Cross-Site Scripting(XSS) 프로세스 진단을 기반으로 한 웹 해킹 대응절차 모델 연구

노시춘*

요 약

웹 해킹 대책을 적용할 경우 어느 한가지 기술이나 방법보다 통합된 대책을 구성하고 정보보안을 단계적으로 실행하는 방법이 필요하다. 웹 해킹 대응을 어느 한가지 방법에만 의존 시 많은 보안 공백을 발생 시킬 수 있다. 본 연구에서는 Cross-site scripting 공격 프로세스를 진단하여 웹 해킹 대응절차를 설계한다. 대응체계는 프레임워크를 구성하고 정보보안을 단계적으로 실행하는 방법이다. 단계적 대응모델은 대책의 구조를 시스템 설계단계, 운용단계, 사용단계로 구성하는 방법이다. 시스템 설계단계는 방법은 개발 라이프 사이클에 기초하여 시큐어코딩 설계로 보안 효율성을 높인다. 사용단계에서는 사용자의 보안 이행사항을 체계화한다. 본 방법론을 실무현장에 적용할 경우 현장에서 필요한 종합적인 접근방법론 모델로 활용할 수 있다.

A Study of Web Hacking Response Procedures Model based on Diagnosis Studies for Cross-Site Scripting (XSS)Process

SiChoon Noh*

ABSTRACT

When applying web hacking techniques and methods it needs to configure the integrated step-by-step and run an information security. Web hackings rely upon only one way to respond to any security holes that can cause a lot. In this study the diagnostic process of cross-site scripting attacks and web hacking response procedures are designed. Response system is a framework for configuring and running a step-by-step information security. Step response model of the structure of the system design phase, measures, operational step, the steps in the method used. It is designed to secure efficiency of design phase of the system development life cycle, and combines the way in secure coding. In the use user's step, the security implementation tasks to organize the details. The methodology to be applied to the practice field if necessary, a comprehensive approach in the field can be used as a model methodology.

Key words : Web Hacking; Response Procedures; Diagnosis Studies; Cross-Site Scripting Process

접수일(2013년 12월 1일), 수정일(1차: 2013년 12월 10일), 게재확정일(2013년 12월 11일)

* 남서울대학교 컴퓨터학과

1. 서 론

몇 년 전 우리나라의 해킹 경유지 발생률(좀비 PC 율이) 세계수준을 기록하던 시기가 있었다. 보안사고 취약요인은 여러 가지가 있지만 먼저 초고속 통신망은 최고 수준으로 구비되어 있으며 인터넷 사용률이 높은 환경에서도 사회 도처에 보안 사각 지대가 존재하는데서 연유한다. 또한 웹 해킹(web hacking)의 대처가 미흡한 데서 그 원인이 크다고 보인다. 웹 해킹 종류 중 Cross-Site Scripting (XSS)은 다른 보안위협에 비해 목표 시스템의 환경에 따른 영향을 크게 받으며 쉽게 사용할 수 있고 공격에 성공할 경우 위력이 큰 공격기법 이다. XSS에 대한 대응은 아키텍처 설계, 애플리케이션 로직, 사용자의 사용방법 측면에서 통합 되고 종합화된 방법으로 대처하지 않으면 효과를 달성할 수 없다. 따라서 웹 해킹 대응 기술체계는 SDLC 측면에서 개발단계 보안은 물론, 애플리케이션 작성, 시스템 운영 시의 보안에 대처해야 한다. 이를 위해 XSS 프로세스를 진단하고 위협을 회피하기 위해 대응책을 강구해야 한다. 급변하는 웹 시스템 환경에서 새로운 보안위협에 대해 대응 할 수 있는 보안방법과 보안 기술을 현장의 환경 에 효율적으로 적용시키기 위한 방법 연구는 매우 절실한 과제이다. 본 연구는 이 같은 측면에서 XSS 공격 프로세스 진단을 기반으로 한 웹 해킹 대응체계 모델을 제안한다. 연구순서는 서론, 웹 시스템 환경 진단, XSS 프로세스 진단, 웹 해킹 대응 절차 체계, 결론의 순서이다.

