리버싱 기초 및 MITRE ATT&CK 개요

강민주

리버싱(Reverse Engineering)이란?

리버싱(Reverse Engineering)이란 소프트웨어의 구조, 기능, 동작 원리를 역으로 분석하여 이해하고, 부족한 부분을 보완하거나 새로운 기능을 추가하는 과정을 의미한다. 리버싱은 다양한 목적으로 사용된다.

→ 개발이 중단된 프로그램에 대한 패치
→ 프로그램의 보안성 평가
→ 악성코드 분석
→ 키젠 프로그램 제작
→ 시리얼 넘버 생성
→ 크랙 버전 제작
→ 게임 핵 제작

리버싱 방법

정적 분석 (Static Analysis)

정적 분석이란 프로그램을 실행하지 않고, 외형적인 관찰만으로 파일의 구조를 분석하는 방법이다. 파일의 종류, 크기, 헤더 정보, 압축 여부, 디지털 인증서 등의 정보를 확인하며, 디스어셈블러를 이용해 내부 코드와 구조를 파악한다.

장점: 전체 구조 파악이 용이. 분석 환경의 제약에서 비교적 자유로움. 악성 프로그램의 위협으로부터 안전.

단점: 난독화된 코드 분석이 어렵다. 동적 요소를 고려하기 어렵다.

동적 분석 (Dynamic Analysis)

동적 분석이란 파일을 실제로 실행하여 프로그램의 동작을 분석하는 방법이다. 디버깅을 통해 코드의 흐름과 메모리 상태를 자세히 살펴보며, 레지스트리와 네트워크 등을 관찰한다.

장점: 코드의 자세한 분석 없이도 개략적인 동작 파악이 가능하다.

단점: 분석 환경 구축이 어려울 수 있다. 안티 디버깅 기술이 적용된 프로그램 분석이 어렵다.

PE 파일 구조

컴퓨터의 기본 요소는 CPU, 메모리, 하드디스크이다. 실행할 수 있는 파일은 기본적으로 HDD 에 저장되며, 윈도우 운영체제에서는 이 실행 파일을 PE(Portable Executable) 파일이라고 한다.

PE 파일은 헤더(PE Header+Section Header)과 바디(Section Data)으로 구성되며, PE Header 는 다시 다양한 세부 정보로 나뉜다.

PE Header: 프로그램 실행에 필요한 기본 정보와 배치 정보를 포함한다.
Section Header: PE 파일의 섹션에 대한 정보를 제공한다.
Section Data: 프로그램의 코드와 데이터가 포함된 섹션이다.

PE 파일의 주요 구성 요소는 다음과 같다.
→ IMAGE_DOS_HEADER: DOS 운영체제가 윈도우용 PE 파일을 실행했을 때 오류 메시지를 보여주며, 실제 윈도우용 PE 헤더의 위치를 가리킨다.
→ MS-DOS Stub Program: DOS 운영체제가 윈도우용 PE 파일을 실행했을 때 보여 줄 오류 메시지를 저장하고 있다.
→ IMAGE_OPTIONAL_HEADER: PE 구조에서 핵심적인 역할을 하며, 메모리에 PE 파일이 저장되는 시작 주소인 Image Base 등을 포함한다.

메모리 구성 및 PE 파일 실행 과정

메모리 구성 요소는 다음과 같다.
→ 코드 영역: 프로그램 코드가 들어가는 영역이다.
→ 데이터 영역: 정적 변수와 전역 변수가 저장되는 영역이다.
→ 스택 영역: 함수 호출 시 사용되는 매개 변수와 지역 변수가 저장되는 영역이다.
→ 힙 영역: 동적 메모리 할당에 사용되는 영역이다.

PE 파일이 메모리에 로딩된 후 프로그램은 다음과 같은 순서로 실행된다.

