REPORT

SW보안개론

(1번과제: 용어이해, Log4j 취약점 원리와 대응)



**과목명 :SW보안개론**

**분반 : 1분반**

**교수명 :조성제**

**학 과 : 소프트웨어학과**

**학 번 : 32162566**

**이 름 : 안찬웅**

**2022년 4월 05일**

EMB0000356c3453

<목 차>

I. 용어이해 및 Log4j 취약점 원리와 대응

1.1 Botnet 3

1.2 Cryptojacking 3

1.3 SQL injection 4

1.4. Cross-site Scripting (XSS) 4

2.1 Attack surface 5

2.2 Attack surface 5

2.3 code review 6

2.4 penetration testing 6

3.1 심각한 Log4j 취약점 3가지 설명 7

3.2 공격자가 이 3개의 취약점들을 악용하여 어떻게 공격 할 수 있는지 설명 8

3.3 이 3가지 취약점을 완화하거나 제거하기 위한 대응 기법을 설명 9

4.0 Discussion 11

**1-1) Botnet 정의**

봇넷은 다양한 사기 및 사이버 공격을 수행한다. 사용되는 하이재킹된 컴퓨터 장치의 네트워크 입니다. 봇넷은 로봇과 네트워크의 합성어로 봇넷 조립은 다계층 체계의 침투 단계입니다. 봇은 데이터 도용과 서버충돌 그리고 멜웨어 베포와 같은 대규모의 공격을 자동화 하는 도구 역할을 합니다. 봇넷은 사용자의 동의 없이 장치를 사용하여 다른 사람에게 사기를 치거나 혼란을 일으킬 수 있습니다.

**1-1) Botnet 동작 방식**

봇넷은 더 큰 공격을 수행하는 해커의 능력을 성장, 자동화 시키고 구속화 하도록 구축되었습니다. 한 사람이나 소규모 해커 팀이 로컬 장치에서 수행할 수 있는 작업은 너무 많습니다. 그러나 적은 비용과 약간의 시간을 투자하게 되면 더 효율적인 운영을 위해 활용 할 수 있는 추가 장비를 얻게 됩니다. 봇 목자는 원격명으로 하이재킹 된 장치 집합을 이끕니다. 일단 봇을 컴파일 하게 되면 목자는 명령 프로그래밍을 사용하여 다음 작업을 수행합니다. 명령을 수행하는 당사자가 봇넷을 설정했다거나 임대로 운영했을 수 있고, 좀비 컴퓨터 또는 봇은 봇넷에서 사용하기 위해 인계된 각 맬웨어에 감염된 사용자 장치를 나타냅니다. 이러한 장치는 봇 허더가 설계한 명령에 따라 작동합니다.

1단계 노출은 해커가 웹사이트, 앱에서 취약점을 찾는 것으로 시작되고 목표는 사용자가 자신도 모르게 멜웨어 감염에 노출되도록 설정하는 것 입니다. 일반적으로 해커가 소프트웨어나 웹사이트 보안 문제를 악용하고 이메일 및 기타 온라인 메시지를 통해 악성코드를 전달합니다. 2단계에서 사용자 장치를 손상시키면 봇넷 악성코드에 감염됩니다. 이런 방법중 대부분은 사용자가 소셜 엔지니어링을 통하여 특수트로이 목마 바이러스를 다운로드 하도록 유도하는 것과 관련이 깊습니다. 다른 공격자는 감염된 사이트를 방문시 드라이브 바이 다운로드를 사용하여 공격적일 수 있습니다. 전달 방법에 상관없이 사이버 범죄자는 여러 사용자 컴퓨터 보안을 침해합니다. 3단계에서 공격자는 감염된 시스템들을 원격으로 관리 할 수 있는 “봇” 네트워크를 구성합니다.종종 사이버 범죄자는 수백만 대의 컴퓨터를 감염시키고 제어 하려고 합니다. 그런 다음에 사이버 범죄자는 대규모 봇넷의 보스 역할을 할 수 있습니다.

