

# **악성코드를 통한 보안 훈련 플랫폼 개발**

## **(Development of Security training platform Using Malware)**

**APTY**

전동혁 최우영 강세현 서혁준

## 목차

1. 서론	1
1.1 프로젝트 배경 및 목적	1
1.2 핵심내용	2
1.3 기대 효과	3
1.4 프로젝트 구성	3
2. 관련연구	6
2.1 악성코드 재현 및 모델링	6
2.1.1 njRAT	6
2.1.2 Gh0st RAT	7
2.1.3 Quasar	7
2.1.4 Cobalt Strike	8
3. 관련 기술	9
3.1 Direct System Call	9
3.2 DLL(Dynamic Link Library) Manual Mapping	10
3.3 Process Hollowing	11
3.4 Executable Binder	12
4. 프로젝트 구현	13
4.1 테스트 환경 구성	13
4.1.1 환경구축	14
4.2 관리용 플랫폼 개발	21
4.2.1 대시보드 메인 페이지	21
4.2.2 피싱 메일 전송 페이지	22
4.2.3 악성 파일 업로드 페이지	26
4.2.4 클라이언트 관리 페이지	27
4.2.5 탈취 파일 관리 페이지	28
4.2.6 브라우저 자격 증명 탈취 페이지	29
4.2.7 키로깅 페이지	29
4.2.8 화면 캡처 페이지	30
4.2.9 상세 로그 페이지	30
4.2.10 패치노트 페이지	32

4.3 악성코드 세부 동작-----	34
4.3.1 Keylogger-----	37
4.3.2 파일 탈취 -----	38
4.3.3 브라우저 자격증명 탈취 -----	39
4.3.4 DDoS -----	40
4.3.5 화면 캡처 -----	41
4.3.6 연결된 클라이언트 정보 출력-----	42
4.3.7 연결된 클라이언트 세션 삭제-----	43
4.3.8 서버 종료 -----	43
4.3.9 도움말 기능 -----	45
4.3.10 스레드 구조 -----	46
4.3.11 자동 업데이트 및 버전 관리-----	48
4.3.12 Defender 설정 조작-----	49
4.3.13 지속 메커니즘 및 사용자 계정 컨트롤 -----	50
4.3.14 키보드 보안 프로그램 설치 유무 -----	51
4.3.15 암호화-----	52
4.3.16 클라이언트 ON/OFF-----	53
4.3.17 클라이언트 중복 접속 처리-----	54
5. 프로젝트 결론-----	55
참고 문헌-----	56
Abstract -----	57
팀원소개 -----	58

## 그림 목차

[그림 1-1] ANY.RUN 2024년 3분기 악성코드 동향 .....	1
[그림 1-2] 텔레그램에서 유통되고 있는 악성코드 .....	2
[그림 1-3] 전체 프로젝트 구조 .....	4
[그림 2-1] njRAT 로고 .....	6
[그림 2-2] Gh0st RAT 로고 .....	7
[그림 2-3] Quasar 로고 .....	7
[그림 2-4] Cobalt Strike 로고 .....	8
[그림 3-1] Direct System Call 동작 원리 .....	9
[그림 3-2] DLL Manual Mapping 동작 원리 .....	10
[그림 3-3] Process Hollowing 동작 원리 .....	11
[그림 3-4] Executable Binder 동작 원리 .....	12
[그림 4-1] 도커 파일의 구조 .....	15
[그림 4-2] 도커 실행 및 컨테이너 적용 .....	16
[그림 4-3] 컨테이너로 생성된 웹 서버 .....	20
[그림 4-4] 컨테이너로 생성된 피싱 사이트 .....	20
[그림 4-5] 대시보드 메인 페이지 .....	21
[그림 4-6] 피싱 메일 전송 페이지 .....	22
[그림 4-7] 악성 파일 업로드 페이지 .....	26
[그림 4-8] 클라이언트 관리 페이지 .....	27
[그림 4-9] 탈취 파일 관리 페이지 .....	28
[그림 4-10] 자격 증명 탈취 파일 관리 페이지 .....	29
[그림 4-11] 키로그 파일 관리 페이지 .....	29
[그림 4-12] 화면 캡처 파일 관리 페이지 .....	30
[그림 4-13] 상세 로그 출력 페이지 .....	30
[그림 4-14] 패치노트 페이지 .....	32
[그림 4-15] 피싱 메일을 통해 피싱 사이트 접속 .....	34
[그림 4-16] 악성프로그램 설치 유무 확인 후 리다이렉션 .....	34
[그림 4-17] 악성프로그램 설치하지 않을 경우 .....	35
[그림 4-18] 위조된 보안프로그램 설치 화면 .....	35
[그림 4-19] 악성코드 지속 메커니즘 등록 .....	35