1. 로더가 PE 헤더 정보를 분석하여 PE 바디를 메모리에 배치한다.
2. 코드 영역과 데이터 영역에 필요한 정보를 메모리에 로딩한다.
3. 프로그램 실행 중 스택과 힙 영역에 데이터가 동적으로 쌓인다.
4. PE 파일의 Entry Point에서 프로그램 실행이 시작된다.

명령을 실행할 때 필요한 데이터는 레지스터에 존재해야 한다.

레지스터

레지스터는 CPU 가 연산 및 중간 데이터를 저장하기 위해 사용하는 고속 기억 장치이다.
x86 CPU 의(32 비트) 주요 레지스터

범용 레지스터
→ EAX: 곱셈/나눗셈 수행 및 함수 반환 값을 저장한다.
→ EBX: 인덱스 레지스터로 사용된다.
→ ECX: 반복 명령어의 횟수를 저장한다.
→ EDX: EAX 와 같이 사용되며 부호 확장 명령에 활용된다.

인덱스 레지스터

→ ESI: 데이터 복사나 조작 시 소스 데이터 주소를 저장한다.

→ EDI: 데이터 복사의 목적지 주소를 저장한다.

포인터 레지스터

→ EBP: 스택 프레임의 시작 주소를 저장한다.

→ ESP: 스택 프레임의 끝 주소를 저장한다.

상태 레지스터

→ EIP: 다음에 실행할 명령어가 저장된 메모리 주소이다.

+세그먼트 레지스터

X64 CPU 의(64 비트) 주요 레지스터

범용 레지스터

→ RAX(accumulator): 산술/논리 연산을 수행하며 함수의 return 값이 저장된다. 시스템콜 함수를 사용하려면 RAX 에 함수의 syscall 번호를 넣어준다.

→ RBX (Base) : 메모리 주소를 저장하기 위한 용도로 사용된다.

→ RCX (Count) : 반복문에서 카운터로 사용되는 레지스터. syscall 을 호출했던 사용자프로그램의 return 주소를 가진다.

→ RDX (Data) : 다른 레지스터를 서포트하는 여분의 레지스터. 큰 수의 곱셈이나 나눗셈 연산에서 EAX 와 함께 사용된다.

인덱스 레지스터

→ RSI (Source Index) : 데이터를 복사할 때 src 데이터, 즉 복사할 데이터의 주소가 저장된다.

→ RDI (Destination Index) : 데이터를 복사할 때 복사된 dest 데이터의 주소가 저장된다.

포인터 레지스터

→ RSP (Stack Point) : 스택프레임에서 스택의 끝 지점 주소(현재 스택 주소)가 저장된다. 즉, 데이터가 계속 쌓일 때 스택의 가장 높은 곳을 가리킨다.

→ RBP (Base Point) : 스택스레임의 시작 지점 주소(스택 복귀 주소)가 저장된다.

플래그 레지스터

→ RFLAGS register : 시스템 제어 용도 혹은 어셈블리에서는 비교/조건문 처리 용도로 사용되는 상태 레지스터이다. 연산 결과에 따라 64 비트 레지스터의 각 비트에 0(clear, reset) 혹은 1(set)로 표시해 처리 결과를 저장한다.
→ CF (Carry Flag) : 부호 없는 수 끼리 연산 결과가 자리올림/자리내림이 발생할 때 1, unsigned int 값을 벗어날 때 1
→ OF (Overflow Flag) : 부호 있는 수 끼리연산 결과가 용량을 초과하였을 경우 1
→ SF (Sign Flag) : 연산 결과 최상위 비트가 1 인 경우 1
→ ZF (Zero Flag) : 연산 결과가 0 이면 1
→ AF (Auximiliary-carry Flag) : 16 비트 연산시 자리올림/자리내림이 발생할 때 1
→ PF (Parity Flag) : 연산 결과가 짝수면 1, 홀수면 0
→ DF(Direction Flag)
→ IF (Interrupt Fla)
→ TF (Trap Flag)

이 외에도, 64 비트에서는 R8 ~ R15 까지 8 개의 레지스터가 추가로 사용되며, 하나의 레지스터는 크기에 따라 쪼개 사용할 수 있다.