2. 웹 시스템 환경 진단

2.1 웹 시스템 아키텍처

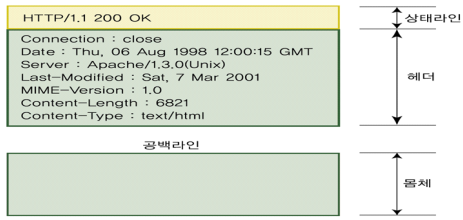
웹 해킹 대응절차 도출을 위해 웹 해킹을 유발 시키는 웹 시스템 구성환경을 살펴본다. 웹 시스 템 구조는 다양하며 본 연구에 인용한 소프트웨어 구조는 윈도우 운영체제 환경에서 스크립트코드 (script code) 언어를 사용, 연동하여 사용자 데이터를 입력하는 방법이다. 이 구조를 인용한 것은 사용자 입력 데이터를 연동하는 방법이 일반적으로 정보시스템 운영 현장에서 가장 많이 사용 하는 사례이기 때문이다. 다음 표는 웹 해킹 공격 과 관련성을 갖는 웹 시스템 환경의 소프트웨어 구조의 사례이다[1][2].

<표1> 웹 시스템 소프트웨어 구조

기 능	소프트웨어	연동
웹 부라우저	HTML4.0, HTML5, CSS2.1	웹서버
애플리케이션언어	개방형 언어 : J2EE5,J2SE5.0, 자바, 서블릿	스크립트 언어
웹언어 : XML 1.0 , XSL 1.1, RDF, AJAX	스크립트 언어	RDBMS
모델링 : UML 2.0	개방형 언어,웹언어	라이브러리
Script Code	PHP, JSP, ASP	MS-SQL

2.2 HTTP 동작과정

웹 브라우저는 웹 서버와 80번 well -known 포트로 TCP 연결을 하고 웹 브라우저가 웹 서버에게 요청을 보내며 웹 서버는 요청을 보낸 웹 브라우저에게 응답 후 연결을 해제한다. HTTP는 Stateless로서 상태 관리를 하지 않으며 TCP 연결 시 7개의 개체를 나누어서 전송하며 모든 객체를 전송받은 다음 연결을 종료한다. HTTP Request는 웹 서버에 데이터를 요청, 전송 시 보내는 패킷으로 GET방식은 일반적 HTTP Request 형태이며 웹 브라우저에 요청 데이터에 대한 인수를 URL(uniform resource locator)을 통해 전송한다. HTTP 요청 메시지는 요청라인(request line), 헤더(header), 몸체(body)로 구성된다. 헤더는 서버 프로그램 종류와 버전, MIME 정보(버전, 정보길이, 종류), 몸체에 담고 있는 문서 길이, 문서의 유형 등 문서 관련정보를 포함 한다. HTTP 응답메시지는 상태라인(Status Line), 헤더(Header), 몸체(Body)로 구성된다[3][4].



(그림1) HTTP 응답메시지 구조

2.3 Script Code 동작과정

Servlet과 JSP는 상호 보완적으로 같이 사용되는 웹 시스템 언어이다. JSP는 사용자 정의 태그를 지정할 수 있는 기능이 있어서 보다 효율적인 웹 사이트를 구성할 수 있다. JSP는 응용 프로그램 인터페이스(API)로서 ASP(Active Server Page)와 유사하나 JSP는 자바 프로그램을 호출하고 ASP는 스크립트(VBScript나 JScript)를 사용 한다. Servlet은 서버에서 수행되는 소형 프로그램으로, 서버에 위치하며 사용자 입력에 의해 데이터베이스에 접근에는 공통 게이트웨이 인터페이스(CGI) 프로그램이 연결된다. 자바 서버 프로그램은 자바 프로그래밍 언어로 CGI 수행 속도가 빠르고, 프로그램 프로세스가 생성되는 것이 아니라 상주 프로그램(daemon)의 하나의 스레드(thread)로 수행된다. 추가(add-on)모듈로 된 자바 Servlet은 넷스케이프 엔터프라이즈와 IIS, 아파치 서버에서 수행된다. Servlet을 보면, 비즈니스 로직(자바 코드)과 프리젠테이션 코드 (HTML)가 강하게 결합되어 있음을 알 수 있다.

