infosec

Threat Intelligence Report

# EQST INSIGHT

2023

01

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

## 보안위협의 변화와 효과적인 보안 취약점 대응 방안

### ■ 보안위협 변화와 위험 관리

최근 대부분의 기업들은 내부정보 유출을 막고 외부에서 접근해 오는 사이버 공격에 대응하기 위해 다양한 보안 솔루션을 도입해서 운영하고 있으며, 별도의 전문적인 보안담당자를 지정하여 보안 솔루션 관리 및 모니터링을 진행하고 있다.

과거에는 보안 인프라가 견고하지 않아 한 번의 공격으로도 공격자가 목적을 달성할 수 있었던 탓에 공격 빈도가 높고, 불특정 다수를 겨냥하는 공격 시도도 많았다. 그럼에도 보안담당자는 단일성 공격을 백신, IPS 등 보안장비의 업데이트만으로도 상당 부분 대응할 수 있었기에 다양한 보안 솔루션들의 관계성은 고려할 필요가 없었다. 이러한 흐름이 지속되면서 공격자들의 악의적인 공격은 성공 확률이 낮아졌고, 성공률을 높이기 위해 점점 더 복잡한 공격 형태로 진화하게 됐다.

이렇듯 보안 솔루션 간 상호 연계가 제대로 이뤄지지 않으면서, 최근 각 보안 솔루션에서 발생하는 대용량 로그가 보안담당자들이 탐지해 분석할 수 있는 가용 범위를 넘어섰다. 이를 해결하기 위해 다양한 보안 솔루션을 통합적으로 연계해 외부 침해 및 내부정보 유출을 방지하기 위한 통합관제(SIEM) 솔루션이 등장했으며, 고도화되고 지능화된 침입을 탐지 및 차단하기 위해 지속적으로 개선되고 있는 상황이다.

특히 최근 금융회사를 타깃으로 하는 공격이 늘고 있다. 금융 사이버 공간은 개인의 중요 금융 정보가 집약되고 관리되는 공간으로 실시간으로 악의적인 공격자들의 APT[1] 공격 침투 시도가 빈번히 발생하고 있는 상황이다. APT 공격은 보안 솔루션 탐지를 우회하기 위해 알려지지 않은 취약점을 악용하거나 샌드박스에서는 작동하지 않도록 설계됐기 때문에 하나의 보안 솔루션으로는 100% 탐지 및 차단할 수 없다. 하지만 금융보안 담당자는 이러한 위협에 선제적으로 대응해야 하며, 사이버위협 발생 시 중요 자산을 신속하고 안전하게 보호할 수 있도록 공격자를 식별하고 대처할 수 있어야 한다.

이에 금융회사들은 금융 관련 법규에 따라 운영리스크, 시장리스크, 신용리스크 등을 체계적이고 정량적으로 관리하고 있다. 하지만 아직 정보보호 리스크를 관리하기 위한 체계가 제대로 마련되지 않은 탓에 IT 보안 리스크를 정량화하여 관리하는 기업은 많지 않은 상황이다.

정보보호에 있어 위험관리는 기술적 데이터로 구성된 위협 및 취약점 정보를 관리적 측면의 정보로 나타낼 수 있는 방안이므로, 정보보호의 중요성을 토대로 이를 운영하기 위한 관리 체계가 반드시 수립되어야 한다. 일반적으로 정보보호 예방 활동에는 취약점 관리 업무와 별도로 정보보호 위험관리 업무가 있는데, 두 업무 간의 기술적, 관리적 측면에서 실시간 연결성을 가져가는 것이 매우 어렵다는 점에서 이러한 관리 체계의 필요성은 더욱 커진다.

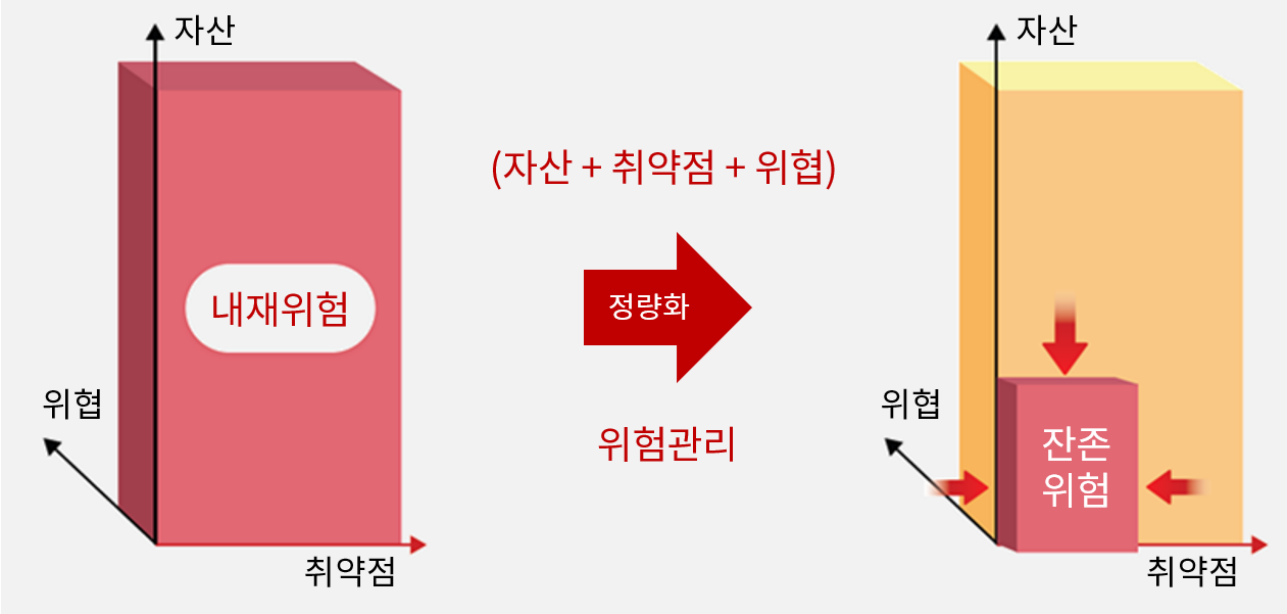
BNK 부산은행에서는 2016 년부터 위와 같은 보안위협의 변화와 위험관리의 필요성을 고려하여 정보보호 통합 플랫폼을 구축하고 있다. 본 기고문은 금융정보보호 컨퍼런스 ‘FISCON 2022’에서 발표된 '정량적 위험관리를 통한 통합 보안관제 구축 사례' 내용을 중점적으로 소개한다.

[1] APT(Advanced Persistent Threat, 지능형 지속 공격 위협) : 특정한 타깃을 대상으로 한 지속적으로 사이버 공격을 의미함.

## ■ 정보보호 위험관리

정보보호 위험관리 대상은 “자산의 가치, 자산이 가지고 있는 취약점, 외부의 위협”으로 구분할 수 있다.

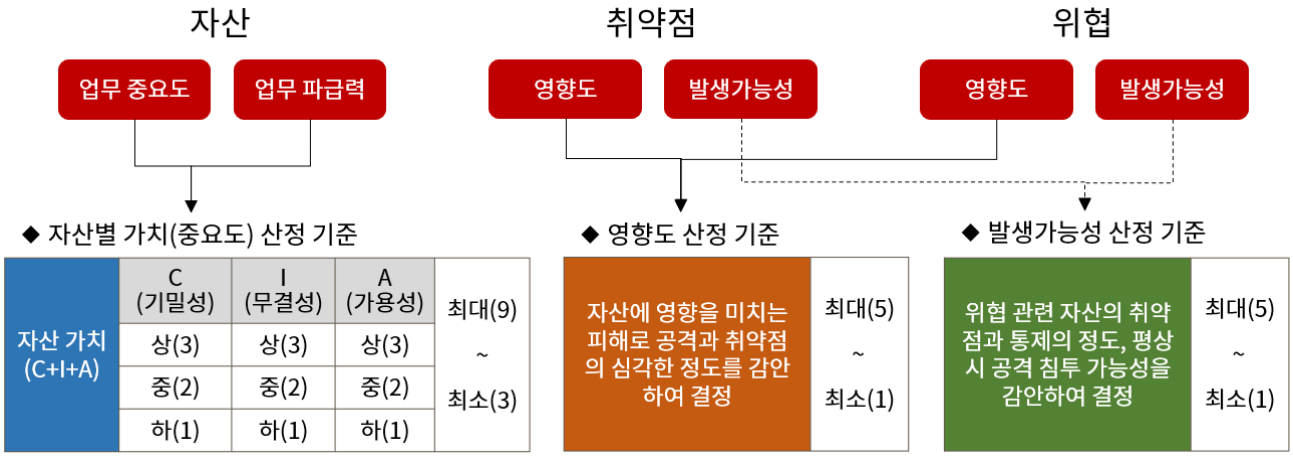
위협하는 자의 위협 행위와 정보를 취급하는 대상 자산의 취약성을 결합하여 위험 측정하고, 측정된 값이 높다는 것은 위험이 사고로 이어질 가능성이 높다는 것을 의미한다. 따라서 정보보호 위험관리라는 것은 “자산과 취약점, 위협을 종합하여 정량화된 수치가 나오고, 이것을 관리” 하는 것이며, “자산, 취약점, 위협”에 대한 내재 위험을 줄임으로써 잔존 위험을 최소화” 하는 것이 목적이다.




위험관리 체계 구축의 핵심은 위험관리의 중요 지표인 위험도를 기 보유한 보안 솔루션을 활용하여 어떻게 정량화하여 평가할 것인가에 대한 방안을 수립하는 것이다. 효율적인 위험관리 활동을 위해서는 모든 정보보호 활동이 위험관리를 중심으로 수행되어야 한다. 일반적인 위험도 평가 산식은 다음과 같으며, ‘자산 가치, 영향도, 발생가능성’을 결합하여 계산된다.

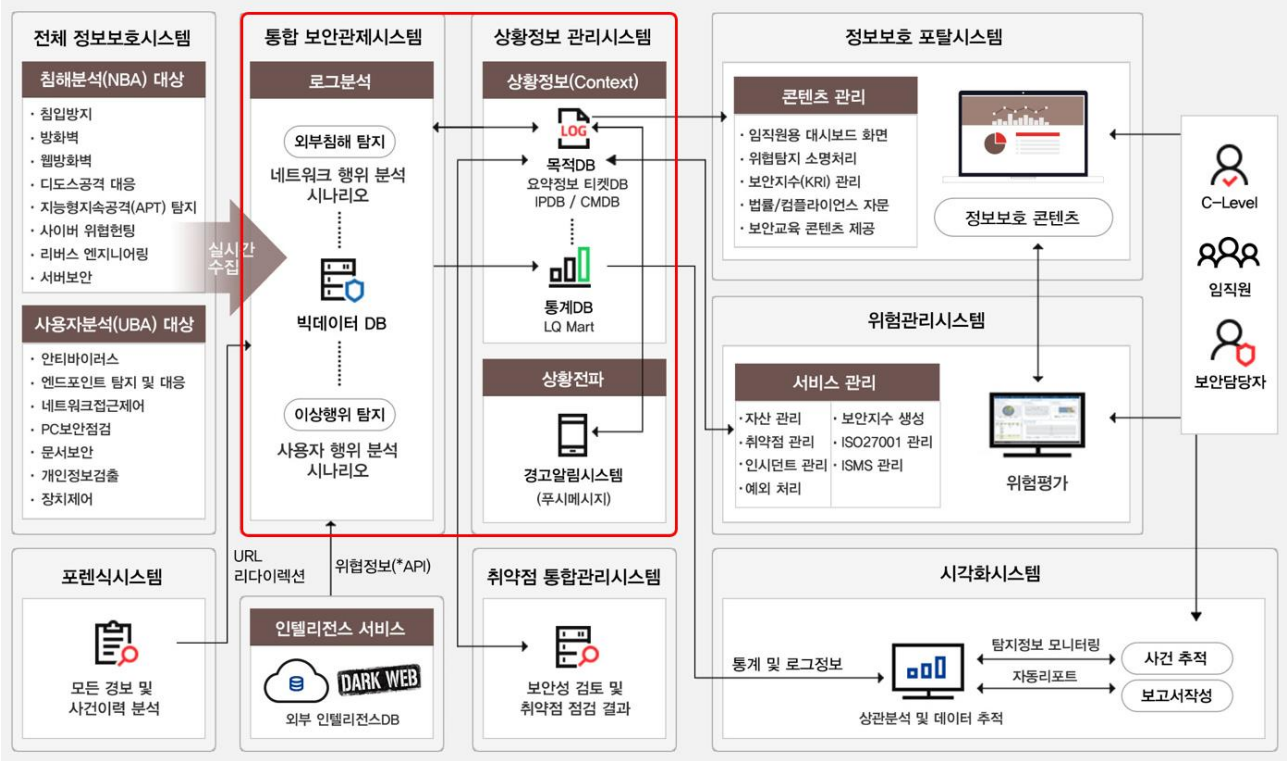
위험도(Risk Value) = 자산 가치(Asset Value) + 2 × 영향도(Impact) × 발생가능성(Likelihood)

자산 가치는 자산의 기밀성, 무결성 및 가용성 기준에 따라 자산 담당자가 정보보호 가이드 기준에 따라 산정한다. 영향도는 자산에 영향을 미치는 피해로써 공격과 취약점의 심각한 정도를 감안하여 결정된다. 발생가능성은 위협 관련 자산의 취약점과 통제의 정도, 평상 시 공격 침투 가능성을 감안하여 결정된다. 위험도를 계산하기 위한 3 대 요소에서 영향도와 발생가능성은 실시간 발생하는 위협에 영향을 받기 때문에 실시간으로 계산되어야 한다. 이를 위해 BNK 부산은행에서는 자산의 취약점과 실시간 수집되는 보안 솔루션의 로그를 결합하여 영향도와 발생가능성을 측정하도록 구성했다.