스택과 스택 프레임

스택은 데이터를 후입선출(LIFO) 방식으로 저장하는 메모리의 한 부분이다. 스택 프레임이란 함수가 호출될 때, 그 함수만의 스택 영역을 구분하기 위해서 생기는 영역이다. 이 공간에는 함수와 관계되는 지역 변수, 매개변수가 저장되며, 함수 호출 시 할당되고 함수가 종료되면 소멸한다. 스택 프레임 덕분에 함수의 호출이 모두 끝난 뒤에, 해당 함수가 호출되기 이전 상태로 돌아갈 수 있다.

→ 함수의 시작을 고정하기 위해 ESP 값을 EBP 에 저장하고 유지한다. ESP 값이 아무리 변해도 EBP 를 기준으로 하기 때문에 함수의 변수, 파라미터, 리턴 주소에 쉽게 접근할 수 있다.

코드 7.1 스택 프레임의 구조

```
PUSH EBP                ; 함수 시작(EBP를 사용하기 전에 기존의 값을 스택에 저장)
MOV EBP, ESP            ; 현재의 ESP(스택포인터)를 EBP에 저장

...                     ; 함수 본체
                        ; 여기서 ESP가 변경되더라도 EBP가 변경되지 않으므로
                        ; 안전하게 로컬 변수와 파라미터를 엑세스할 수 있음

MOV ESP, EBP            ; ESP를 정리(함수 시작했을 때의 값으로 복원시킴)
POP EBP                 ; 리턴되기 전에 저장해 놓았던 원래 EBP 값으로 복원
RETN                    ; 함수 종료
```

주소 지정 방식

1. 즉시 지정방식(immediate addressing)
   : mov $0x1, %eax (eax에 1 값 할당)과 같이 메모리의 주소 내용을 꺼내지 않고 직접 값을 대응시키는 방식
2. 레지스터 지정방식(register addressing)
   : mov $esp, %ebp (ebp에 esp의 값을 할당)과 같이 레지스터에서 직접 레지스터로 값을 대응시키는 방식
3. 직접 주소 지정방식(directly addressing)
   : mov %eax, $0x80482f2 (0x80482f2에 있는 값을 eax에 할당)과 같이 메모리의 주소를 직접 지정해서 바로 찾아오는 방식. 가장 일반적이다.
4. 레지스터 간접 주소 지정 방식
   : mov (%ebx), %eax (ebx의 값을 주소로 해서(간접적으로) eax 레지스터에 할당). ( )가 들어간다면 간접 지정으로 볼 수 있음.
5. 베이스 상대 주소 지정 방식
   : mov 0x4(%esi), %eax (esi레지스터에서 4바이트를 더한 주소의 값을 eax에 할당)

어셈블리 명령어

조작 명령어

→ call: 프로시저 호출

→ ret: call 로 호출된 함수를 종료하고 그 다음 명령줄로 이동

→ nop: 아무것도 하지 않음

→ jmp: 무조건 분기

조건 점프 명령어: cmp 연산 결과에 따라 jmp

→ je: cmp A B 에서 A=B 일때 특정 라벨로 jmp
→ jne: cmp A B 에서 A != B 일때 특정 라벨로 jmp
→ ja: cmp A B 에서 A > B 일때 특정 라벨로 jmp
→ jb: cmp A B 에서 A < B 일때 특정 라벨로 jmp
→ jae: A >= B
→ jbe: A <= B