3. XSS 프로세스 진단

3.1 웹 프로그램 코드 특징

웹 시스템 환경에서 어떤 DB에 값을 입력, 삭제 시 웹 프로그램은 ASP, JSP, PHP 등 스크립트 코드가 사용된다. hello.jsp 페이지에 나타나고 있는 코드는 대부분 HTML이다. 자바코드를 JSP 페이지에 작성하기 위해 <%와 %>, <%= 와 %>태그를 사용한다. <%= 와 %> 태그 사이에 있는 name은 자바 프로그래밍 언어로 선언한 변수이름이다. 이 태그들은 웹 컨테이너에게 런타임에 자바코드를 실행시켜야 함을 알려주기 위한 목적으로 사용된다. JSP는 HTML과 흡사한 태그로 어려운 코딩 없이 자바 객체를 사용할 수 있다. 웹 서버가 JSP 페이지에 대한 요청을 받았을 때 이 요청이 JSP컨테이너로 전해진다는 사실은 웹 브라우저가 알 필요가 없고 알 수도 없다. 해당 파일에 있는 코드를 읽고 해석하는 동적 콘텐츠를 만들고 이것을 정적 콘텐츠에 끼워 넣은 다음 최종 페이지를 HTTP서버에게 돌려주는 일들을 JSP 컨테이너가 수행한다. 웹 브라우저는 HTML 로 만들어진 최종 페이지만을 보여주며 주변 상황 인식을 하지 않는다. JSP 수행과정은 ① HTML 과 Java Code가 결합된 JSP Web Page를 작성 한다. ② JSP를 최초로 호출하면 해당 JSP File을 Servlet으로 변환하고, Servlet을 Class Code로 Compile 한다. ③ Class Code가 생성되면 Class Loader에 의해 Class를 Loading한다. ④ Class가 Loading 되면 다음 요청이 있을 때 마다 재사용 된다.

3.2 XSS 공격 프로세스

XSS 공격자는 보안 취약점이 존재하는 웹 페이지의 사용자 입력으로 “test”와 같이 정상적 스트링을 입력하는 것이 아니라 <script>로 시작하는 악성 스크립트 코드를 입력한다. 웹서버는 공격자 가 입력한 악성 스크립트 코드가 포함된 웹 페이지를 생성해서 클라이언트에게 되돌려준다. 웹 페이지에 포함된 스크립트 코드는 클라이언트 측 브라우저상에서 실행된다. 공격자는 웹 메일이나 게시판 등을 이용하여 리턴되는 웹페이지를 공격목표가 되는 일반 사용자에게 전달한다. 공격자는 웹 메일에 악성 스크립트 코드를 포함하여 전달하거나 게시판에 악성 스크립트 코드가 포함된 글을 포스팅한 후 읽도록 한다[7][8].

3.3 웹 시스템의 XSS에 대한 취약 요인

3.3.1 Client-Side Script 언어 측면의 취약요소

자바스크립트처럼 클라이언트 측에서 실행되는 Client-Side Script 언어인 Java Script, VB Script, Flash, ActiveX, XML/XSL DHTML 등에서 작성된 코드를 사용자 입력으로 주게 되면 이 코드가 그대로 클라이언트 컴퓨터에게 전달되어 브라우저상에서 실행된다. 공격자는 XSS 취약점이 존재하는 웹 사이트를 이용하여 자신이 만든 악의적인 스크립트를 일반 사용자의 컴퓨터 에 전달하여 실행시킬 수 있다. 이 공격방법을 통 해 사용자 쿠키를 훔쳐서 해당 사용자 권한으로 로그인하거나 브라우저를 제어하는 취약요소가 있다[9].