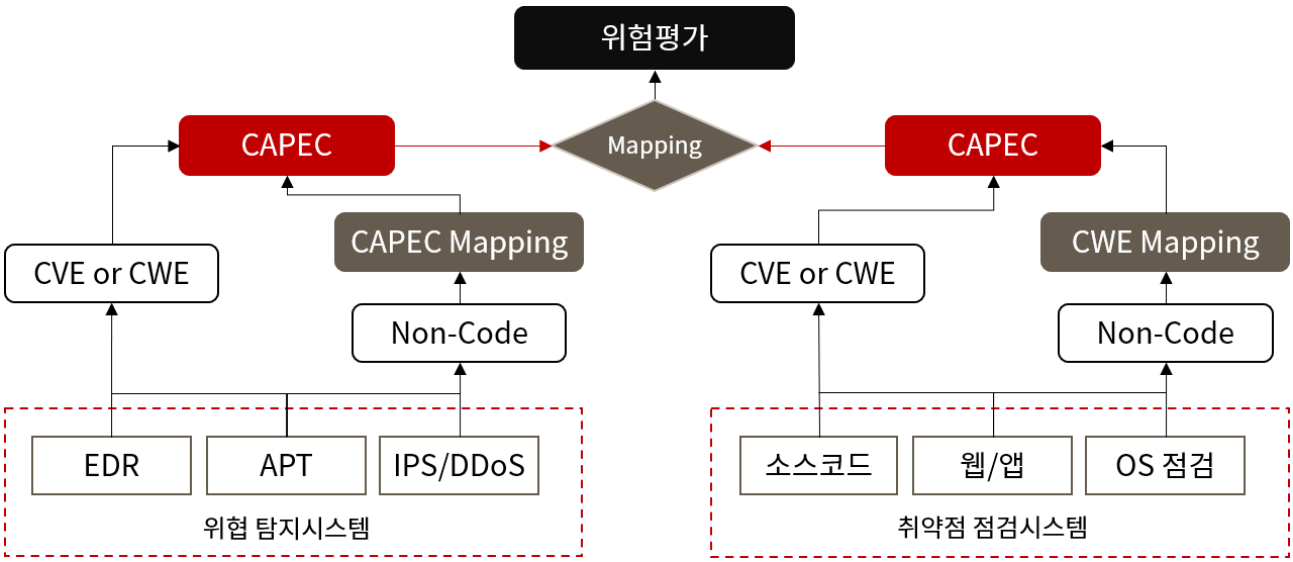


## ■ 정보보호 통합 플랫폼 구성

BNK 부산은행의 정보보호 통합 플랫폼 구성 현황은 다음과 같으며, 타 기관과 차별화된 통합 보안관제시스템 및 상황정보 관리시스템을 구축하고 있다. 특히 상황정보(Context)는 상황에 대한 올바른 판단과 활동을 위해서 행위 구성요소(대상자, 행위, 행위자)에 대한 현재 상태를 사람이 직관적으로 이해할 수 있도록 구성해야 한다.



통합 보안관제시스템에서는 행위를 기반으로 해킹 공격을 분석 및 평가하기 위해, 국제 표준인 CAPEC(Common Attack Pattern Enumeration and Classification, 사이버 공격 패턴 및 목록) 정보에 기반하여 취약점과 위협 간 관련성을 설계했다. 위협 탐지 시스템은 각자 가지고 있는 정책에 따라 위협을 탐지하게 되고, 시스템에 따라 CVE[2]나 CWE[3] 코드 등 탐지된 취약점을 매핑하여 나타내어 준다. 다음 그림과 같이 CVE 나 CWE 코드가 있으면 자동으로 CAPEC 으로 매핑되도록 구축하였고, 코드가 매핑되지 않으면 직원들이 수기로 CAPEC 코드를 매핑한다.



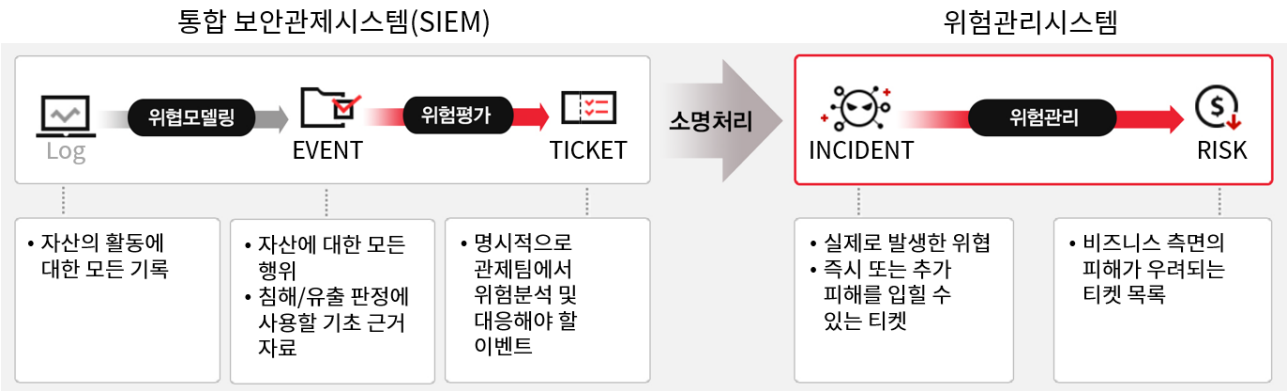

위협 탐지시스템과 유사하게 취약점에 대한 CVE 나 CWE 코드를 매핑하여 출력해 주는 취약점 점검 시스템도 있으며, 코드가 매핑되지 않으면 수기로 CWE 코드를 매핑한다. 이를 통해 탐지된 위협과 자산의 취약점을 연결시킬 수 있으며, 탐지된 위협이 자산의 취약점과 일치한다면 해당 위협에 대한 위험 평가를 높게 평가할 수 있다.

[2] CVE(Common Vulnerabilities and Exposures) : 공개적으로 알려진 컴퓨터 보안 결함 목록

[3] CWE(Common Weakness Enumeration) : 소프트웨어 및 하드웨어의 약점들을 찾아 분류해 놓은 목록

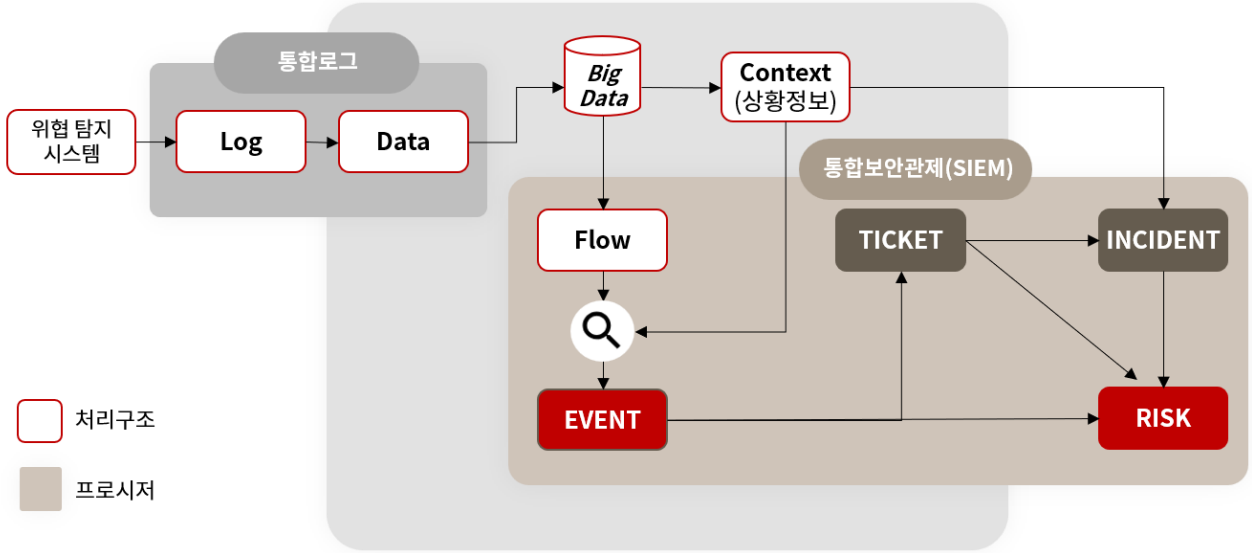
## ■ ETIR 모델 정의

통합 보안관제시스템과 위험관리시스템을 하나의 프로세스로 연동하여 관리하기 위해, BNK 부산은행은 데이터 흐름관점에서 ETIR 모델[4]을 정의했다. ETIR 모델에서는 예측 가능한 공격자의 위협 행위로부터 피해 정도를 예측하고, 인지된 사고에 대한 사실관계를 파악해 비즈니스적 위험 기준에서 위험 완화 전략을 수립한다.




위협 탐지 시스템에서 수집한 통합로그는 통합 보안관제시스템에서 ETIR 모델 관점으로 다음 그림과 같이 처리된다. 일반적인 기업과 동일하게 위협 탐지 시스템에서 생성된 로그는 파싱 및 정규화를 통해서 데이터화하고, 빅데이터 시스템에 저장하는 순으로 처리된다. 이후 대부분의 기업은 빅데이터를 역할에 기반하여(Role-based) 이벤트와 티켓으로 처리하지만, BNK 부산은행은 '상황 정보와 플로우'라는 개념을 통해서 이벤트를 생성한다.

[4] ETIR 모델(Event, Ticket, Incident, Risk) : 위협 정보로부터의 위험관리 과정을 일원화한 모델

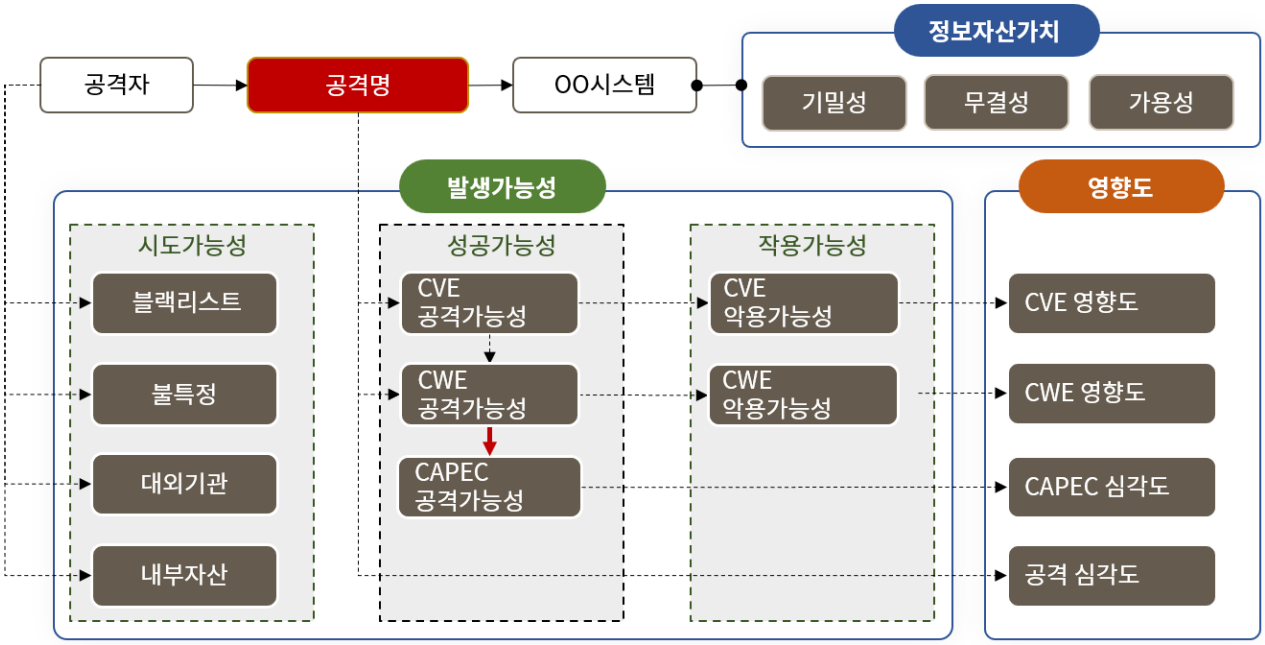



위험도는 정보자산가치와 발생가능성, 영향도를 합쳐서 계산된다. 정보자산가치는 일반적인 CIA[5](기밀성, 무결성, 가용성) 평가에 따르고, 발생가능성과 영향도는 위협과 자산 취약점 정보를 결합하여 계산한다. 발생가능성은 시도가능성과 성공가능성, 적용가능성을 결합하여 계산된다.