**1-2) Cryptojacking**

Cryptojacking은 해커가 해커를 대신해서 암호화 화폐를 불법적으로 채굴하기 위하여 표적의 컴퓨팅 능력을 이용해서 공격하는 사이버 공격의 한 유형입니다. Cryptojacking은 개별 소비자, 대규모 기관과 산업 제어 시스템을 대상으로 합니다. Cryptojacking과 관련된 맬웨어 변종은 채굴 프로세스가 다른 합법적인 활동보다 우선시하기 때문에 감염된 컴퓨터의 속도를 늦춰줍니다. 또한 Cryptojacking과 브라우저 마이닝의 합법적 관행 사이경계가 항상 확실한 것은 아닙니다. Cryptojacking은 피해자의 기계를 활용하여서 채굴하거나 암호화폐의 블록체인을 업데이트 하는데 필요한 계산을 수행하여 새로운 토큰을 만들고 그 과정속에서 수수료를 생성합니다. 이 새로운 토큰과 수수료는 공격자가 소유한 지갑에 들어가는 반면에 전기 및 컴퓨터 마모를 포함한 채굴 비용은 피해자가 부담하게 됩니다.

**1-3) SQL injection**

SQL injection기법은 공개된지 몇 년이 지났지만 웹해킹 공격중 가장 위험한 공격이다. 웹프로그래밍은 자료의 효율적인 저장 과 검색을 위해서 DBMS를 반드시 사용하고 있다. 주로 PHP,JSP,ASP등의 스크립트 언어를 이용하여 DBMS와 연동한다. 이러한 웹 어플리케이션에서 클라이언트의 잘못된 입력값을 확인하지 않아 비정상적인 SQL 쿼리가 발생할 수 있다. 이런 비정상적인 쿼리는 사용자 인증을 우회하거나 데이터베이스에 저장된 데이터를 노출시킬 수 있다. 공격자는 SQL injection의 취약점을 이용해 아이디와 암호를 모르더라도 웹기반 인증을 통과할 수 있고, 데이터베이스에 저장된 데이터를 볼 수 있다. SQL injection에 대한 대한으로 다수의 방법이 발표되었는데, 한가지 방법에 의존하는 경우에는 보안 공백이 발생할 수 있다. 단계적 대응모델은 사고 예방적 측면에서 소스코드 작성단계, 서버 운동단계, 데이터베이스 핸드링단계와 사용자 입력 값 검증 활용 단계에서 대책을 프레임워크로 구성하여서 적용하는 방식이다. 이 대응모델을 적용하면 운용 과정을 통해서 존재하는 SQL injection의 공격가능성을 효과적으로 차단이 가능하다.

**1-4) Cross-site Scripting (XSS) 정의**

XSS는 공격자가 악성 실행 스크립트를 신뢰할 수 있는 응용 프로그램이나 웹 사이트의 코드에 삽입되는 공격이다. 공격자는 종종 사용자에게 악성 링크를 보내고 사용자가 클릭하도록 유인하여 XSS 공격을 시작한다. 앱이나 웹사이트에 적절한 데이터 삭제가 없다면 악성 링크가 사용자 시스템에서 공격자가 선택한 코드를 실행한다. 공격자는 사용자의 활성 세션 쿠키를 훔칠 수 있다.

**1-4) Cross-site Scripting (XSS) 작동방법**

Cross-site Scripting(XSS)가 공격을 수행하기 위해 공격자는 사용자가 제공한 입력에 악성 스크립트를 삽입한다. 공격자가 요청을 수정하여 공격을 수행할 수 도 있고, 웹 앱이 XSS공격에 취약한 경우에 사용자가 제공한 입력이 코드로 실행된다. 예를들어서 아래 요청에서 스크립트는 “XSS” 라는 텍스트가 있는 메시지 상자를 표시한다.

[*http://www.site.com/page.php?var=<script>alert('xss');</script*](http://www.site.com/page.php?var=%3cscript%3ealert('xss');%3c/script)*>*

XSS 공격을 유발시키는 방법에는 여러가지들이 있는데, 예를들어 페이지가 로드되는 것 또는 사용자가 페이지의 하이퍼링크 같은 특정요소 위로 마우스를 가져갈 때 실행이 자동으로 트리거 될 수 있다. XSS공격의 잠재적인 결과는 사용자 키 입력을 캡처하고 사용자를 악성 웹사이트로 교차사이트 스크립팅 공격의 잠재적 결과는 사용자 키 입력을 캡처하고, 사용자를 악성 웹사이트로 rederection하고, 웹 브라우저 기반 익스플로잇을 실행한다. 어떤 경우 XSS 공격으로 인해서 피해자 계정이 완전히 손상되고, 공격자는 모든 정보를 공격자에게 제공하는 가짜 양식에 자격 증명을 입력하도록 사용자를 속일 수 있다.