3.3.2 데이터 안전성 관리 측면의 취약요소

XSS는 데이터 암호화 또는 검증하는 과정 없이 사용자의 입력 데이터 값을 애플리케이션이 그대 로 받아들이거나, 웹 브라우저로 응답할 때 발생 한다. 사용자의 브라우저에서 스크립트를 실행 하게 함으로써 사용자의 세션을 가로채거나 변조 또는 악성코드를 업로드 등 공격이 가능하다. XSS 취약점은 무엇보다 웹 애플리케이션이 사용자 입력으로 받은 악성코드를 필터링하지 않은 데이터를 그대로 동적으로 생성된 웹 페이지에 포함하여 재전송 하는데 기인한다[10].

3.3.3 사용자측면의 취약요소

클라이언트 서버 간 통신 과정에서는 사용자 측면의 취약요소가 중요한 비중을 차지한다. 악의적으로 조작된 URL은 보통 http://도메인 네임으로 시작하는데, URL이 아무리 길어도 친숙한 도메인 네임으로 시작하기 때문에 사용자 들이 별다른 의심을 하지 않게 된다. 조작된 웹 화면에 그 URL 이 링크로 포함된다면 사용자는 누구든지 별다른 의심 없이 그 링크를 클릭하게 될 것 이다. 또한 사용자 측면에서 데이터를 암호화하지 않거나, 키를 안전하지 않은 장소에 보관하는 등 보안수칙 을 철저히 이행하지 않는 경우가 위험에 노출되는 큰 원인을 제공한다[11].

4. 웹 해킹 대응절차 체계

4.1 대응절차의 프레임워크

이상으로 제기한 XSS 프로세스 진단을 기반으로 한 웹 해킹 대응절차 모델은 웹 시스템 구조, 소프트웨어 구조, HTTP 요청과 응답 메시지 구조, HTTP 동작과정을 유의하여 발생 가능한 취약점 요소를 찾아 대비한다. 취약점 요소를 찾기위해 웹 프로그램 코드형식, XSS 공격 프로세스, 웹 시스템의 XSS에 대한 취약요인을 진단 한다. XSS 공격 대응체계는 한 가지 종류의 기술이나 대책이 아닌 대책 프레임워크를 구성하고 정보보안을 단계적으로 실행하는 방법이다. 특정 방법만 의존 시 많은 보안 공백을 발생 시킬 수 있다. 본 연구에서는 대응방안 모델로 대책의 구조를 시스템 설계단계, 운용단계, 사용단계로 구분하여 발생 가능한 취약성에 대처한다. 각 단계에서 각각 적용할 이행방안을 도출하여 구성 하는데 설계자 - 시스템 운용자- 사용자로 시스 템 흐름에 의거한 대응책을 발굴하여 개발, 운용 절차를 정립하는 과정이다.

<표2> 웹 해킹 대응모델

정보시스템 SDLC 과정			
보안 단계	시스템 설계단계	시스템 운용단계	시스템 사용단계
이해 관계자	시스템 개발자 (developer)	시스템 운용자 (maintainer)	시스템 사용자 (user)
수행 과정	분석,설계,프로그래밍,테스트	구현,시험운영, 업무전환	시스템 사용
이행 사항	●사용자 요구사항 분석과정의 보안 ● 개발과정의 보안수칙 ●소프트웨어아키텍쳐측면보안	●소스코드 운영보안 ●웹서버 운용 ●서보안버 운용 ●DB핸드링 운용보안	● 사용자시스템이용보안 ● 데이타입력시 보안이행사항