- 시도 가능성 : 공격자가 어떤 그룹에 속해 있는지에 따라 점수를 평가함
- 성공 가능성 : 탐지된 CAPEC 의 공격가능성 값을 활용하여 평가함
- 작용 가능성 : CVE 또는 CWE 의 악용가능성을 활용하여 평가함

[5] CIA(Confidentiality, Integrity, Availability) : 기밀성, 무결성, 가용성으로 정보의 보안 위험성을 측정하고 적절한 보안 정책을 수립하는 기준이 되는 보안의 기본 요소임

마지막으로 영향도는 CWE 영향도, CAPEC 심각도, 공격 심각도를 참고하여 평가한다. 이처럼 다양한 관점으로 발생가능성과 영향도를 측정하며, BNK 부산은행의 위험도 계산 주요 요소는 아래 그림과 같다.



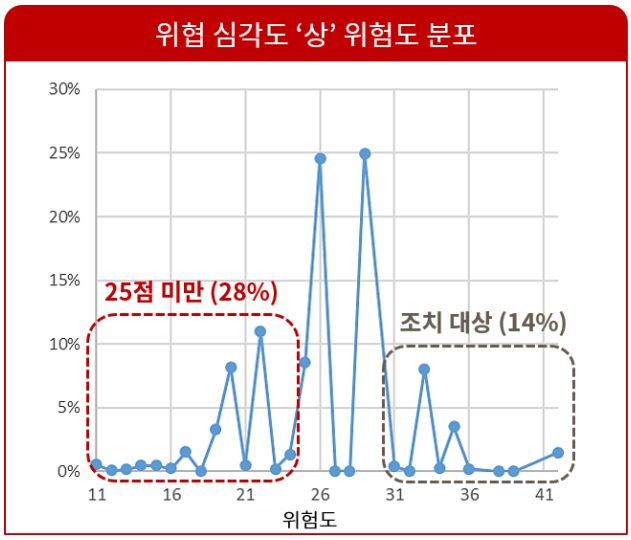

이후 이벤트에서 계산된 위험도가 수용 가능한 위험수준(DoA, Degree of Assurance)을 초과하면 티켓을 발행하며, 위험도가 DoA 미만인 경우에도 티켓과 연관된 이벤트는 분석 처리한다.

발생된 티켓은 보안 관제 직원이 소명 처리를 통해 사고 및 위험 유무를 식별하고, 이러한 소명 처리 내용은 위험관리 시스템에 축적된다. 위험관리 시스템에는 티켓뿐만 아니라 이벤트와 인시던트도 같이 저장하여 관리된다.

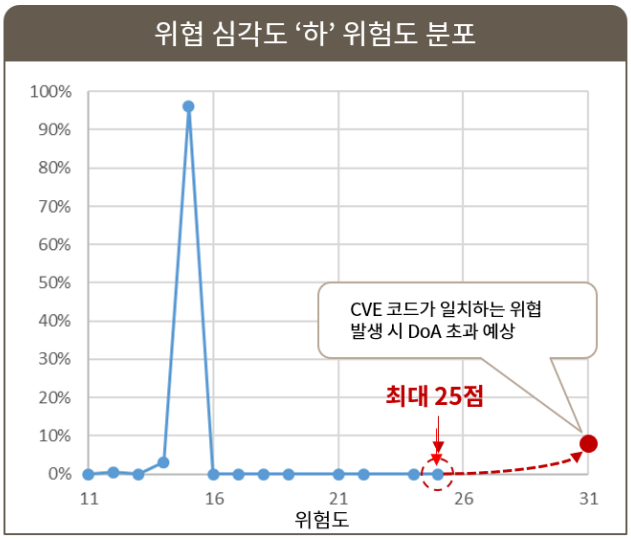
저장된 사고 및 위험 목록은 향후 사고로 발생할 가능성이 있는 경우(잔존위험) 위험 목록에 등록하고, 자산의 가치를 훼손하는 위험의 우선순위화를 통해 높은 순위부터 대응한다.

## ■ ETIR 티켓 효과

ETIR 모델에서는 실제 발생가능성이 높은 위협에 대한 티켓을 생성한다. 따라서 심각도 '상' 위협에서 위험도가 DoA 이상인 이벤트 14%에 대해서는 우선 조치한다. 또한 심각도 '하' 위협에서 가장 높은 위험도 점수는 25 점이였으며, 이는 심각도 '상' 위협 하위 28%에 해당한다. 즉, 위협의 심각도가 낮더라도 CVE 코드가 일치하는 위협이 발생하면 티켓이 생성될 수 있으므로, 위협 탐지 정확도가 증가하는 효과가 있다.

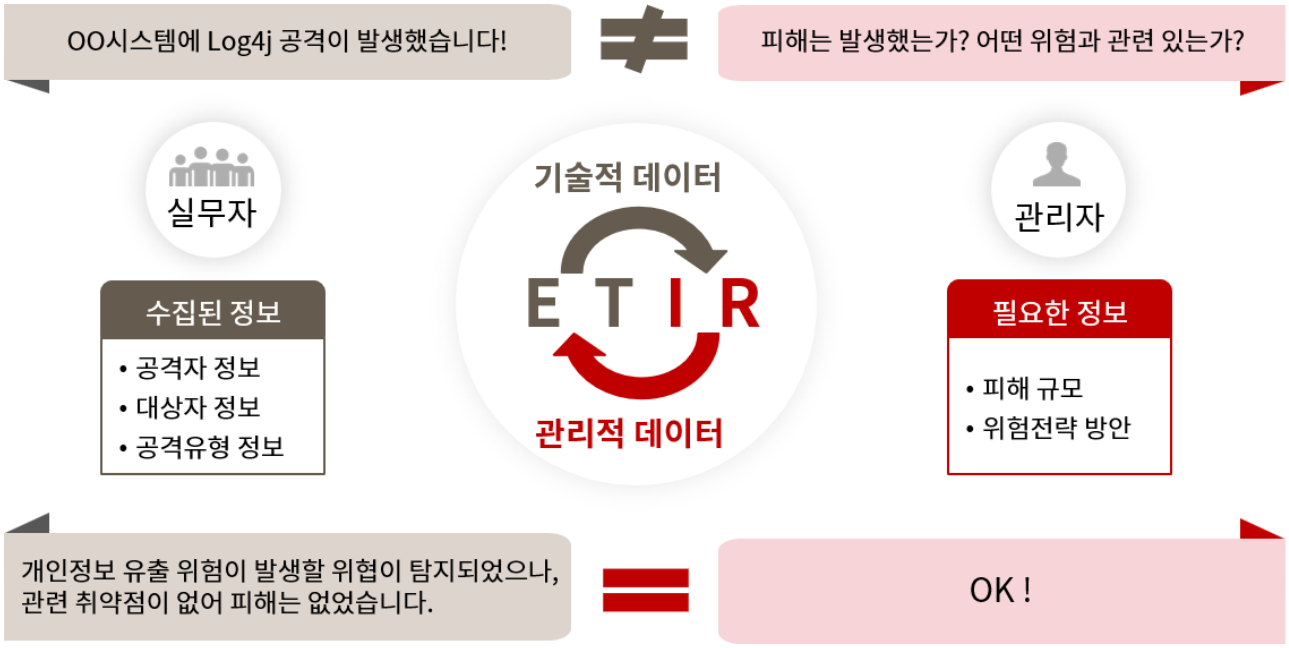



위험도 20~33 구간에서 88% 이벤트가 분포하며,
고른 분포로 인해 **정밀한 DoA 조정이 필요**함




위험도 14~15 구간에서 99% 이벤트가 분포하며,
DoA 25 이상에서는 **심각도 '중' 이상 중점 분석**이 필요함

한편 실무자와 관리자는 바라보는 관점이 다르기 때문에, 서로 의사소통을 쉽게 할 수 있는 방안을 마련하는 것이 중요하다. 즉 아래 그림과 같이 실무자는 수집된 기술적 데이터(공격자, 공격 유형 등)를 관리자에게 보고하지만, 관리자 관점에서는 관리적 데이터(피해 규모, 위험 전략 방안 등)를 필요로 한다. 이러한 소통을 원활히 제공할 수 있는 방안으로 ETIR 모델이 적합할 것으로 기대된다.




> 취약점을 찾는 것도 중요하지만, 식별된 취약점에 대한 활용방안 또한 중요하다. 취약점을 보안관제 및 컨설팅 영역에서 적합하게 활용할 수 있는 방안 가이드도 보안전문가에게 필요한 역량이라 할 수 있다.

# Special Report

## 웹 취약점과 해킹 매커니즘#9 CSRF(Cross-Site Request Forgery)

### ■ 개요

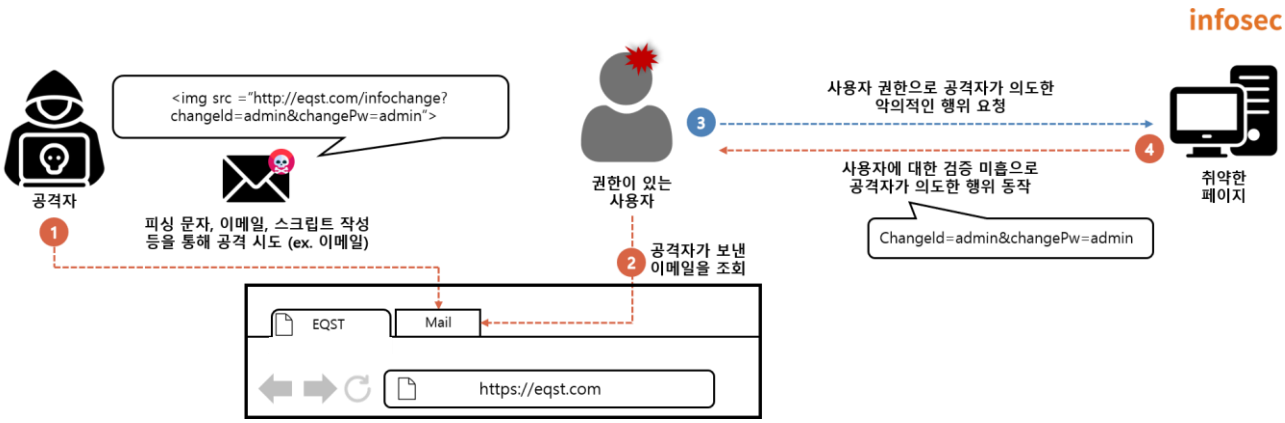
CSRF(Cross-Site Request Forgery)는 웹 취약점 공격의 하나로, 공격자가 타 사용자의 권한을 이용해 자신이 의도한 동작을 서버에 요청하게끔 유도하는 방식이다. 서버에 사용자 요청에 대한 적절한 검증 절차가 없을 때, 정상적인 요청과 조작된 요청을 구분하지 못할 경우 발생한다. 특히, 공격당한 사용자의 권한을 그대로 사용한다는 점에서 사용자의 권한 수준에 따라 피해 범위가 달라질 수 있다.

이번 Special Report 에서는 서버가 사용자의 정상적인 요청과 조작된 요청을 구분하지 못해 발생하는 취약점인 CSRF 의 개념, 동작 원리, XSS(Cross-Site Scripting)와의 비교, 보안 대책과 우회 방법에 대한 내용을 소개한다.

## ■ CSRF 동작 원리

CSRF 공격에 성공하기 위해서는 공격 대상 페이지의 로직을 분석하여 파라미터의 전송 형태와 각 파라미터의 기능을 알아야 한다. 이후 공격자는 피해자가 서버로 HTTP 요청을 보내도록 유도하는 스크립트를 작성하여, 피해자가 해당 스크립트를 읽었을 때 공격이 발생하도록 한다. 피해자는 자신의 권한으로 서버에 공격자가 의도한 악의적인 요청을 보내게 되는데, 이 때 사용자에 대한 검증 로직이 미흡할 경우 공격이 성공하게 된다. 공격 대상의 권한에 따라 공격자가 의도한 행위가 수행되기 때문에 공격 대상이 관리자일 경우, 피해를 가중시킬 수 있다.

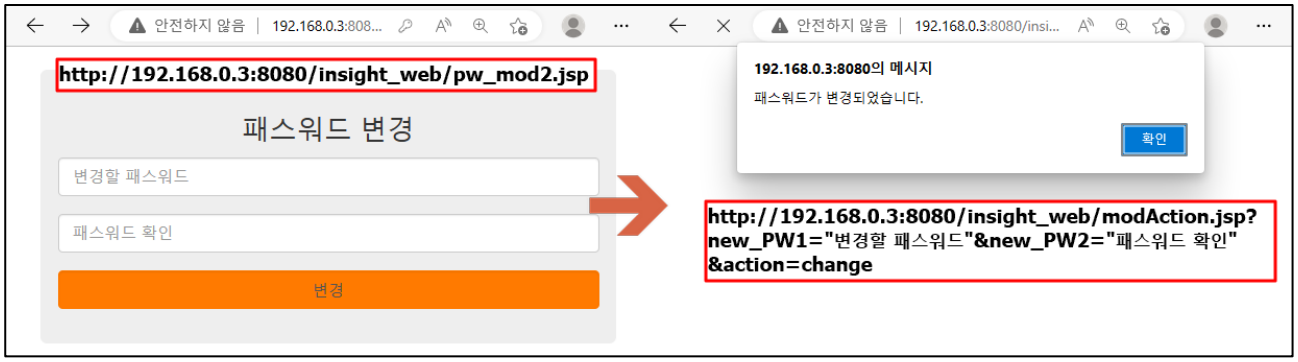
다음은 CSRF 동작 원리를 나타낸 그림이다.



