어야 한다.

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 웹 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 제어기는 일부 결과가 도달하지 않아도 일치된 결과를 사용자에게 전달하도록 하여 응답 시간을 줄이도록 하였다.

3.2 시스템 구조 설계

제시된 중복 방안을 적용한 시스템의 구조는 (그림 1)

과 같다.



(그림 1) 다양성과 중복성을 적용한 시스템 구조

N 개의 노드는 모두 공통의 취약점을 제거 하기 위해 다양성에 기반 둔 중복이다. 이를 위해 각 노드마다 각각 다른 OS와 소프트웨어 컴포넌트로 구성 된다. 중복의 개수 N을 정하는 문제는 동시에 몇 대의 노드에 문제가 발생하는 것을 감내할 것인가에 달려있다. Byzantine 일치 알고리즘[4]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, 필요한 중복 노드 수 조건은 $3t < N$ 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.

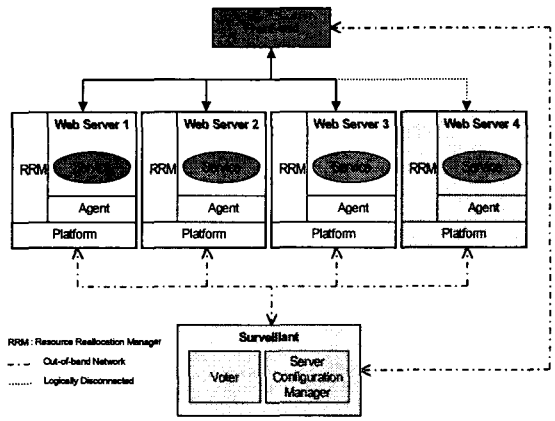
(그림 1)에서 Server 1 부터 Server N 까지는 능동 상태에 있고 Server N+1은 (hot standby) 백업이다. 즉, 실제로 요청을 수행을 하지는 않지만 성공한 다른 서버들의 결과 값을 갖고 있어서 언제든지 능동 상태로 손쉽게 전환될 수 있다. 중요한 역할을 담당하는 것이 감독자이다. 이 노드는 물리적으로 별도의 네트워크로 서버와 연결되어 있어서 일반 사용자들에게는 노출되지 않는다. 동작하는 순서는 다음과 같다.

먼저, 사용자 요청은 프론트엔드를 통해서 각 서버들에게 똑같이 전달되며, 이 요청은 서버에 의해 감독자도 전달되고 능동 서버들은 이 요청을 수행한다. 그 다음, 결과에 대하여 각 서버들은 수용성 시험 (Acceptance Test)을 실시한 후, 시험 결과와 수행 결과를 하나의 메시지 형태로 감독자의 선출기(Voter)에게 전달한다. 마지막으로 선출기는 결과 비교를 통하여 침입 여부를 판단하며, 침입이 발생하지 않은 경우에는 결과 값을 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. 복구가 완료된 후 백업 서버로 다시 시스템에 결합한다.

Ⅳ. 구현 결과

4.1 중복성 적용 구조와 구성요소

중복성을 적용하여 구현한 웹 서비스 침입 감내 시스템의 구조는 다음과 (그림 2)와 같다.



(그림 2) 구현 시스템

(1) 프론트 엔드 구현

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 에이진트로 보낸다.

(2) 에이전트 구현

프론트엔드로부터 들어오는 요청 메시지를 수신하고, 이를 자신의 웹 서버에게 전달하며, 웹 서버의 서비스 결과에 대한 수용성 시험을 행하는 구성 요소이다. 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.

(3) 감독자 구현

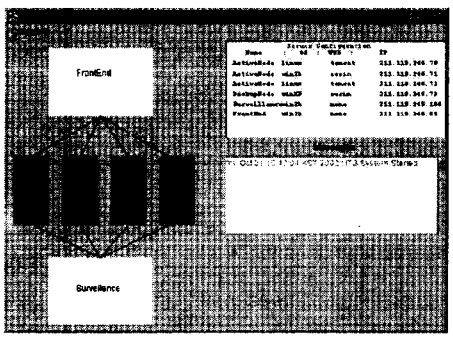
감독자는 각 웹 서버의 에이전트가 보내준 결과 데이터를 비교 선출한다. 모든 능동 서버의 결과를 기다리지 않고 일치된 결과가 2개 이상 발견되면 그 값을 자신이 가지고 있는 데이터베이스에 저장하고 프론트엔드에 전달한다. 변경된 데이터 값은 백업 노드에도 전달되어 백업 데이터 갱신이 사용된다.

(4) 노드내 자원 재할당 관리자 구현

다양성에 기반을 둔 각 플랫폼의 Native API를 이용하여 필수 서비스를 비롯한 기타 양보 가능한 서비스의 자원 이용량을 감시하고, 노드 차원의 자원 재할당을 수행한다. 아울러 서버 구성 관리자(SCM)와 통신을 유지한다.

(5) 서버 구성 관리자 구현

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. 다음 (그림 3)은 시스템 운용 상태를 감시하는 화면이다.



(그림 3) 시스템 운용 상태 감시 화면

4.2 실험

구현된 시스템에 대하여, 서버 노드 내에자원 고갈, 파일 수정과 같은 침입 증세를 유발하는 코드를 내장하고, 구현한 시스템이 이를 적절히 대응하는지 여부와 시스템 재구성 과정에서 생기는 데이터 일치성 보장여부를 조사하였다. 실험한 결과, 각 서버에 구현된 RRM과 SCM의 기능이 DoS와 같은 공격에 필수 서비스의 생존성을 보호하였으며, 에이전트의 수용성 시험 모듈과 감독자의 선출기 모듈의 기능이 임의의 파일 수정 및 비정상적인 정보 노출과 같은 침입에 대해 대응하였다. 아울러, 서버 재구성 과정에서 생기는 데이터 일치성 보장 조건을 감독자의 선출기 모듈 기능에서 만족시켰다.

V. 결론

본 논문에서는 공통된 취약성을 제거하기 위해 다양성과 중복성에 기반을 둔 웹 서비스 침입 감내 시스템을 설계하고 구현하였다. 구현된 시스템의 타당성을 검토하기 위하여 침입 감내 평가 실험하였다. 실험 결과, 비교적 적은 추가 비용으로 효과적인 침입 감내 시스템 구축 기술의 하나가 될 수 있다는 가능성을 보였다. 인터넷 응용에서 흔히 볼 수 있는 자원을 고갈시키는 DoS 공격, 파일이나 시스템 정보를 노출시키는 기밀성 공격에 매우 유력할 것이다.

참고문헌

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

[2] 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.

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

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