4.2 시스템 구조설계 단계의 보안

웹 시스템 구조설계 단계의 보안은 시스템 설계 전반에 걸친 정보보호 설계로 개발단계는 물론 애플리케이션의 변화관리, 형상 관리, 운영까지 대처한다. 시스템 개발 라이프 사이클 전반의 기반 구축 작업으로서 개발 프로젝트를 내부 정책부터 컴플라이언스 준수까지의 모든 과정을 개발 프로젝트와 통합하는 시스템적인 시큐어 코딩 기반설계 이다. 시스템 구조 설계 보안사항은 서비스 메뉴 구조, 각 항목별 기능, 업무 흐름도, 데이터 흐름도, 서비스 구성도를 보안측면에서 설계한다. 이는 SDLC 전반 위협요소 진단의 기초이며 이러한 위협을 회피하기 위해 소스코드 어느 부분이 수정되어야 하는지 추적할 수 있는 토대가 된다.

<표3> 웹 시스템 구조설계 사항

검증 항목	세부항목
시스템의 개요	시스템의 기능, 대표적 시스템, 서브시스템
서비스 메뉴 구조	메인메뉴, 서브메뉴
각 항목별 기능 설명	Component, Process, Thread
업무 흐름도	업무의 기능적 흐름
데이터 흐름도	Data flow, Event, 공유자원에 대한 동기화

서비스 구성도	서비스 내부 구성도
주요정보 정의	주요 정보에 대한 정의

4.3 소프트웨어 아키텍처 설계 보안

프로그래밍 단계 보안은 애플리케이션 자체의 소스 코드 보안이다. 애플리케이션 자체의 소스 코드 보안은 소스코드에서 보안 취약점이 발생할 수 있는 지점을 정확하게 찾아내고 이를 개발자가 수정할 수 있는 가이드를 개발한다. 이를 위해 코드 함수 추적, 변수 추적 등을 통해 소스코드 라인별로 취약점을 정확하게 찾는다. 컨퍼넌트(component) 종류, 프로세스관계도(relationship), 코드관계도(relationship), 매핑 관계도의 변화 관리와 형성관리가 포함된 배포 기능까지 갖춰 소프트웨어 아키텍처 설계를 시행한다.

<표4> 소프트웨어 아키텍처 설계사항

항목	설계기능
시스템기능	대표적 시스템, 서브시스템 기능
컨퍼넌트(component) 의 종류	시스템 배포 후 프로세스, 쓰레드 생성될, 커뮤니케이션, 자원공유방식
프로세스관계도(relationship)	Data flow, Event, 공유자원에 대한 동기화(synchronization on shared resources)
코드관계도(relationship)	class, object, procedure, function, 혹은 이것들의 추상적 집합체 (subsystem, layer, module)
프로세스관계도(relationship)	call, method invocation, 논리적 컨테인먼트 관계(is-a-sub -module-of)
매핑관계도	시스템의 기능이 어떻게 코드로 매핑 되는지 파악

4.4 시스템 운영과정의 보안

● 시스템 관리 보안

보안에 안전한 개발절차에 따라 개발되었더라도 시스템 운영과정에서 존재할 수 있는 보안 문제 들을 점검하고 대처하는 과정이 필요하다. HTTP 헤더의 특성상 특수문자가 사용되므로, 차단 설정 후에는 일정기간 동안 로그 모니터링을 통해 서비스에 문제가 없는지 확인해야 한다. 웹 방화벽은 보완장치로 활용해야 하며, 프로그램 수정 조치 등 근본적 원인제거가 반드시 수행되어야 한다. 프로그램 소스 상에서 입력값 검증이 적절히 이루어졌는지 점검(white box test)하고 웹 취약점 점검도구를 병행하여 점검(black box test)한다.