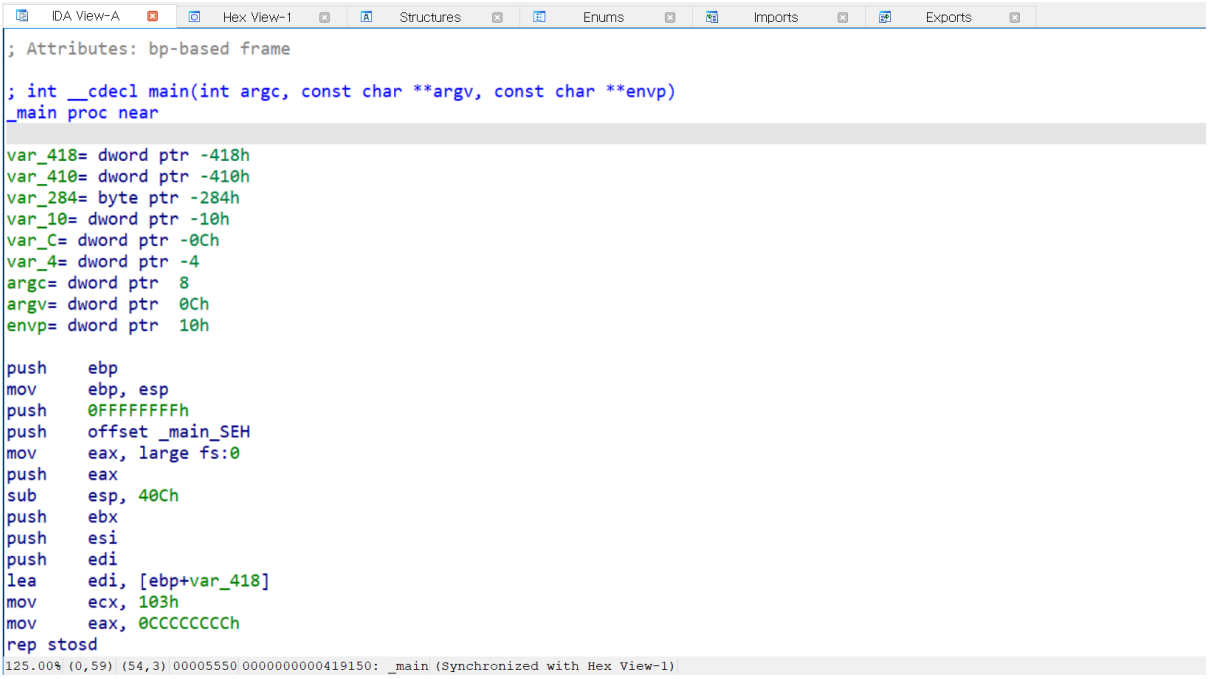
데이터 전송 명령어
→ push: 스택에 값을 넣음
→ pop: 스택에서 값을 가져옴
→ mov: 인자 2 값을 인자 1 에 대입(전달)
→ lea: 인자 2 주소를 인자 1 에 대입(전달)

산술 명령어
→ inc: 인자의 값을 1 증가
→ dec: 인자의 값을 1 감소
→ add: 인자 2 값을 인자 1 에 더함
→ sub: 인자 2 값을 인자 1 에서 뺌
→ mul: ax 와 오퍼렌드를 곱셈하여 결과를 ax 또는 DX:AX 에 저장
→ imul: 부호화된 곱셈
→ div: AX 또는 DX:AX 내용을 오퍼랜드로 나눔. 몫은 AL, AX 나머지는 AH, DX 로 저장
→ neg: 오퍼랜드의 2 의 보수, 즉 부호 반전
→ cmp: 인자 1,2 의 값을 비교. 주로 위의 조건점프 명령어와 세트로 사용한다.
→ test: 인자 1 과 인자 2 를 AND 연산한다. 이 연산의 결과는 ZF(zero flag)에만 영향을 미치고 Operand 자체에는 영향을 미치지 않는다. 보통 rax 의 값이 0 인지 확인할 때 rax 0, 0 이런 식으로 사용된다. 만약 TEST 의 연산결과가 0 이라면 ZF 는 1 로, 연산결과가 1 이라면 ZF 는 0 으로 세트된다.