**1-4) Cross-site Scripting (XSS) 접근방식**

저장된 XSS는 악성 페이로드가 데이터베이스에 저장될 때 발생한다. 출력 인코딩이나 삭제가 없는 경우에는 데이터가 요청될 때 다른 사용자에게 렌더링 된다. 반사된 XSS는 웹 애플리케이션이 공격자가 제공한 문자열을 피해자의 브라우저에 보내 브라우저가 문자열의 일부를 코드로 실행하도록 할 때 발생한다. 페이로드는 서버 측 출력 인코딩이 없기 때문에 응답으로 다시 에코된다. DOM기반 XSS는 공격자가 응답에 스크립트를 삽입할 때 발생한다. 공격자는 무선 개체 모델 데이터를 읽고, 조작하여 악성 URL을 만들 수 있다. 공격자는 무선 개체 모델(DOM) 데이터를 읽고 조작하여 악성 URL을 만들 수 있고, 공격자는 이 URL을 사용하여 사용자가 클릭하도록 속인다. 사용자가 링크를 클릭하면 공격자는 사용자의 활성 세션 정보와 키 입력을 훔칠 수 있다. 저장된 XSS 와 반영된 XSS와 달리 전체 DOM기반 XSS공격은 클라이언트 브라우저에서 발생한다.

**2-1) Attack surface 정의**

공격자가 해당 시스템, 구성 요소 또는 환경에 들어가거나 영향을 미치거나 데이터를 추출할 수 있는 시스템, 시스템 구성 요소 또는 환경의 경계에 있는 지점 집합입니다.[[1]](#footnote-1)

**2-2) Attack surface를 줄이는(최소화하는) 방법은?**

1. 제로 드러스트를 가정한다. 사용자는 자신의 신원과 장치 보안을 입증할 때 까지 리소스에 액세스 할 수 없다. 이러한 요구 사항을 완화하고 사람들이 모든 것을 볼 수 있도록 하는 것이 더 쉽지만 보안을 최우선으로 생각하는 사고 방식이 회사를 더 안전하게 유지하게 한다.

2. 강력한 사용자 액세스 프로토콜을 만든다. 일반적으로 회사에서 사람들은 놀라운 속도로 영향력을 행사한다. 사람 은 좋은 일을 하기 위해 네트워크에 액세스 해야 하지만 그 해당 사람이 더 이상 조직이 아닐 때 이런 권한을 제거해야 한다. 인적 자원과 짝을 이루어 암호 정책을 강화해야 한다.

3. 강력한 인증 정책을 사용한다. 액세스 프로토콜 위에 강력한 인증을 계층화하는 것을 고려하자. 속성 기반 엑세스 제어 또는 역할 기반 엑세스 제어를 사용하여 적합한 사람이 데이터에 엑세스 할 수 있도록 한다.

4. 백업을 보호해라. 코드와 데이터의 복제본은 평균적인 회사의 공격 표면의 일반적인 부분이다. 사용자에게 해를 끼칠 수 있는 사람들로부터 이런 백업을 안전하게 유지하려고 한다면 엄격한 보호 프로토콜을 사용해야 한다.

5. 네트워크 세분화하라. 많은 방화벽을 구출할수록 해커가 빠르게 비즈니스 핵심에 침투하기가 어려워진다. 올바르게 수행하려고 한다면 단일 시스템 또는 사용자에게만 보안제어 적용을 할 수 있다. [[2]](#footnote-2)

**2-3) Code review에 대해 설명하시오**

여러명의 개발자가 본인이 만들지 않은 코드의 내용을 점검하고, 피드백을 하는 과정이다. 피드백은 오타나 버그, 개발표준 등에 대한 의견이나 좋은 코드에 대한 긍정적인 피드백을 의미한다. 코드리뷰의 중요한점은 버그를 조기에 발견할 수 있고, 개발 컨벤션을 준수하고 중복 코드 방지 및 모듈의 재사용성이 증가된다. 그리고 조직 간 수준이 상향 평준화될 수 있다.