● 세션 관리

XSS 취약성 발생요인인 URL 접근 제한 실패를 막기 위해서는 기본적으로 모든 웹 페이지에 세션 에 대한 인증을 수행한다. 모든 웹 페이지에 대해 일관성 있는 인증 로직을 적용하려면 기업 단위 에서 또는 웹 사이트 단위에서 세션. 인증 로직을 표준화하고, 모든 웹 페이지를 개발할 때 해당 표준을 준수한다.

● 입력값 검증

웹 브라우저는 HTTP request에 URL, Query string, Form Fields, Hidden Fields, Cookies, Headers 등을 포함하여 웹 어플리케이션에 전송 한다. 어플리케이션은 이 정보를 가지고 웹 페이지를 생성하므로 request를 수정하여 공격할 수 있다. 따라서 모든 parameter에 대해 사용 전에 검증을 수행하여 허용된 값만을 받아들이는 Positive 방식으로 parameter 검증을 실시한다.

● 접근 통제

사용권한이 없는 사용자가 시스템에 접근하여 권한 없는 기능을 수행하는 문제가 기본 원인이다. 시스템 운용 무자격자를 통제하는 시스 템 설계가 취약하여 공격자가 SQL-Injection 등의 방법을 사용하여 다른 사용자 계정으로 접근할 수 있다. 애플리케이션의 접근통제 요구 사항을 반영하여 보안정책을 명시한다. 또한 접근통제 메커니즘의 우회 가능성을 검증한다.

● 인증 관리

인증 메커니즘도 패스워드 분실 등 취약한 사용자의 credential 관리로 인해 침해받을 수 있다. 인증토큰이 적절히 보호되지 않으면 공격자가 현재 활성화된 사용자의 세션을 가로채 다른 사용자를 가장하여 접속 가능하다. 대처방안은 패스워드 강도가 높은 강력한 패스워드를 사용한다. 전송중의 credential을 보호하고 세션 아이디를 보호한다.

4.5 사용자 보안

● 사용자의 필터링 방법

XSS공격으로 인한 피해를 최소화하기 위해 입력 문자열에서 미리 정해놓은 정상적인 문자들을 제외한 나머지 모든 문자들을 제거한다. 사용자가 이용할 수 있는 기능들을 몇 개의 영역으로 나누어서 영역경계를 넘어갈 경우 재 인증을 요구 한다. 웹 메일 프로그램에서 패스워드 변경이나 개인정보 열람 등에 대해서 재 인증을 요구하면, 공격자가 쿠키 스니핑에 성공하여 메일 내용은 열람할 수 있어도, 패스워드 변경이나 개인정보 열람은 불가능하다. 게시판의 경우 가급적 HTML 포맷 사용자 입력 불가능으로 설정한다.

● 코드를 사용한 특수문자 필터링

웹 해킹의 가장 기본적인 형태 중 하나인 인수 조작은 예외적인 실행을 유발시키기 위해 일반적으로 특수문자를 포함하게 되어있다. 아이디와 패스워드를 넣는 부분에 '문자열을 입력받지 못하도록 ASP 코드를 수정한다. CSS(cascading style sheets) 기반의 언어는 웹 프록시를 통해 웹 브라우저에 전달되기 때문에 웹 프록시를 통해 전달되는 과정에서 변조될 가능성이 있다. 웹 문서의 전반적인 스타일을 미리 저장해 둔 스타일 시트인 CSS 기반의 언어로 필터링할 경우 공격자가 필터링 로직만 파악하면 쉽게 필터링이 무력화 된다.

● 웹 어플리케이션 사용자 보안

웹 어플리케이션 측면에서 클라이언트가 제공한 특수 문자를 제거하거나 특수 문자를 HTML 문자로 바꾼다. 필터링 대상은 GET 질의 문자열, POST 데이터, 쿠키, URL, 그리고 일반적으로 브라우저와 웹서버가 주고받는 모든 데이터를 포함한다. 어플리케이션 차원에서 HTTP 헤더, 쿠키, 쿼리스트링, 폼 필드, 히든필드 등 인자에 대해 허용된 유형의 데이터만을 받아들인다. 웹 어플리케이션을 통해 악성코드를 전송하여 시스템 콜을 통한 OS 호출, 쉘 명령어를 통한 외부 프로그램 사용, SQL구문을 통한 백 엔드 데이터베이스 호출 등을 수행한다.