인터럽트 명령어
→ int: 운영체제에 할당된 인터럽트 영역을 system call.

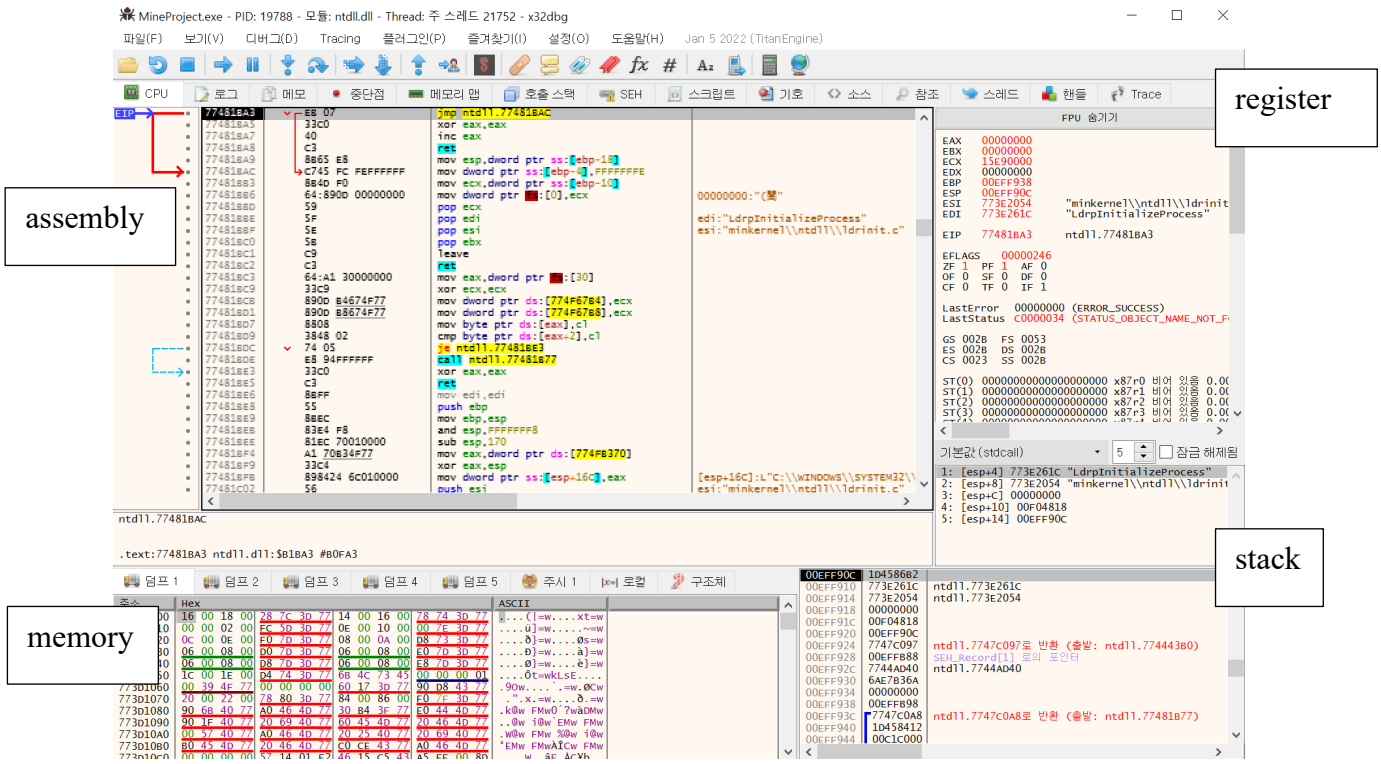
리버싱 툴 소개

정적 분석 툴: IDA



프로그램의 어셈블리 코드를 볼 수 있다. IDA 는 문자열이나 함수를 어디에서 사용하는지 보여주는 '상호 참조 기능', 함수의 실행 흐름을 보기 쉽게 해주는 '제어 흐름 그래프' 등의 기능이 존재한다.