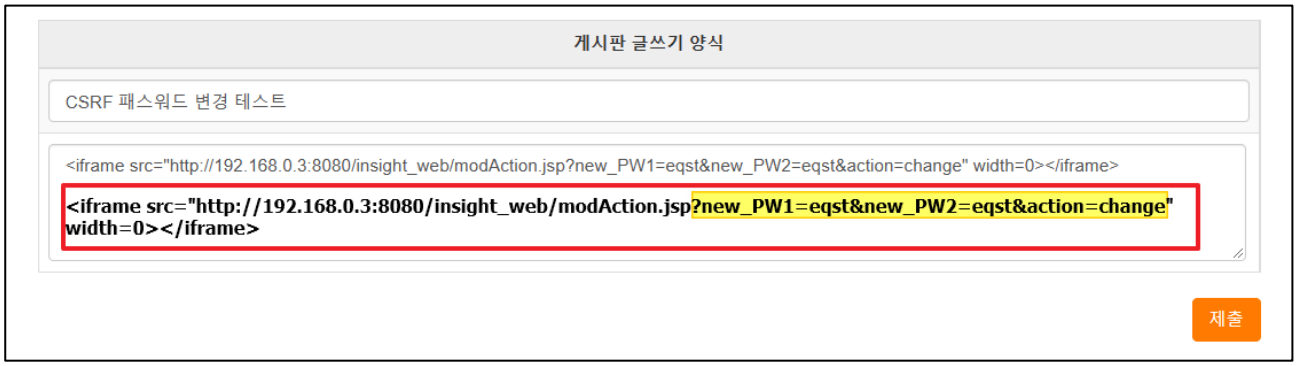

공격자는 CSRF 공격을 통해 공격 대상의 권한으로 게시글을 작성하거나 계정 정보 변경, 권한 상승 등이 가능하며 권한을 도용하여 피싱 메일 전송, 금융 거래 등의 특정 행위를 진행할 수 있다.

## ■ CSRF 공격 예시

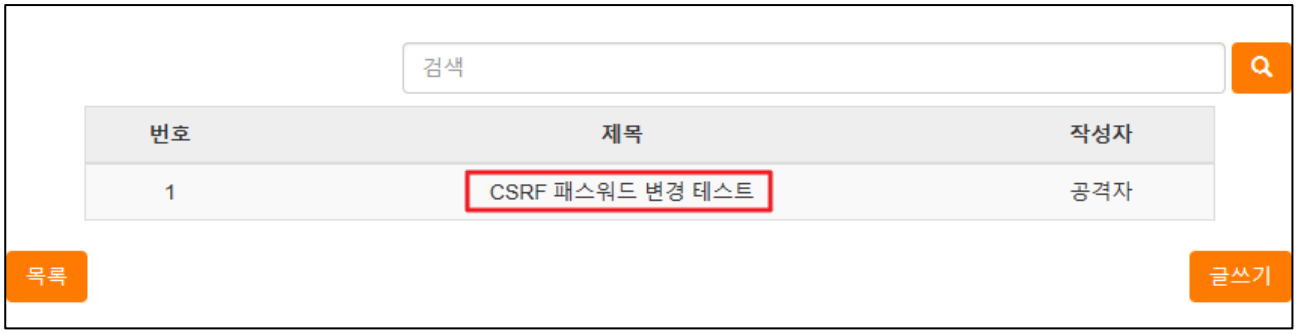
step 1) 패스워드 변경 페이지에서 패스워드 변경 요청을 보내면 아래와 같은 파라미터를 전송하여 패스워드가 변경된다.



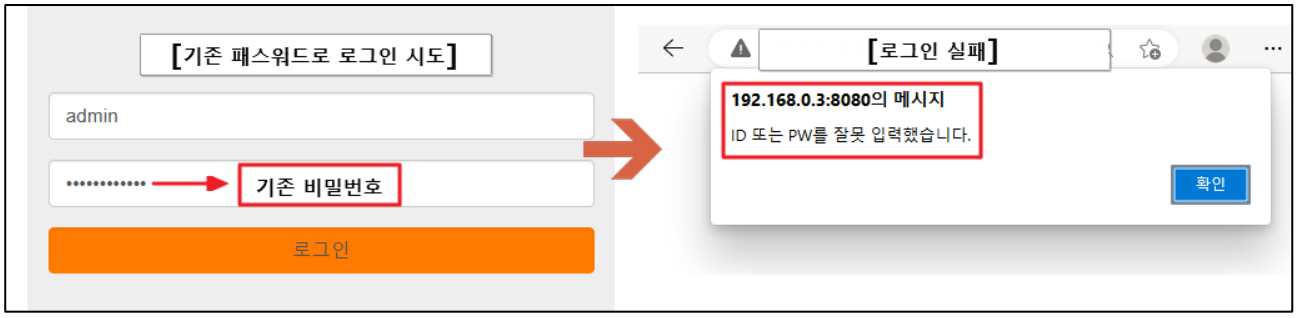

step 2) 공격자는 자신이 원하는 패스워드(eqst)로 변경을 유도하는 CSRF 스크립트를 작성하여 게시글을 등록한다.



step 3) 피해자는 공격자가 등록한 CSRF 스크립트가 삽입된 게시글에 접근한다. 피해자 측에서 패스워드 변경 스크립트가 동작하여 피해자의 패스워드가 변경된다.




step 4) 피해자(admin)가 기존 패스워드로 로그인을 시도하면 로그인에 실패하는 것을 확인할 수 있다. 또한 공격자는 피해자의 아이디와 자신이 변경한 패스워드를 통해 피해자 계정으로 로그인할 수 있다.



## ■ XSS vs CSRF

CSRF 는 공격 시 악의적인 스크립트를 사용한다는 점에서 XSS 과 연관 지어 언급되는 경우가 많지만, CSRF 와 XSS 는 다른 취약점 특성을 갖고 있다는 점에서 주의가 필요하다.

XSS 와 CSRF 의 가장 큰 차이점은 공격 대상에 있다. XSS 의 경우 불특정 다수의 클라이언트를 대상으로 공격을 진행하지만, CSRF 는 서버를 대상으로 공격을 진행한다. 즉 공격자에 의해 삽입된 스크립트가 서버에서 실행되는지, 클라이언트 측에서 실행되는지에 따라 구별할 수 있다.

공격 목적에 있어서도 차이가 있다. XSS 는 클라이언트 측에서 동작하여 클라이언트(사용자)의 정보 탈취에 목적이 있지만, CSRF 는 공격 대상의 권한을 통해 공격자가 원하는 요청을 보내는 공격이므로 공격자의 의도대로 서버가 동작하게끔 유도한다.

다음의 표는 XSS 와 CSRF 를 비교한 내용이다.

| | XSS | CSRF |
|---|---|---|
| 발생 원인 | 클라이언트가<br>웹 서버를 신용하여 발생 | 웹 서버가<br>클라이언트를 신용하여 발생 |
| 공격 대상 | 클라이언트 | 서버 |
| 공격 목적 | 쿠키/세션 탈취, 악성 사이트로의 이동 등 | 권한 도용, 권한 상승 등<br>공격자가 원하는 행위 수행 |
| 공격 행위 | 악의적인 스크립트 작성하여<br>페이지에 포함시킨 후 실행 유도 | 서버에서 제공하는 기능을<br>페이지에 포함시킨 후 실행 유도 |

## ■ 보안 대책 및 우회 방법

CSRF 공격이 가능한 이유는 요청에 대한 서버의 적절한 검증 절차가 없기 때문이다. 이에 서버로 전달되는 요청이 정상적인 요청인지, 공격자에 의한 비정상적인 요청인지에 대한 검증 로직이 중요하다.

다음의 표는 CSRF 보안 대책과 우회 방법, 보안 강도에 대한 내용이다.

| 보안 대책 | 구분 | 설명 | 보안 강도 |
|---|---|---|---|
| **XSS Filtering** | 보안 기법 | 조작된 요청 생성에 사용되는 ?, &, <, > 등의 특수문자를 필터링하여 사이트에서 스크립트가 실행되지 않도록 한다.<br>*XSS Filtering 에 대한 내용은 웹 취약점과 해킹 매커니즘#7, #8 에서 확인할 수 있다. | 미흡 |
| | 우회 방법 | Filtering 규칙이 미흡할 경우 필터링 우회 가능성이 존재한다. 또한 외부 사이트에서 실행되어 요청을 보내는 경우 공격이 발생할 수 있다. | |
| **GET/POST 구분** | 보안 기법 | GET 메소드를 활용한 URL 공격을 하지 못하도록 POST 메소드를 사용한 요청만 허용하도록 한다. | 미흡 |
| | 우회 방법 | POST 요청을 보내는 스크립트를 작성하여 우회가 가능하다. | |
| **사용자 요청 검증** | 보안 기법 | Submit Button 등을 이용하여 정상 사용자가 보내는 요청인지 검증하는 수단을 만든다. | 일부 미흡 |
| | 우회 방법 | 공격자가 검증 값을 포함한 요청을 보내도록 스크립트를 작성할 시 우회가 가능할 수 있다. | |
| **보안 토큰 사용** | 보안 기법 | 중요 기능 동작 시 랜덤으로 발행되는 보안 토큰을 발행해 토큰 값 일치 여부에 따라 요청이 동작하도록 구현한다. | 일부 미흡 |
| | 우회 방법 | 토큰 발급 페이지에 접근한 후 토큰 값을 획득해 요청을 보내면 정상 요청으로 판단하여 우회가 가능하다, | |
| **추가 인증** | 보안 기법 | 중요 기능에 대해 2 차, 3 차의 추가 인증을 필수적으로 수행하도록 구현한다. | 양호 |
| | 설명 | 비밀번호 입력, 휴대폰 인증(문자, ARS), Captcha 등으로 동작 행위자만 수행할 수 있는 인증을 추가로 구현한다. 이는 보안성이 뛰어나지만 사용자로부터 매번 인증을 수행하게 하여 사용자의 편의성이 떨어진다. | |

각 보안 대책별 상세 내용과 우회 방안은 다음과 같다.

### 1) XSS Filtering – (미흡)

#### - 보안 대책

공격자에 의해 악의적인 요청 생성에 사용되는 ?, &, 〈, 〉 등의 특수문자 필터링한다. 특수 문자 필터링의 예시는 아래와 같다.

*XSS Filtering 에 대한 내용은 웹 취약점과 해킹 매커니즘#7, #8 에서 확인할 수 있다.

| 특수문자 | < | > | ' | " | ( | ) |
|---|---|---|---|---|---|---|
| Entity | < | > | ' | " | ( | ) |

#### - 우회 기법

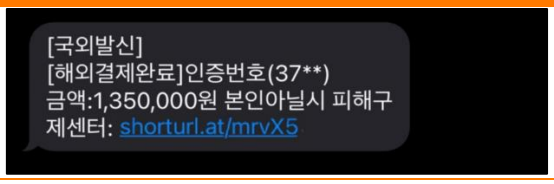
XSS 취약점을 제거하는 것은 CSRF 의 공격 범위를 줄여주는 역할만 하기 때문에 안전한 보안 대책은 아니다. 두 취약점은 전혀 별개의 취약점으로 해당 사이트에서 XSS 공격이 불가능해도 CSRF 취약점이 존재할 수 있다. 공격자는 피싱 문자, 피싱 이메일, 악의적인 스크립트가 포함된 게시글 작성을 통해 공격을 시도한다. 이에 대한 전제조건은 피해자가 로그인 등을 통해 사이트에 대한 권한을 획득하고 유지한 상태여야만 한다.

예를 들어, SK 쉴더스 사이트에 보안 담당자가 사이트에 발생할 수 있는 XSS 를 모두 조치한다고 하더라도 피싱 메일이나 SMS, 혹은 XSS 취약점이 존재하는 타 사이트를 통해 접근하여 공격하는 시나리오가 가능하기 때문에 XSS 를 막는 것은 CSRF 와 전혀 상관이 없다. 다만 CSRF 공격이 XSS 와 자주 연계되는 공격이며, XSS 가 완벽하게 막혀 있다면 아래 소개될 토큰을 통한 보안 적용이 함께 이뤄지기 때문에 이를 고려하여 보안대책으로 활용할 수 있다.