5. 결 론

급변하는 웹 시스템 환경에서 새로운 보안 위협에 대해 대응할 수 있는 보안 기술과 최근 추세, 보안기술을 기업 환경에 효율적으로 적용시키기 위한 방법 연구는 매우 절실한 과제이다. 이런 측면에서 XSS 공격 프로세스 진단을 기반으로 한 웹 해킹 대응체계 모델을 설계하였다. XSS에 대한 단계적 대응방안 모델은 실무현장 사례를 활용하여 대책의 구조를 시스템 설계단계, 운용 단계, 사용단계로 구분하여 발생 가능한 취약성에 대해 단계적 대책을 도출했다. 시큐어 코딩은 개발 라이프 사이클과 접목해 보안을 적용하는 전체 과정 애플리케이션 개발단계 소스코드 보안성을 점검, 취약성 발생 요인을 제거하고 운영단계의 '애플리케이션 데이터 보안' 영역으로 시큐어 코딩 체계를 설계할 때 실현된다. 이를 위해 SDLC 단계에 기초하여 각 단계에 적용할 이행방안을 프레임워크로 설계했다. 제안된 프레임워크는 설계자 - 시스템 운용자 - 사용자 시스템 기능 흐름에 의거한 대응책의 기반으로서 업무현장의 보안 이행의 토대로 활용되기를 기대한다.

참고문헌

[1] David Gourley and Brian Totty, "HTTP: The Definitive Guide", O'Reilly Media, 2002.

[2] Jeom goo Kim · SiChoon Noh,A Study of Step-by-step Countermeasures Model through Analysis of SQL Injection Attacks Code, Mar.2012.

[3] Stepen Cost, An Introduction to SQL Injection Attacks,for Oracle develops, 2007.3

[4] David Gourley and Brian Totty, "HTTP: The Definitive Guide", O'Reilly Media, 2002.

[5] http://www.owasp.org/index.php/Cross-Site _Request_Forgery

[6] OWASP, CSRF Guard, http://www.owasp.org/index.php/CSRF_Guard

[7] J. K. Kwon, S. Park and D. K. Sung, "Log-likelihood ratio(LLR) conversion schemes in orthogon

al code hopping multiplexing," IEEE Comm. Letters, vol. 7, no. 3, pp. 104-106, Mar. 2003.

[8] N. Jovanovic, E. Kirda, and C. Kruegel, "Preventing Cross Site Request Forgery attacks", In IEEE International Conference on Security and Privacy in Communication Networks (SecureComm), 2006.

[9] Yia-an Huang, Wenke Lee, "A Cooperative Intrusion Detection System for Ad hoc Networks," Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks, 2003, pp. 135-147.

[10] Sichoon Noh, Dong Chun Lee, and Kuimam J. Kim, "Improved Structure Management of Gateway Firewall
Systems for Effective Networks Security", Springer, 2003.

[11] Stephen Marsh and Mark R. Dibben, "Trust, Untrust, Distrust and Mistrust – An Exploration of the Dark(er) side", iTrust 2005, LNCS 3477, pp. 17-33, 2005.

[저 자 소 개]

노 시 춘 (SiChoon Noh)

1987년 : 고려대학교
경영정보학(석사)
2005년 : 경기대학교
정보보호기술(박사)
2002년 : KT 시스템보안부장
2004년 : KT 충청전산국장
2005년~현 재 : 남서울대학교
컴퓨터학과 교수
2011년~현 재 : 남서울대학교
IT융합연구소 연구위원

email : nsc321@nsu.ac.kr