동적 분석 툴: OllyDbg, x64dbg



디버거 중 하나인 x64dbg 를 사용하면 실행 중인 어셈블리 코드, CPU 의 레지스터 상태, 메모리 상태, 스택의 값을 확인하면서 분석을 진행할 수 있다.

리버싱 예제

https://codeengn.com/challenges/

MITRE ATT&CK 란?

사이버 범죄자의 알려진 악의적 행동을 기반으로 사이버 보안 위협을 모델링, 탐지, 예방 및 대응하기 위해 지속적으로 업데이트 되는 지식 기반이다.
기존의 cyber kill chain 과 같은 모델에서 공격의 흐름과 프로세스 관점에서 개념의 단계로만 기술된 부분을 TTP(tactics, techniques, procedures)매핑을 통해 실용성을 높이고 실제 공격 사례를 제공해서 침해사고 대응에 적용하기 용이한 모델로 발전시킨 것이다.

ATT&CK Matrix

ATT&CK 메트릭스는 공격자가 사이버 공격을 수행하는 과정을 시각적으로 표현한 표 형식의 도구이다. 공격의 각 단계에서 사용될 수 있는 전술(tactics)과 기술(techniques)을 체계적으로 정리한 것이다.
→ Enterprise matrix: 엔터프라이즈 인프라에 대한 공격에 사용되는 모든 악의적 기법 + 윈도우, 맥 os, 리눅스 플랫폼에 대한 하위 매트릭스 + 준비 기법의 사전 매트릭스
→ Moblie matrix: 모바일 디바이스에 대한 직접 공격 + 네트워크 기반 모바일 공격에 사용되는 기법 + ios/android 모바일 플랫폼용 하위 매트릭스
→ ICS matrix: 산업 제어 시스템(공장, 유틸리티, 운송 시스템 등등)의 운영을 제어하거나 자동화하는 데 사용되는 기계, 디바이스, 센서 및 네트워크에 대한 공격에 사용되는 기법

MITRE ATT&CK/ 전술(tactics)

공격 전술 정보는 공격자의 공격 목표에 따른 행동을 나타내며. 사이버 공격의 단계에 밀접하게 대응된다. 공격 목적에 따라 지속성, 정보 탐색, 실행, 파일 추출 등 다양하게 분류된다. 총 40 개의 tactics 를 제공한다(Enterprise : 14, Mobile : 14, ICS : 12).

ATT&CK for Enterprise tactics

정찰 (Reconnaissance)	내부정찰단계로 다른 시스템으로 이동하기 위해 탐구하는 단계
자원 개발 (Resource Development)	다른 시스템으로 이동하기 위한 정보로 계정 등을 확보하는 단계
초기 접근 단계 (Initial Access)	네트워크 진입을 위해 사용자 환경에 대한 정보를 취득하는 것을 목적으로 함
실행 (Execution)	공격자가 로컬 또는 원격 시스템을 통해 악성코드를 실행하기 위한 행동
지속 (Persistence)	공격 기반을 유지하고 시스템에 지속적으로 접근하기 위한 행동
권한 상승 (Privilege Escalation)	공격자가 시스템이나 네트워크에서 높은 권한을 얻기 위한 행동
방어 회피 (Defense Evasion)	공격자가 침입한 시간 동안 탐지 당하는 것을 피하기 위한 행동
접속 자격 증명 (Credential Access)	시스템, 도메인 서비스, 자격증명 등을 접근하거나 제어하기 위한 행동
탐색 (Discovery)	시스템 및 내부 네트워크의 정보를 얻기 위한 행동
내부 확산 (Lateral Movement)	네트워크 상의 원격 시스템에 접근한 후 이를 제어하기 위한 행동
수집 (Collection)	공격 목적이나 관련 정보가 포함된 데이터를 수집하기 위한 행동
명령 및 제어 (Command And Control)	공격자가 침입한 대상 네트워크 내부 시스템과 통신하며 제어하기 위한 행동
유출 (Exfiltration)	공격자가 네트워크에서 데이터를 훔치기 위한 행동
임팩트 (Impact)	공격 목표의 가용성과 무결성을 손상시키기 위한 행동