2) GET/POST 구분 – (미흡)

- 보안 대책

GET 메소드 허용 시 단순 URL 로도 CSRF 공격이 가능하며, 이는 POST 메소드를 사용하여 막을 수 있다. GET 메소드의 경우 외부로부터 변수 값에 직접적으로 접근 가능하기 때문에 공격에 취약하다. 아래는 위에서 언급한 피싱을 통한 공격 시나리오 중 피싱 문자에 해당하는 경우로, 메시지 내 링크를 클릭할 시 공격자가 의도한 악의적인 행위가 실행될 수 있다.

| 화면 | 공격 스크립트 |
| --- | --- |
|  | "https://eqst.com/infochange?changeId=admin&changePw=admin"<br>(*URL 은 shorturl 을 통해 변형되었다.) |

- 우회 기법

공격자는 HTML 태그나 javascript 사용이 가능한 게시판과 같은 곳에서 POST 메소드로 요청을 보내도록 스크립트를 작성하여 공격 대상이 해당 요청을 전송하도록 유도한다. Iframe 을 사용할 경우 창 크기를 0 으로 만들어 공격자는 사용자가 알지 못하도록 요청을 숨겨 보낼 수 있다. 아래는 피해자의 계정 정보를 공격자가 원하는 계정 정보로 바꾸도록 하는 스크립트 예시이다.

```
<iframe width="0" height="0" border="0" name=" stealthframe" id="stealthframe"
style="display: none;"></iframe>
<body onload="document.csrf_form.submit();">
<form method="POST" name="csrf_form" action="https://eqst.com/infochange"
target="stealthframe">
<input type ="hidden" name="changeId" value="admin">
<input type ="hidden" name="changePw" value="admin">
</form>
```

### 3) 사용자 요청 검증 – (일부 미흡)

#### - 보안 대책

서버로 전송되는 요청이 정상적인 사용자에 의한 요청인지 확인하는 검증 로직을 추가하는 방법이다. 검증 로직은 Submit Button, Alert 창 등을 이용해, 피해자 몰래 실행되는 것이 아닌 알림창을 확인하고 실행하도록 로직을 변경하는 것이다. 이 보안대책을 적용하면 아래와 같이 로직 실행 확인 용도의 파라미터가 추가된다.

| 검증 값을 추가하기 전 | 검증 값을 추가한 후 |
|---|---|
| changeId=admin& changePw=admin | changeId=admin& changePw=admin<br>&**confirm=ok** |

#### - 우회 기법

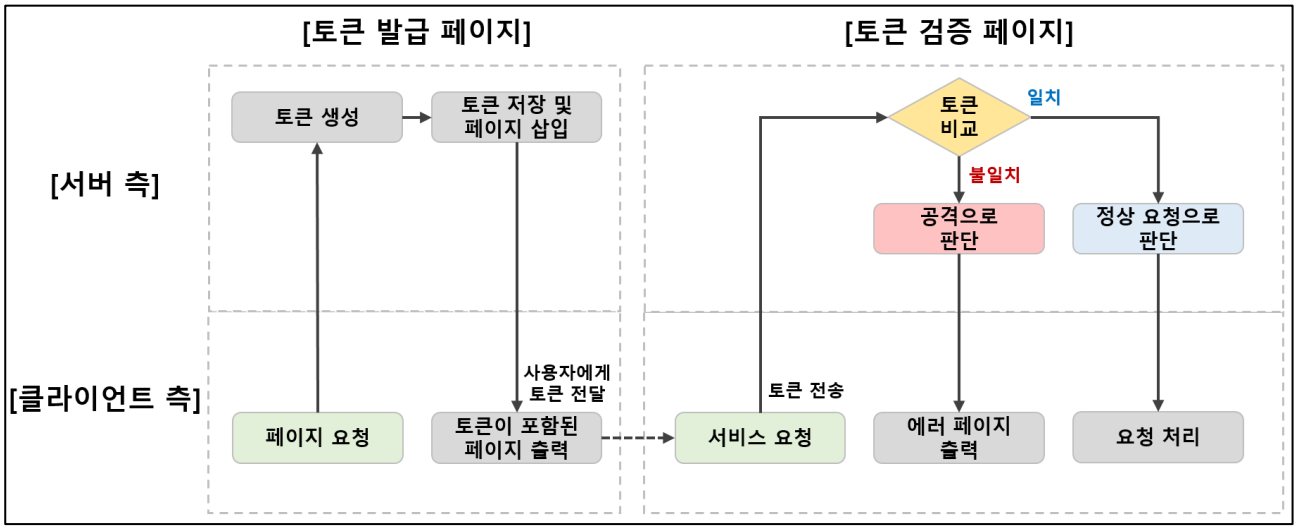
공격자 입장에서는 alert창을 확인한 이후에 파라미터명/값만 알게 되면 파라미터를 미리 설정할 수 있다. 이 값이 고정이고, Y,N,ok 등과 같은 파라미터를 쓸 경우 쉽게 유추가 가능하다. 특히 오픈소스로 개발되었거나 무료게시판을 사용하는 사이트는 공격자가 미리 구조를 파악하여 공격하기 때문에, 피해가 더 자주 발생하는 편이다.

```
<iframe width="0" height="0" border="0" name=" stealthframe" id="stealthframe"
style="display: none;"></iframe>
<body onload="document.csrf_form.submit();">
<form method="POST" name="csrf_form" action="https://eqst.com/infochange"
target="stealthframe">
<input type ="hidden" name="changeId" value="admin">
<input type ="hidden" name="changePw" value="admin">
<input type ="hidden" name="confirm" value="ok">
</form>
```

4) 보안 토큰 사용 – (일부 미흡)

– 보안 대책

사용자 요청 검증이 고정값이고 유추하기 쉬워 우회가 가능하다면, 이를 고도화한 것이 토큰 방식이다. 보안 토큰의 경우 토큰 '발급' 페이지에서 공격자가 유추할 수 없는 랜덤값을 매번 생성하고, 요청 시마다 변경되며, 토큰 '검증' 페이지에서 유효성 검사를 한다. 매번 한 번만 사용하고 만료처리를 하여 재사용을 방지하는 보안 로직이다. 예를 들어 관리자만 접근이 가능한 사용자 권한 조회 페이지와 사용자 권한 수정 페이지가 존재한다고 가정하면, 권한 조회페이지에서 토큰을 발급하고, 수정페이지에서 토큰을 검증하는 형태다. 이렇게 보안대책을 설정할 경우, 게시판에서 토큰값이 없는 요청이 갑자기 온다 하더라도 로직을 실행시키지 못하기 때문에 안전하게 사용할 수 있다.




[토큰 동작 방식]

- 우회 기법

공격자가 공격을 수행하기 위해서는 토큰 발급 페이지에 접근이 가능해야 한다. 토큰 발급 페이지에서 토큰을 발급한 후 생성된 토큰을 이용하는 스크립트를 작성해 공격을 수행한다. 토큰 획득시에 다른 사이트에서 XSS 를 통한 접근이 불가능하여 같은 사이트 내에서 발생하는 XSS 공격을 이용해 토큰을 획득해야 한다.

토큰 우회 로직은 다음과 같다.

(1) 토큰 발급 페이지에서 토큰을 생성해 변수로 받아온다.
(2) 서버로 보내는 요청에 변수로 받아온 토큰을 자동으로 제출하도록 공격 스크립트를 작성한다.
(3) 같은 사이트 내에서 XSS 공격과 연계해 스크립트를 실행시킨다.
(4) 토큰 검증 페이지에서 정상요청으로 판단되어 요청이 처리된다.

아래는 실제 공격에 사용되는 페이로드이다.

```
<script>
var readToken = function(){
   var doc = document.getElementById("frame1").contentDocument
   var token = doc.getElementsByName("csrf_token")[0].getAttribute("value");

   var frame2 = document.getElementById("frame2");
   frame2.src = "http://eqst.com/updateTokenAuth?loginId=admin&csrf_token=" + token;
}
</script>

<iframe id = "frame2" >
</iframe>

<iframe id = "frame1" onload="readToken()" src="http://eqst.com/authpage">
</iframe>
```

위 로직이 토큰 '발급' 페이지에서 토큰값을 받아서 토큰 '검증' 페이지 요청 시 파라미터에 자동제출 하도록 스크립트를 작성하는 것인데, 이 공격기법에는 제약조건이 하나 있다. SOP(Same-Origin-Policy, 동일 출처 정책) 설정인데, 같은 사이트 내 자원만 참조가 가능한 설정이다.

예를 들면, EQST 사이트에서 EQST 사이트 자원을 참조하거나 훔치는 것은 되지만, EQST 사이트에서 타 사이트 자원을 훔치거나 참조하는 것은 불가능하다는 의미다. 따라서 스크립트가 동작해야 하는 사이트 내에서 토큰을 훔쳐서 자동 제출할 수 있어야만 공격이

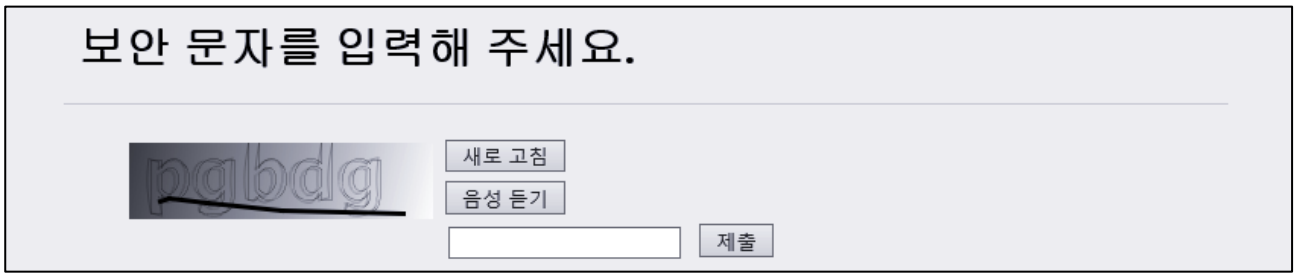
가능하다고 볼 수 있다. 다만, XSS 취약점이 존재하지 않는 사이트에서는 토큰 방식이 안전할 수 있기 때문에, XSS 와 토큰방식을 동시에 사용한다면 안전할 수 있다.

5) 추가 인증 – (양호)

- 보안 대책

지금까지 살펴본 보안 대책들은 공격자가 보안 코드를 우회하거나 검증 로직 값을 자동 제출함으로써 우회가 가능했다. 하지만 근본적으로 해커가 예측할 수 없는 값, 예를 들어 사용자 본인만 알고 있는 비밀번호나 SMS 인증 등으로 2Factor 구조 설계를 한다면 CSRF 공격은 절대로 성립될 수 없다.

따라서 2 차 인증은 사용자 본인만 알 수 있는 값으로 추가인증을 구현하거나, 사용자 본인이 수행하지 않은 요청을 방지하는 Captcha 를 통해 보안 대책을 적용할 수 있다. Captcha 는 해커가 미리 값을 설정할 수 없기 때문에 안전하다.




[Captcha 사용 예시]

## ■ 맺음말

CSRF 는 중요 로직에 한해서만 취약점으로 분류되고, 중요도에 따라서 보안대책을 다르게 적용할 수 있다. 예를 들어 100 원만 이체하는 경우 Token 과 XSS 를 활용하는 정도로 보안 적용이 가능하지만 100 만원을 이체하는 경우 추가 인증을 구현함으로써 중요도에 따른 대책을 적용시켜야 한다.

사용자 편의성과 보안성을 동시에 만족할 수 있다면 좋겠지만, CSRF 의 경우 그러기 쉽지 않은 탓에 어떤 보안대책을 어떻게 구성할지는 담당자와 개발부서가 협의하여 페이지마다 적절한 보안대책을 적용해야 한다.

이어지는 2 월호 Special Report 에서는 서버 측에서 이루어지는 요청을 변조하여 공격자가 의도한 서버로 임의의 요청을 하게 만드는 공격인 SSRF(Server-Side Request Forgery)에 대한 내용을 소개한다.

# Research & Technique

## ProxyNotShell, Microsoft Exchange Server 원격 코드 실행 취약점 (CVE-2022-41040, CVE-2022-41082)

### ■ 취약점 개요

2022 년 8 월, 국내를 비롯해 전 세계 많은 기업에서 사용하는 메시징, 협업 소프트웨어 제품인 Microsoft 의 MS Exchange Server 에서 공격자가 유효한 메일 서버 계정을 이용해 시스템에 원격 코드 실행을 가능하게 하는 취약점이 발견됐다.

ProxyNotShell 이라 불리는 이 취약점은 MS Exchange Server 의 프론트엔드 서비스에서 제공하는 autodiscover[1] 서비스를 악용하여 유저가 직접 접근할 수 없는 백엔드의 파워 쉘을 이용하여 원격 코드 실행을 가능하게 하는 취약점이다.

인터넷상에서 공개된 MS Exchange Server 는 Shodan 과 같은 OSINT 검색 엔진을 통해 취약한 서버 운영 정보를 쉽게 확인할 수 있다. Shodan 검색 결과 2023-01-04 기준 전 세계에는 186,769 개의 MS Exchange Server 가 존재한다. 그리고 인터넷 보안 비영리 단체인 Shadowserver Foundation 은 ProxyNotshell 에 취약한 서버가 2023-01-02 기준 60,865 개 존재한다고 발표했다. 더욱이 최근 랜섬웨어 조직에서도 해당 취약점을 활용하고 있어 취약한 버전의 MS Exchange Server 사용 시 각별한 주의가 필요하다.

[1] MS Exchange Server autodiscover 프론트엔드란 Client access services 라고 불리며, Active Directory 도메인에 가입된 컴퓨터에게 내부/외부의 필요한 정보를 검색, 연결할 수 있는 서비스다. 외부에서 사용자들이 쉽게 서비스를 접근하도록 OWA(Outlook Web App)를 제공하는데 이를 autodiscover 프론트엔드라고 부른다.