**2-4) Penetration testing 정의**

Penetration testing은 악용 가능한 취약점을 확인하기 위해서 컴퓨터 시스템에 대한 사이버 공격을 시뮬레이션 한다. 웹 애플리케이션 보안의 맥락에서 침투 테스트는 일반적으로 웹 어플리케이션 방화벽을 강화하는데 사용한다. Penetration testing에는 코드 주입 공격에 취약한 비위생입력 과 같은 취약성을 발견하기 위해서 여러 어플리케이션 시스템에 대한 위반시도를 포함한다.

**2-4) Penetration testing 방법**

Penetration testing은 웹 어플리케이션 자체, 회사 웹 사이트, 이메일 그리고 도메인 이름서버와 같이 인터넷에서 볼 수 있는 자회사의 자산을 대상으로 하는데 목표는 액세스 권한을 얻고 중요한 데이터를 추출하는 것이다. 내부테스트에서 방화벽 뒤 애플리케이션에 액세스 할 수 있는 테스터는 악의적인 내부자의 공격을 시뮬레이션 한다. 이것은 반드시 불량 직원을 시뮬레이션 하는 것은 아니고, 일반적인 시작 시나리오는 피싱 공격으로 인해서 자격 증명이 도용된 직원일 수도 있다. 블라인드테스트에서 테스터는 대상이 되는 기업의 이름만 받고 이를 통해서 보안 담당자는 실제 애플리케이션 공격이 어떻게 발생하는지를 실시간으로 볼 수 있다. 이중맹검 테스트에서 보안 요원은 모의 공격에 대한 지식이 없다. 현실 세계에서 마찬가지로 침입 시도 전에 방어를 강화 할 시간이 없다. 표적 테스트에서 테스터와 보안요원은 함께 작업하고 서로의 움직임을 평가하는데, 이것은 보안 팀 해커의 관점에서 실시간 피드백을 제공하는 중요한 훈련이다.

**3-1) Log4j 정의**

Log4j는 java에서 사용하고 있는 로그 생성 모듈이다. Log를 생성할 때 사용하는 라이브로 로그를 생성하는 포맷을 쉽게 만들고 양식을 표준화 할 수 있어서 java개발시 사용하는 기본 모듈과 같다.

**3-1) 가장취약한 3가지**

CVE-2021-44228 - Apache Log4j2 2.0-beta9 ~ 2.15.0(보안 릴리스 2.12.2, 2.12.3 및 2.3.1 제외) 구성, 로그 메시지 및 매개변수에 사용되는 JNDI 기능은 공격자가 제어하는 ​​LDAP 및 기타 JNDI 관련 엔드 포인트로부터 보호하지 않는다. 로그 메시지 또는 로그 메시지 매개변수를 제어할 수 있는 공격자는 메시지 조회 대체가 활성화된 경우 LDAP 서버에서 로드 된 랜덤의 코드를 실행할 수 있습니다. log4j 2.15.0부터 이 동작은 기본적으로 비활성화되어 있다. 버전 2.16.0(2.12.2, 2.12.3 및 2.3.1 포함)부터 이 기능이 완전히 제거되었다. 이 취약점은 log4j-core에만 해당되며 log4net, log4cxx 또는 기타 Apache Logging Services 프로젝트에는 영향을 미치지 않는다.[[3]](#footnote-3)