ATT&CK for ICS tactics

수집 (Collection)	목표 대상의 데이터 및 시스템에 대한 정보를 수집하기 위한 전술
명령 및 제어 (Command and Control)	ICS 환경에 침입하여 장악한 시스템, 컨트롤러 및 플랫폼에 대해 통신과 제어하려는 전술
탐색 (Discovery)	ICS 의 원격 시스템에 접근한 후 이를 제어하기 위해 사용되는 전술
회피 (Evasion)	공격자가 침입한 시간동안 탐지 당하는 것을 피하기 위해 사용되는 전술
실행 (Execution)	악성코드나 프로그램을 임의로 실행
임팩트 (Impact)	ICS 에 대해 데이터 조작, 시스템 중단 및 파괴하려는 전술
제어권 손상 (Impair Process Control)	물리적 제어 절차를 비활성화하거나 손상하려는 전술
응답 거절 (Inhibit Response Function)	안전과 관련된 기능에 대해 응답하지 못하도록 하는 전술
초기 접근 (Initial Access)	ICS 환경에 접근하기 위한 전술
내부 확산 (Lateral Movement)	ICS 의 원격 시스템에 접근한 후 이를 제어하기 위해 사용하는 전술
지속 (Persistence)	침입한 ICS 네트워크 환경에 대해 저속적으로 접속 유지를 위한 전술
권한 상승 (Privilege Escalation)	ICS 환경에 침입하여 시스템이나 네트워크에서 높은 권한을 얻기 위한 전술

MITRE ATT&CK/ 기술(techniques)

공격 기술 정보는 공격자가 목표에 대한 전술을 달성하기 위해 사용하는 구체적인 방법이나 수단을 말한다. 기술은 더 세분화된 하위 기술로 나뉠 수 있다. 총 392 개 Techniques 제공한다. (Enterprise : 185(Sub-Tech : 367), Mobile : 118, ICS : 89)

MITRE ATT&CK/ 공격 완화 정보(Mitigations)

방어자가 공격을 예방하고 탐지하기 위해 취할 수 있는 행동을 의미한다. 과거 유사 사례에서의 대응책 정보를 활용해서 새로 탐지된 공격에 대한 해결방안을 제시할 수 있다. 총 106 개의 Mitigations 제공한다. (Enterprise : 42 개, Mobile : 13, ICS : 51 개)

MITRE ATT&CK/ 공격 단체, 조직 정보

공개적으로 명칭이 부여된 해킹 단체에 대한 정보와 공격 기법을 분석해서 정리한다. 주로 사용된 공격 방법과 활동 분석, 공식 문서 등을 바탕으로 해킹조직을 특정하여 정의한다. 공격에 사용된 Technique 과 Software 목록을 포함하고 있으며 이와 매핑하여 해킹그룹이 자주 사용하는 공격 형태를 제공하므로 새로운 공격 발생 시, 기존 Matrix 를 활용, 비교할 수 있다. 총 131 개의 Groups 정보 제공한다. (Enterprise/Mobile : 122, ICS : 9)

MITRE ATT&CK/ 공격 도구 정보

공격자가 목표 대상을 공격할 때 사용된 공격코드 또는 운영체제(OS)에 포함된 기본 도구나 공개적으로 사용 가능한 도구(Open-Source S/W) 등을 목록화하여 정리한다. 총 604 개의 공격 도구 정보를 제공한다. (Enterprise/Mobile : 585 개, ICS 19 개)