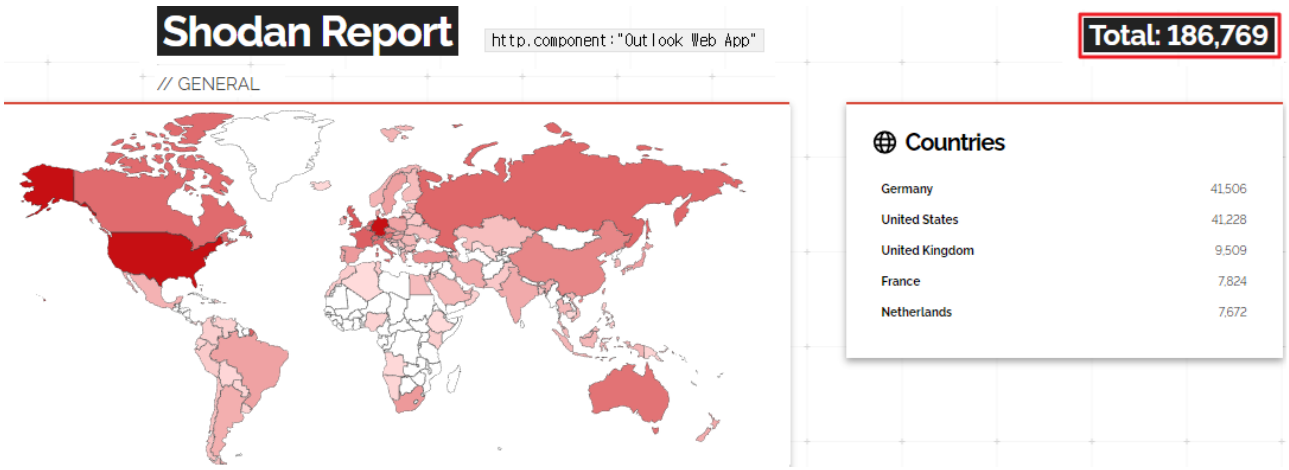



그림 1 전 세계 MS Exchange OWA(Outlook Web App) Server

## ■ 영향받는 소프트웨어 버전

ProxyNotShell 에 취약한 소프트웨어는 다음과 같다.

| S/W 구분 | 취약 버전 |
|---|---|
| MS Exchange Server | 2013, 2016, 2019 version |

※ 2022 년 11 월 8 일 이후 업데이트된 보안 패치를 적용하지 않은 서버는 취약하다.

## ■ 공격 시나리오

ProxyNotShell 를 이용한 공격 시나리오는 다음과 같다.

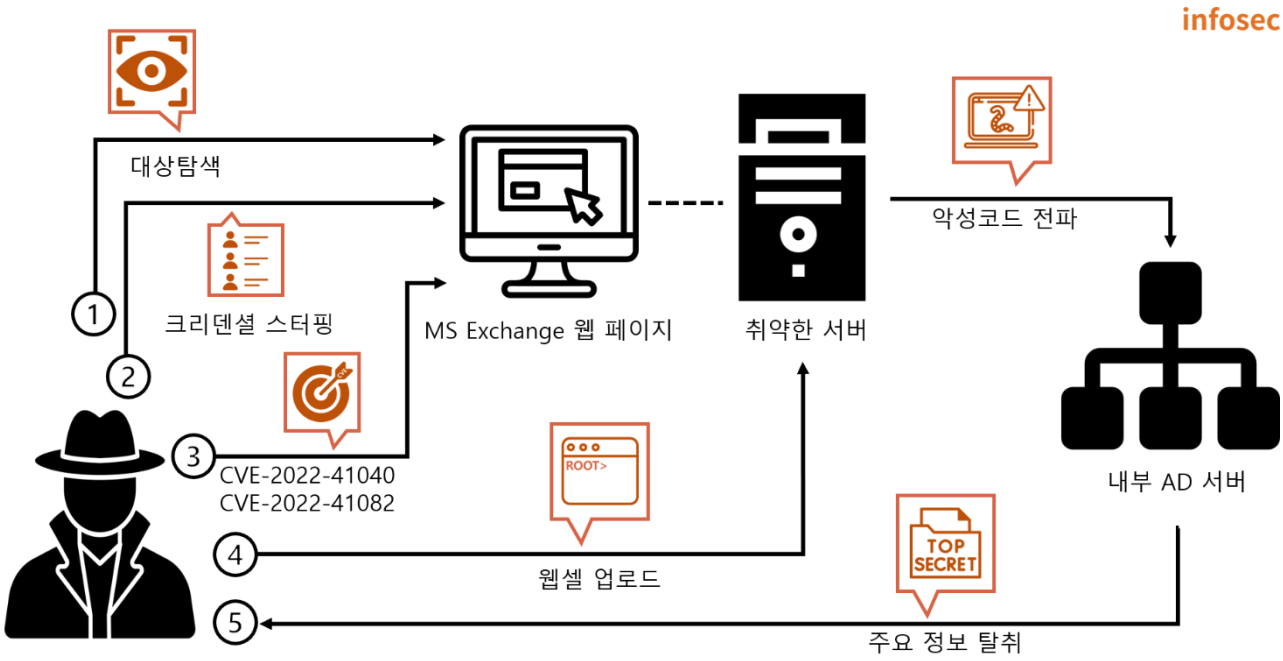



그림 2 공격 시나리오

① 공격자는 ProxyNotShell에 취약한 대상을 탐색
② 공격자는 크리덴셜 스터핑, 브루트 포싱, 피싱 등을 활용해 계정 정보 확보
③ 공격자는 MS Exchange Server 에 접속 가능한 사용자 계정을 통해 CVE-2022-41040(SSRF) 실행
④ 공격자는 CVE-2022-41082(RCE)를 악용한 백도어/웹쉘 설치, 악성코드 및 랜섬웨어 배포
⑤ 공격자는 피해자 PC 의 제어권 획득, 주요 정보 탈취 등 공격 수행

## ■ 테스트 환경 구성 정보

테스트 환경을 구축하여 ProxyNotShell 의 동작 과정을 살펴본다.

| 이름 | 정보 |
|---|---|
| **피해자** | Window Server 2012 R2<br>AD server (계정 정보: Administrator/EQST12#$)<br>MS Exchange Server 2016<br>DNS (eqstlab.local) |
| **공격자** | Window 10 Pro<br>AD user (계정 정보: user1/EQST12#$) |

## ■ 취약점 테스트

Step 1. PoC 테스트

테스트를 위한 PoC 가 저장된 GitHub URL 은 다음과 같다.
- URL: https://github.com/testanull/ProxyNotShell-PoC

1) 다운로드 받은 PoC 코드를 활용하여 calc.exe 를 실행하도록 원격 코드 실행 시도

| 명령어 | $ python poc_aug3.py https://eqstlab.local user1 EQST12#$ calc.exe<br>$ python poc_aug3.py [도메인 경로] [ID] [Password] [명령어] |
|---|---|

```
C:₩Users₩user1₩ProxyNotShell-PoC-main>python poc_aug3.py https://eqstlab.local user1 EQST12#$ calc.exe

[+] Create powershell session
[+] Got ShellId success
[+] Run keeping alive request
[+] Success keeping alive
[+] Run cmdlet new-offlineaddressbook
[+] Create powershell pipeline
[+] Run keeping alive request
[+] Success remove session
```

그림 3 RCE 명령 전송

2) 피해자(MS Exchange Server 2016) 서버에서 calc.exe 동작 확인

```
오후 3:28:23.4724772  cmd.exe        12760  CloseFile   C:₩Windows₩System32₩calc.exe          SUCCESS
오후 3:28:23.4727315  MSExchangeH... 13060  WriteFile   C:₩Program Files₩Microsoft₩Exchang... SUCCESS
오후 3:28:23.4792873  w3wp.exe       9576   Thread Exit                                        SUCCESS
오후 3:28:23.4803896  calc.exe       4912   Load Image  C:₩Windows₩System32₩calc.exe          SUCCESS
```

그림 4 RCE 동작 확인

## ■ 취약점 상세 분석

ProxyNotShell 은 CVE-2022-41040 과 CVE-2022-41082 의 연계 취약점이다.

### Step 1. CVE-2022-41040

MS Exchange Server 는 사용자가 직접 접근 가능한 프론트엔드(Client access services)와 사용자가 직접 접근이 불가능한 백엔드(Backend Services)로 구성된다. ProxyNotShell 의 첫 번째 공격 단계는 CVE-2022-41040 을 활용한 SSRF [2] 취약점으로 OWA(Outlook Web App)의 autodiscover 프론트엔드를 악용한다. autodiscover 서비스란 API 엔드포인트를 이용하여 내부 서버에 필요 기능을 요청하는 서비스다. 해당 취약점은 autodiscover 프론트엔드에서 PowerShell API 엔드포인트 URL 경로를 PowerShell 접근 경로로 변경하여 접근 불가능한 백엔드의 액세스 권한을 획득한다.

※ PowerShell API 엔드포인트 경로는 https://%exchangeserverdomiaon%/powershell 이다.

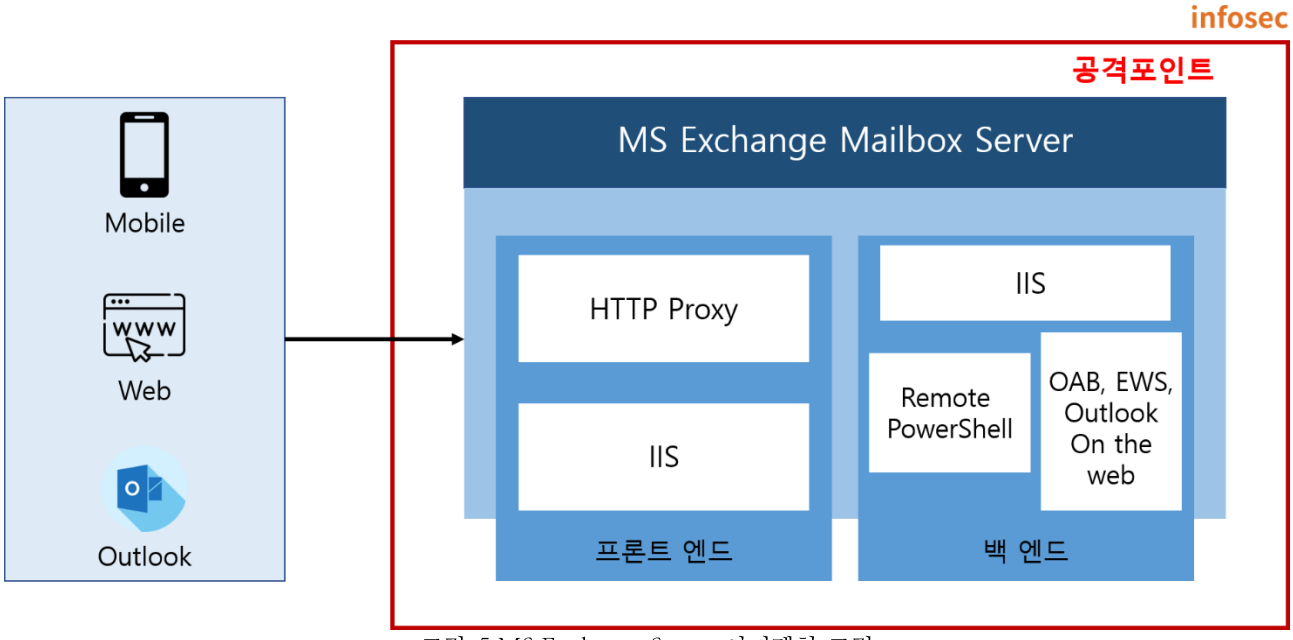



그림 5 MS Exchange Server 아키텍처 그림

[2] SSRF(Server-Side Request Forgery)의 약자로 서버 측에서 위조된 요청을 보내도록 하는 취약점으로, 내부 서버의 정보를 노출시키거나 서비스 거부 공격을 할 수 있다.

CVE-2022-41040 취약점은 2021 년 공개된 MS Exchange Server 취약점인 ProxyShell(CVE-2021-34473, SSRF)의 보안대책을 우회한 취약점이다.

- ProxyShell 란?

CVE-2021-31207(MS Exchange Server 보안 기능 우회 취약점), CVE-2021-34523(MS Exchange Server 권한 상승 취약점), CVE-2021-34473(MS Exchange Server RCE 취약점)의 연계 취약점으로 인증되지 않은 사용자가 원격 코드 실행이 가능한 취약점이다.

ProxyShell 취약점은 autodiscover 를 이용하여 URL 을 검색할 때, 필터링이 미흡한 백엔드 URL 을 악용하여 백엔드로 접근하는 SSRF 취약점이다. 2021 년 7 월에 인증 요소를 추가함으로써 백엔드의 액세스를 제한하여 악용할 수 없도록 패치 되었다. ProxyShell 의 보안패치가 적용된 서버에서 공격을 시도해 본 결과, 그림 6 과 같이 응답 값으로 401 Unauthorized 에러를 반환한다. 하지만, 그림 6 처럼 응답 값 부분에 인증 방식 NTLM(NT Lan Manager)[3]과 Basic[4]이 노출되어 이를 이용한 공격이 가능하다.