CVE-2022-23305 - 의도적으로 Log4j 1.2.x의 JDBCAppender는 SQL 문을 구성 매개변수로 허용하며, 여기서 삽입할 값은 PatternLayout의 변환기이다. 메시지 변환기 %m은 항상 포함되어 있을 수 있다. 이를 통해 공격자는 의도하지 않은 SQL 쿼리를 실행할 수 있도록 기록된 응용 프로그램의 입력 필드 또는 헤더에 조작된 문자열을 입력하여 SQL을 조작할 수 있다. 이 문제는 기본값이 아닌 JDBCAppender를 사용하도록 특별히 구성된 경우에만 Log4j 1.x에 영향을 미친다. 버전 2.0-beta8부터 JDBCAppender는 매개변수화된 SQL 쿼리에 대한 적절한 지원과 로그에 작성된 열에 대한 추가 사용자 정의와 함께 다시 도입되었다. Apache Log4j 1.2는 2015년 8월에 수명이 종료되었다. 사용자는 Log4j 2로 업그레이드해야 이전 버전의 수많은 다른 문제를 해결할 수 있다[[4]](#footnote-4)

CVE-2022-23307 – Log4j 1.x의 전기톱 구성 요소에 대한 심각한 심각도. 이것은 Chainsaw 2.1.0에서 수정된 CVE-2020-9493에서 수정된 것과 동일한 문제이지만 Chainsaw는 Log4j 1.2.x의 일부로 포함되었다. CVSS는 9.8이다. [[5]](#footnote-5)

**3-2) 공격자가 이 3개의 취약점들을 악용하여 어떻게 공격할 수 있는지 설명하시오.**

Log4j 취약점을 악용하여 악성코드 변종, 봇넷, 암호화폐 채굴 악성코드, 이메일 기반 공격 등 점점 더 많아지고 있다. 처음 공개된 Khonsari 랜섬웨어는 Windows 시스템을 타겟으로 한다. log4j 취약점 발견에 따라 대체로 활발하지 않은 랜섬웨어 패밀리였던 TellYouThePass도 활동을 재개했다. TellYouThePass는 Linux와 Windows에서 실행되는 버전이 있다. 글 작성 시간 기준 Log4j 취약점을 악용한 랜섬웨어 공격이 성공한 케이스는 알려진 바가 없다. Log4j 공격 시도들은 알려진 악성 코발트 스트라이크 서버에 연결하는 것이 관찰되었다. 코발트 스트라이크는 해커들이 자주 사용하는 모의해킹 도구로, 원격 정찰과 내부 확산 공격과 같은 활동이 가능하다.[[6]](#footnote-6)

**3-3) 이 3가지 취약점을 완화하거나 제거하기 위한 대응 기법을 설명하시오.**

**방법1. JndiLookup 클래스 제거**

Log4j 취약점은 Log4j 가 자바 기능인 JNDI(Java Naming and Directory interface)를 사용하는 방식때문에 발생한다. JNDI는 런타임 실행중에 추가 자바 객체의 로딩을 허용하도록 설계됐다. 여러 프로토콜에 걸친 원격 네이밍 서비스에서 자바 객체를 로드하는 데 사용된다. 최초의 취약점 공격은 LDAP(Lightweight Directory Access Protocol)를 이용했다. LDAP는 가장 흔한 프로토콜일 뿐이며, DNS(Domain Name System), RMI(Remote Method Inovocation), NDS(Novell Directory Services), NIS(Network information Service), CORBA(Common Object Request Broker Architecture)와 같은 다른 프로토콜도 지원한다. 취약점을 고치는 방법인 JNDI 메시지 조회 기능을 중단하는 것, 원래 Log4j 2.16.0의 기능이다. 하지만 Log4j를 사용하지 않는 방법도 있다. 영향을 받은 Log4j 패키지에서 전체 JndiLookup 클래스를 아예 삭제하는 것이다. 자바 컴포넌트는 기본적으로 ZIP 아키이브 이기 때문에 관리자는 아래 명령을 실행해 취약 패키지 인스턴스를 수정 및 패치 할 수 있다.

**방법2. 자바 에이전트를 이용한 핫패칭**

핫패칭(Hotpactching)은 재 시작하지 않고 실행중인 프로세스에 패치를 전개하는 방법이다. 자바는 JVM(Java Virtual Machine)에서 API(Instrumentation API)와 런타임 동안 JVM에 동적으로 첨부 할 수 있는 JAR(Java Archive) 파일인 자바 에이전트를 사용하여 이미 실행중인 바이트 코드를 정지하거나 재시작 하지 않고, 변경하는 것을 지원한다. Log4k 취약점에 대흥해서 AWS코레토팀은 로딩된 모든 ‘org.apache.logging.log4j.core.lookup.JndiLookup’ 인스턴스의 lookup() 메서드를 패치하는 자바 에이전트를 개발했다. 원격 서버로 연결하는 대신에 ‘Pathched JndiLooup::lookup()’ 문자열을 반환한다. AWS 코레토팀의 자바 에이전트는 git hub에서 다운로드 할 수 있고, 임시 컨테이너로서 기존 쿠버네티스 팟으로 전개되어 다른 컨테이너에서 이미 실행중인 어플리케이션 패치에도 사용할 수 있다. 임시 컨테이너는 쿠버네티스 1.16 및 이후 버전에서 지원된다.

**방법3. 취약점을 이용한 임시적인 악용 방지**

영향을 받은 서버의 취약점을 이용하여 라이브 시스템 및 어플리케이션에 특정한 변경을 만들어서 추가 악용을 방지하는 방법이 있다. 보안업체 Cybereason연구팀이 개발하고 루나섹(LunaSec) 연구팀이 개선하여 라이브 서버에 공개하였다. 이 방법은 전면적 패치가 불가능한 서드파티 솔루션 업체의 제품이나 공식 업데이트를 지원받지 못하는 오래된 제품에 적합한 제품이다. 다만 취약점 자체를 활용하는 것은 단기적인 해법일 수 있다. 주의해야 할 점은 첫번째 효과가 일직이라는 것이다. 취약점으로 만든 변경은 실행 중인 자바 프로세스에만 적용되고, JVM이 재 시작되면 변경 전으로 복귀된다. 서버를 재 시작할 때 마다 변경 사항을 다시 적용해야 한다. 두번째, 여러 개의 구성 및 시스템에서 시험된 방법이지만 항상 유효한 것은 아니며 충동을 일으킬 수도 있다. 충돌로 인하여 복구 작업시에는 서버 재부팅이 필요할 수 있으므로 다운타임이 발생해서는 안되는 중요한 시스템에서는 이 방법을 지양한다. 마지막으로 이 방법은 취약점을 이용하는 방법이다. 이 때문에 소유권이 없고, 자신이 제어하지 않는 서버에서는 이 방법을 사용하는 것은 악의가 없더라도 불법일 수도 있다.

**방법4. 취약시스템 식별방법**

Log4j 취약점에 노출된 애플리케이션과 시스템을 모두 식별 하는 것이다. 이것은 쉬운 일은 아니다. 각각 애플리케이션이 자체 Log4j 인스턴스를 번들로 묶을 수도 있고, 서드파티 의존성의 일부로 번들을 동적으로 로드 할 수 있기 때문이다.

**취약점을 완화하는 방법**

**방법1.자바 버전 업그레이드** : 자바 버전 업그레이드만으로는 충분하지는 않지만, 최초의 취약점 공격은 6u212, 7u202, 8u192 또는 11.0.2 이후의 자바 버전에 효과가 없었다. 이들 버전의 기본 구성이 원격 서버에서 JNDI를 사용한 클래스 로딩을 금지하기 때문이다. 하지만 공격자가 원격 클래스패스 대신에 어플리케이션 자체 클래스패스의 클래스를 이용하는 페이로드를 제작할 수 있음이 밝혀졌고 업그레이드만으로 모든 공격을 방지할 수는 없다.

**방법2. formatMsgnNoLookups 플래그** : formatMsgNoLookups 플래그가 모든 취약점 공격을 방지하는 것은 아니다. 초기 Log4j 개발자를 비롯하여 여러 전문가들은 2.10이상의 Log4j 버전에서 ‘formatMsgNoLookups’라는 속성을 ‘true’로 설정하거나 ‘LOG4J\_FORMAT\_MSG\_NO\_LOOKUPS’라는 속성을 ‘true’로 설정하거나‘LOG4J\_FORMAT\_MSG\_NO\_LOOKUPS’라는 환경 변수를 설정하라고 권고 하였다. 하지만 이런 플래그를 활성화하더라도 서비스 거부 공격 취약점에는 효과적이지 못하다. 개발자는 메시지를 조회할 수 있는 Log4j 코드 경로가 여전히 존재한다. 대표적인 것은 Logger.printf(“%s”, userlnput)나 커스템 메시지 팩토리를 사용하는 어플레케이션이다. 이런 환경에서 결과 메시지가 StringBuilderFormattable를 실행하지 않는다고 말했다.