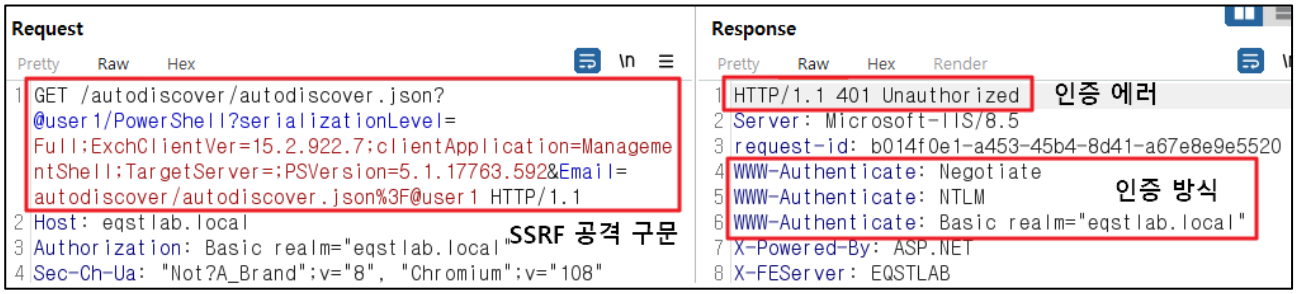



그림 6 ProxyShell 401 인증오류

[3] NTLM 이란 NT Lan Mange 의 약어로, 윈도우에서 제공하고 있는 인증 프로토콜 중 하나로 Challenge-Response 의 인증 프로토콜 방식으로 암호를 전송하지 않고, Exchange 호스트에 연결할 수 있다.

[4] Basic 이란 웹 서버에 보낼 아이디와 암호를 Base64 방식으로 인코딩 후 전달하여 인증하는 방식으로 Basic 인증을 이용해 EWS(Exchange Web Service), 원격 PowerShell, autodiscover 등을 사용할 수 있다.

따라서 SSRF 페이로드를 포함하여 ProxyNotShell 공격을 위해 그림 7 처럼 유효한 인증 정보인 user1:EQST12#$를 base64 로 인코딩하여 Request 를 보낸다.

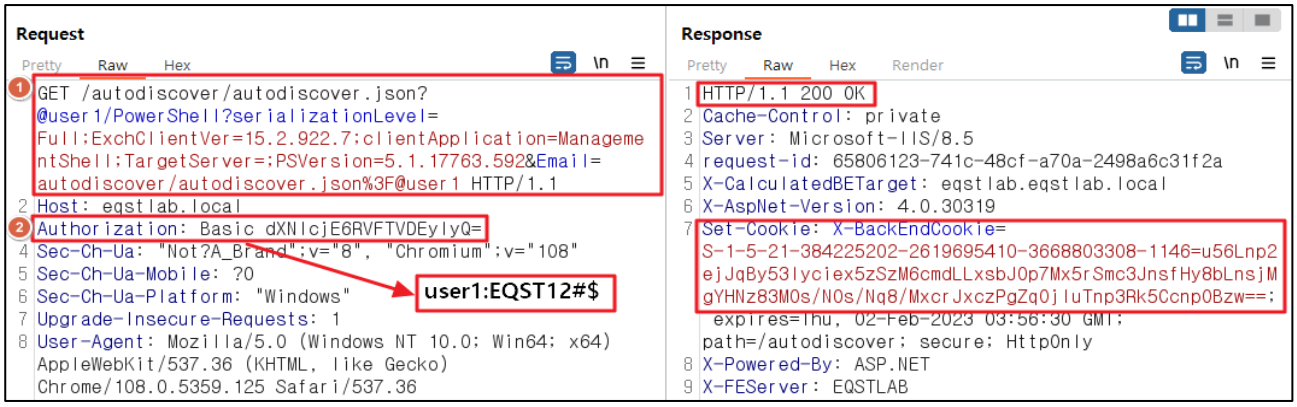



그림 7 ProxyNotShell 이용 PowerShell 내부 접근

유효한 인증 정보를 헤더에 추가하면 ProxyShell 보안대책이 우회 가능하여, autodiscover 를 악용한 SSRF 가 여전히 동작하여, 접근 권한이 저장된 백엔드의 쿠키 값을 얻을 수 있다.

Step 2. CVE-2022-41082

Exchange PowerShell 백엔드 접근에 성공했으므로, PowerShell Remoting5 에서 역직렬화 취약점을 악용해 원격 코드 실행이 가능한 취약점인 CVE-2022-41082 를 실행한다. 이 공격에는 직렬화(Serialization)와 역직렬화(Deserialization)가 사용된다. 직렬화란 객체를 네트워크를 통해 다른 곳으로 전송할 수 있는 형식이나 파일에 저장할 수 있는 데이터인 바이트 또는 스트림으로 변환시키는 것을 의미한다. 역직렬화란 직렬화한 데이터를 본래의 객체로 되돌리는 변환을 의미한다. CVE-2022-41082 는 공격자가 악의적으로 조작된 직렬화 데이터를 저장한 뒤, 역직렬화 과정에서 원격 코드 실행이 가능한 취약점이다.

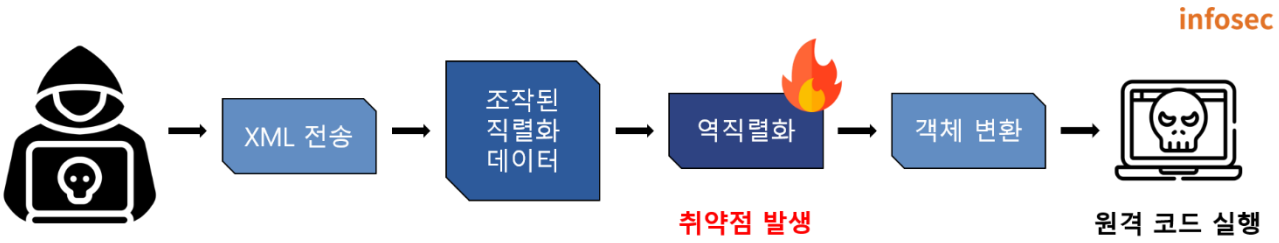



그림 8 CVE-2022-41082 역직렬화 취약점 동작 과정

[5] PowerShell Remoting 이란 사용자가 원격으로 PowerShell 명령을 실행할 수 있는 서비스이다.

Step 3. PoC 동작 분석

아래의 PoC 는 SSRF 를 통해 공격을 성공한 뒤, 객체를 루트에 반환하는 메소드인 XamlReader 를 통한 역직렬화 과정에서 페이로드를 객체로 Xaml 을 로드하여 RCE 를 실행하는 코드의 일부이다.


그림 9 PoC RCE 주요 페이로드

역직렬화 허용을 위해 1 번과 같이 TargetTypeForDeserialization 을 설정하고, 2 번과 같이 System.Exception[6] 로 Type 유형을 선언한다. Type 을 Exception 으로 설정한 이유는 취약한 BinaryFormatter 를 포함된 클래스인 SerializationTypeConverter 를 활용하면 RCE 페이로드 부분을 악용할 수 있기 때문이다. MS Exchange Server 에서 객체를 직렬화 할 때는 UnitySerializationHolder[7] 클래스를 활용한다. 따라서, 〈MS〉태그(4 번)에 객체를 직렬화 할 때 필요한 UnitySerializationHolder 와 XamlReader 를 인코딩하여 포함시켜 직렬화하면, 역직렬화 될 때 RCE 페이로드 문자열이 저장된 〈S〉태그를 객체로 활용할 수 있다.

[6] System.Exception 클래스는 애플레이케이션 실행 중 발생하는 예외를 다루기 위한 클래스로 .NET 에 포함되어 있다. https://learn.microsoft.com/ko-kr/dotnet/api/system.exception?view=net-7.0 에 정의되어 있다.

[7] UnitySerializationHolder 클래스는 라이브러리 "mscorlib"에 포함되어 있으며, MS Exchange Server 에서 직렬화를 다룰 때 사용하는 클래스이다.

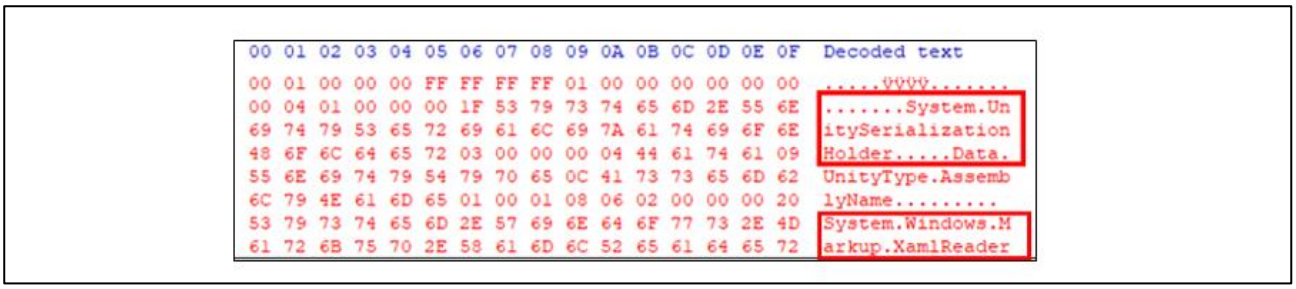



그림 10 디코딩 된 값

정리하자면 UnitySerializationHolder 와 XamlReader 를 직렬화하면, 역직렬화 시 XamlReader 에서 페이로드를 객체로 사용하여 실행하고, Xaml 이 로드되어 원격 코드 실행이 가능하다.

```
  <TypeConverter>
    <TypeName>Microsoft.Exchange.Data.SerializationTypeConverter</TypeName>
  </TypeConverter>
</Type>
<Type>
  <Name>Deserialized.Microsoft.Exchange.Data.Unlimited`1[[Microsoft.Exchange.Data.EnhancedTimeSpan, Microsoft.Exchange.Data,
       Version=15.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]]</Name>
  <Members>
    <MemberSet>
      <Name>PSStandardMembers</Name>
      <Members>
        <NoteProperty>
          <Name>TargetTypeForDeserialization</Name>
          <Value>Microsoft.Exchange.Data.Unlimited`1[[Microsoft.Exchange.Data.EnhancedTimeSpan, Microsoft.Exchange.Data,
               Version=15.0.0.0, Culture=neutral,  PublicKeyToken=31bf3856ad364e35]]</Value>
        </NoteProperty>
      </Members>
```

그림 11 xml 의 등록된 모습

역직렬화 시 페이로드의 {CMD}에 calc.exe 가 객체로 전달되어 cmd.exe /c calc.exe 가 완성되어 RCE 가 실행된다.

```
<![CDATA[<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:System="clr-namespace:System;assembly=mscorlib"
 xmlns:Diag="clr-namespace:System.Diagnostics;assembly=system"><ObjectDataProvider x:Key="LaunchCalch"
 ObjectType="{{x:Type Diag:Process}}" MethodName="Start">
 <ObjectDataProvider.MethodParameters><System:String>cmd.exe</System:String><System:String>/c {CMD}
 </System:String> </ObjectDataProvider.MethodParameters> </ObjectDataProvidr> </ResourceDictionary>]]>
```

그림 12 전달객체

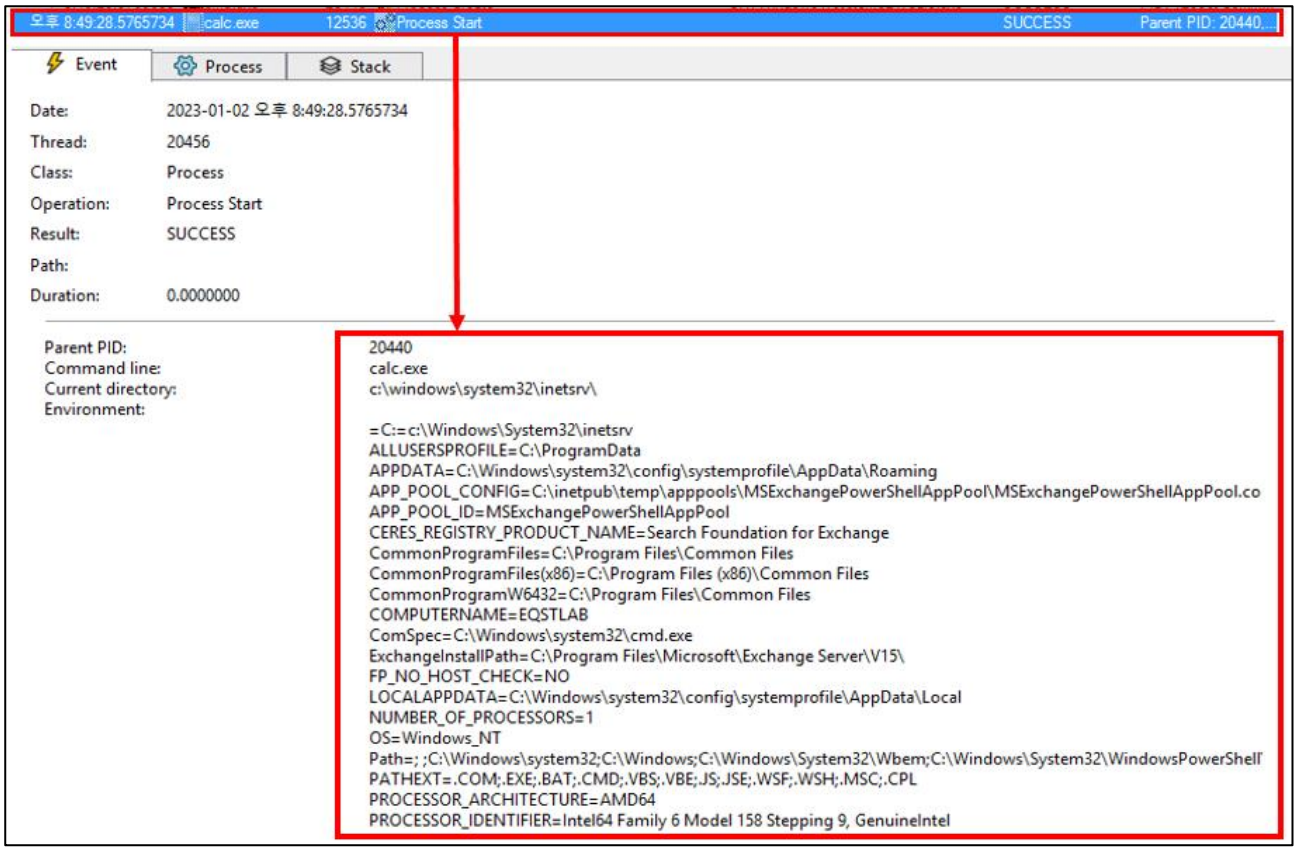



그림 13 RCE 실행 결과 모습

다음은 Python PoC 의 동작 과정의 설명이다. RCE 를 실행하기 위해서 MS Exchange Server 주소, ID, 비밀번호와 실행할 명령어를 인자 값으로 전달하면, 주소를 기반으로 autodiscover 를 통해 PowerShell 엔드 포인트 URL 에 base64 로 인코딩 된 ID 와 비밀번호를 전달하여 SSRF 를 진행한 뒤, 그림 13 의 {CMD}에 실행할 명령어가 객체로 전달되어 원격 코드 실행이 가능하다.