**방법3. 로그문장 포맷 변경**

로그 문장 포맷을 변경해 조회를 비활성화 하는 방법도 완전하지는 않다. 어플리케이션의 로그 문장 포캣을 %m, &msg 또는 %message에서 %m{nolooups}, %msg{nolookups} 또는 %message{nolookups}으로 변경하여 메시지조회 기능을 중단하는 것이다. 이것은 formatMsgNoLookups와 같은 이유로 효과가 없다. 그러나 잘못된 안전감을 유발해 훨씬 위험하다. 하나의 의존성에서 로깅 문장 갱신을 놓치거나 취약한 %m 문장을 다시 도입하기 쉽기때문이다. 이 완화 책을 이용하고 있으면 이 방법에만 의존하는 것을 피해야 한다.

방법4. WAF(web Application Firewall) : WAF로 취약점 공격을 차단할 수는 있다 그러나 잠재적은 공격 문자열을 모두 포착하는 것은 힘들다. Log4j 취약점이 발견된 후에 많은 전문가들이 WAF 필터링 규칙을 우회하는 중첩되고 불명료한 페이로드 제작사례들을 보여주었다. 이에 따라서 WAF 및 IPS 업체가 Log4Shell 탐지 정책을 끊임 없이 갱신하기 때문에 즉각적이고 임시적인 대응이나 다른 방법을 보완하는 추가 방어 계층으로 활용하기에 적절하다. 원문보기:

**Discussion (과제 수행하면서 어려운점, 배우고싶은 내용)**

들어보지 않은 생소한 보안용어들을 배워서, 어려움을 느꼈다. 그러나 익숙해지려고 노력했고 과제를 마무리한 지금 시점에서는 조금 친숙해진 것 같다. 다음 과제에서는 어렸을 때부터 관심이 많았던 암호 알고리즘에 관한 내용을 알아보고 싶다. 박규식 교수님 멀티미디어시스템 수업시간에 클라우드 컴퓨팅에 관한 보고서를 작성하였는데, “secret key” 문자열을 사용하여 암호화 복호화 프로그램과 aes\_256\_cbc 암호화 알고리즘에 대해 조사한적이 있다. 그때 당시 매우 흥미로웠다. 이번 강의를 수강하면서 아직은 보안과 많이 친해지지 않아, 보안용어들과 지금현재 대두 되고 있는 Log4j 뿐만 아니라 여러 보안문제의 대해 익히고 싶다. 앞으로 개발자가 되려면 프로그래밍 개발할 때 발생하는 여러 보안문제를 아는 것은 필수라고 생각한다. 내용이 정말 어렵지만 쉽게 설명을 해주셔서 내용이 잘 이해되었다. 앞으로도 반복해서 익히면서 보안용어들과 친숙해지고, 전반적인 SW보안 지식을 넓히고 싶다.

1. JOINT TASK FORCE, 12-10-2020 , Security and Privacy Controlsfor Information Systems and Organizations , https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf [↑](#footnote-ref-1)
2. 공격 표면을 줄이는 5가지 방법 . (2018년 8월).  <https://www.securitymagazine.com/articles/89283-ways-to-reduce-your-attack-surface>

   공격 표면 분석 치트시트. 치트시트 시리즈.

   <https://cheatsheetseries.owasp.org/cheatsheets/Attack_Surface_Analysis_Cheat_Sheet.html> [↑](#footnote-ref-2)
3. NIST, (12-10-2021), https://nvd.nist.gov/vuln/detail/CVE-2021-44228 [↑](#footnote-ref-3)
4. NIST, (01-18-2022), https://nvd.nist.gov/vuln/detail/CVE-2022-23305 [↑](#footnote-ref-4)
5. NIST, (01-19-2022), <https://nvd.nist.gov/vuln/detail/CVE-2022-23307> [↑](#footnote-ref-5)
6. MANDIANT, (12-15-2021), https://www.mandiant.com/resources/log4shell-recommendations [↑](#footnote-ref-6)