## ■ 공격 변화 양상

최근 MS Exchange Server 관련 취약점들이 다양하게 발생하고 있다. 아래의 표는 2021 년 발생한 ProxyLogon부터 최근 공격자들 사이에서 많이 사용되는 OWASSRF에 대한 내용을 정리한 표이다.

| 취약점 | 특징 |
|---|---|
| **ProxyLogon** | • CVE-2021-26855(SSRF 취약점) + CVE-2021-27065(Arbitrary File Write 취약점)<br>• 2021 년 1 월 발견<br>• 인증되지 않은 사용자가 외부 서버를 이용하여 내부 서버로 실행가능 파일 저장 가능<br>• 대응방안 - 인증 로직 추가 및 파일 확장자 검사 로직 추가, KB5000871 패치 적용 |
| **ProxyShell** | • CVE-2021-34473(MS Exchange Server RCE 취약점) + CVE-2021-34523(MS Exchange Server 권한 상승 취약점) + CVE-2021-31207(MS Exchange Server 보안 기능 우회 취약점)<br>• 2021 년 4 월 발견<br>• autodiscover 를 활용한 백엔드에 접근 후 임의 파일 쓰기를 통한 RCE 실행<br>• 대응방안 - 파일 확장자 제한, 인증 로직 추가, KB5004778 패치 적용 |
| **ProxyNotShell** | • CVE-2022-41040 (Auto Discover SSRF 취약점) +CVE-2022-41082 (RCE 취약점)<br>• 2022 년 8 월 발견<br>• autodiscover 를 활용한 백엔드 접근 후 RCE 실행<br>• 대응방안 - URL 차단 규칙 추가, 비 관리자를 위한 원격 PowerShell 비활성화, KB5019758 패치 적용 |
| **OWASSRF** | • CVE-2022-41080(OWA SSRF 취약점) + CVE-2022-41082(RCE 취약점)<br>• 2022 년 11 월 발견<br>• OWA 엔드포인트를 통한 백엔드 접근 후 RCE 실행<br>• 대응방안 - OWA 비활성화, 원격 PowerShell 제한, KB5019758 패치 적용, |

취약한 MS Exchange Server 를 노린 공격은 2021 년 발견된 ProxyLogon 이후 더욱 활발해졌다. ProxyLogon 은 인증되지 않은 사용자가 외부 서버를 통해 내부 서버에 Serverside 언어(.aspx)를 저장한 후 실행 가능한 취약점이다.

이후 MS Exchange Server 관련 취약점 연구가 증가했으며, 같은 해 보안 기능 우회/권한 상승/원격 코드 실행 취약점을 연계한 ProxyShell 이 등장했다. 2021 년 5 월~7 월에 각 취약점에 대한 패치가 진행되었지만, 2022 년에 해당 패치를 우회한 ProxyNotShell 이 등장했다.

ProxyNotShell 역시 MS 에서 발표한 긴급 보안 대책을 우회한 OWASSRF 로 발전해 현재 Play 랜섬웨어 집단에서 공격 방법으로 사용하고 있다. OWASSRF 는 OWA(Outlook Web Access) 엔드포인트를 통해 이뤄지며 /owa/<email_address>/ProwerShell 과 같이 OWA URL 사서함을 활용한 뒤, ProxyNotShell 의 CVE-2022-41082(RCE)를 실행한다.

2022 년 발견되어 현재까지 실제 공격에 이용되고 있는 ProxyNotShell 과 OWASSRF 의 차이점은 SSRF 공격에 사용되는 프론트엔드의 엔드포인트 공략 지점에 있다. ProxyNotShell 은 autodiscover 를 사용하고, OWASSRF 는 OWA 엔드포인트를 활용한다. 엔드포인트를 통한 백엔드 접근 이후 원격 코드 실행 취약점인 CVE-2022-4182 를 활용하는 점은 동일하다. 다음은 ProxyNotShell 과 OWASSRF 의 차이점을 나타낸 그림이다.

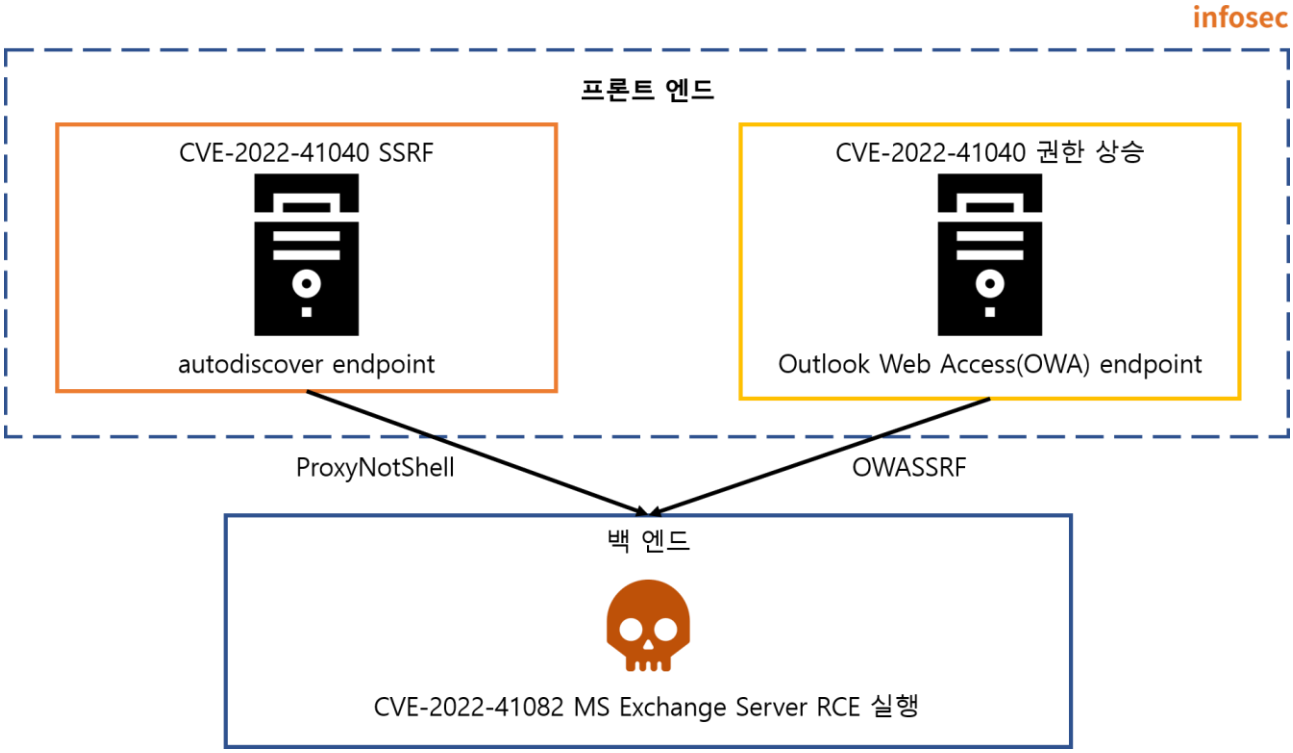


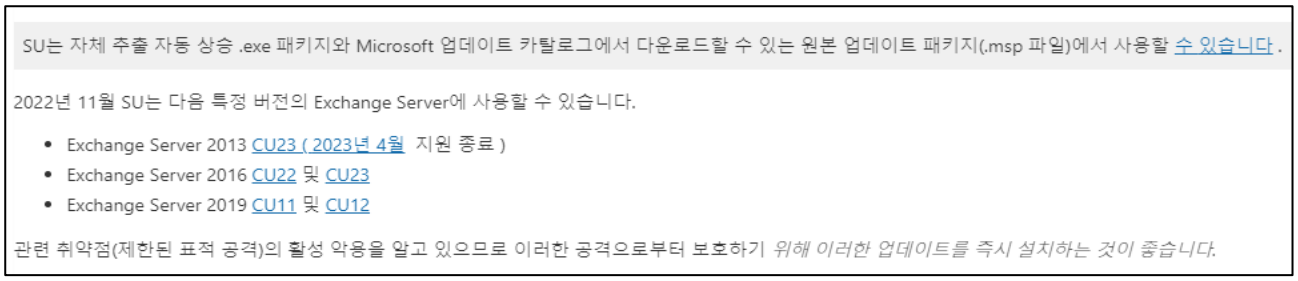

그림 14 ProxyNotSHell VS OWASSRF

## ■ 대응 방안

취약점 최초 발견 당시 Microsoft 에서는 URL 차단 규칙을 수동으로 설정하는 것을 권고했다. 하지만 이 방법은 우회 가능성으로 인해 효과적이지 않으며 새로 발생한 취약점인 OWASSRF 를 방지할 수 없는 문제가 있다.

따라서 취약점에 대비하기 위해 2022-11-8 이후 업데이트된 패치(KB5019758)를 적용해야 한다. 업데이트된 패치에 대한 내용은 아래의 링크에서 확인할 수 있다.

https://techcommunity.microsoft.com/t5/exchange-team-blog/released-november-2022-exchange-server-security-updates/ba-p/3669045


SU는 자체 추출 자동 상승 .exe 패키지와 Microsoft 업데이트 카탈로그에서 다운로드할 수 있는 원본 업데이트 패키지(.msp 파일)에서 사용할 수 있습니다 .

2022년 11월 SU는 다음 특정 버전의 Exchange Server에 사용할 수 있습니다.

- Exchange Server 2013 CU23 ( 2023년 4월 지원 종료 )
- Exchange Server 2016 CU22 및 CU23
- Exchange Server 2019 CU11 및 CU12

관련 취약점(제한된 표적 공격)의 활성 악용을 알고 있으므로 이러한 공격으로부터 보호하기 위해 이러한 업데이트를 즉시 설치하는 것이 좋습니다.

그림 15 MS Exchange Server 패치 내역

만약 불가피하게 업데이트를 할 수 없는 경우 패치 적용 전까지 관리자가 아닌 사용자가 PowerShell 를 원격으로 실행하지 못하게 원격 PowerShell 를 비활성화해야 한다. 또한, ProxyNotShell 의 우회 취약점인 OWASSRF 에 대비하기 위하여 OWA 를 비활성화해야 한다. 웹 방화벽이나 EDR(Endpoint Detection & Response) 도구를 활용하여 악의적인 요청(PowerShell 실행, cmd 실행 등)을 탐지해야 하며, X-Forwarded-For 헤더가 프록시 서비스에 대한 요청을 위해 실제 외부 IP 주소를 기록하도록 구성되었는지 확인해야 한다. 그럼에도 임시 대책으로 우회할 가능성이 있으므로, 업데이트된 패치를 상시 확인 후 적용해야 안전하다.

## ■ 참고 사이트


• URL: https://www.crowdstrike.com/blog/owassrf-exploit-analysis-and-recommendations/
• URL: https://www.zerodayinitiative.com/blog/2022/11/14/control-your-types-or-get-pwned-remote-code-execution-in-exchange-powershell-backend
• URL: https://www.rezilion.com/blog/proxyshell-or-proxynotshell-lets-set-the-record-straight/
• URL: https://twitter.com/wdormann/status/1593630153036500993/photo/1
• URL: https://www.shodan.io/search/report?query=http.component%3A%22Outlook+Web+App+%22

# EQST INSIGHT

2023.01

SK쉴더스

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

발행인 : SK쉴더스 EQST사업그룹
제　작 : SK쉴더스 커뮤니케이션그룹