[그림 4-20] 스레드 구조 구성도 .....	46
[그림 4-21] 자동 업데이트 및 버전 관리 구성도.....	48
[그림 4-22] Defender 우회 구성도 .....	49
[그림 4-23] 지속 메커니즘 및 UAC 구성도 .....	50
[그림 4-24] 피싱 사이트 리다이렉션 구성도 .....	51
[그림 4-25] 암호화 구성도 .....	52
[그림 4-26] 클라이언트 ON/OFF 구성도.....	53
[그림 4-27] 클라이언트 중복 접속 처리 구성도 .....	54

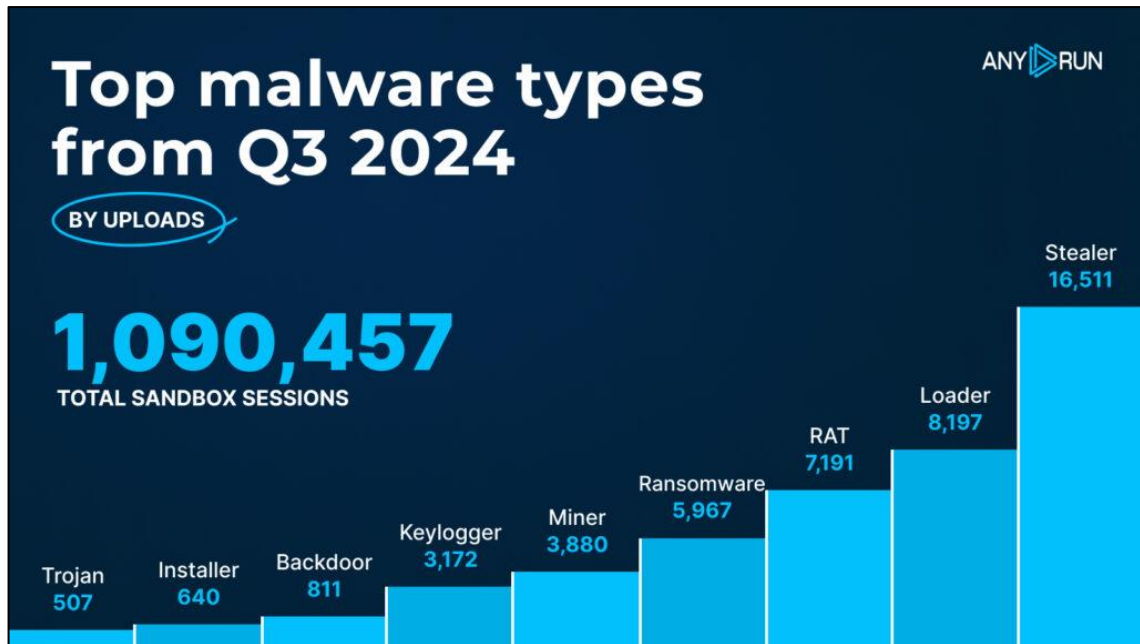
## 표 목차

[표 4-1] C2서버 구동 환경 .....	13
[표 4-2] 클라이언트 구동 환경.....	13
[표 4-3] 모의 악성코드 개발 환경.....	13
[표 4-4] 대시보드 환경 .....	14
[표 4-5] wget을 이용하여 도커 파일 설치 .....	15
[표 4-6] start.sh 스크립트 실행(1).....	15
[표 4-7] start.sh 스크립트 실행(2).....	16
[표 4-8] 악성코드 지속 메커니즘 등록 .....	36
[표 4-9] 키로깅 명령어 .....	37
[표 4-10] 키로깅 시작 및 클라이언트 입력.....	37
[표 4-11] 키로깅 종료 및 C&C 서버에 로그 파일 전송 .....	37
[표 4-12] 파일 탈취 명령어.....	38
[표 4-13] 파일 탈취.....	38
[표 4-14] 브라우저 자격증명 탈취 명령어.....	39
[표 4-15] 탈취된 브라우저 자격증명 파일 .....	39
[표 4-16] 브라우저 자격증명 파일이 존재하지 않을 경우.....	40
[표 4-17] DDos 명령어.....	40
[표 4-18] 공격을 통한 요청 로그 .....	40
[표 4-19] 화면 캡처 명령어 .....	41
[표 4-20] 전송된 클라이언트 화면 캡처 파일.....	41
[표 4-21] 연결된 클라이언트 정보 출력 명령어 .....	42
[표 4-22] 연결된 클라이언트 정보.....	43
[표 4-23] 연결된 클라이언트 세션 삭제 명령어 .....	43
[표 4-24] 연결된 클라이언트 세션 삭제.....	43
[표 4-25] 서버 종료 명령어.....	43
[표 4-26] exit (y 입력시).....	44
[표 4-27] exit (n 입력시).....	44
[표 4-28] 도움말 명령어 .....	45
[표 4-29] 도움말 .....	45
[표 4-30] 스레드 구성 요소 .....	47

## 1. 서론

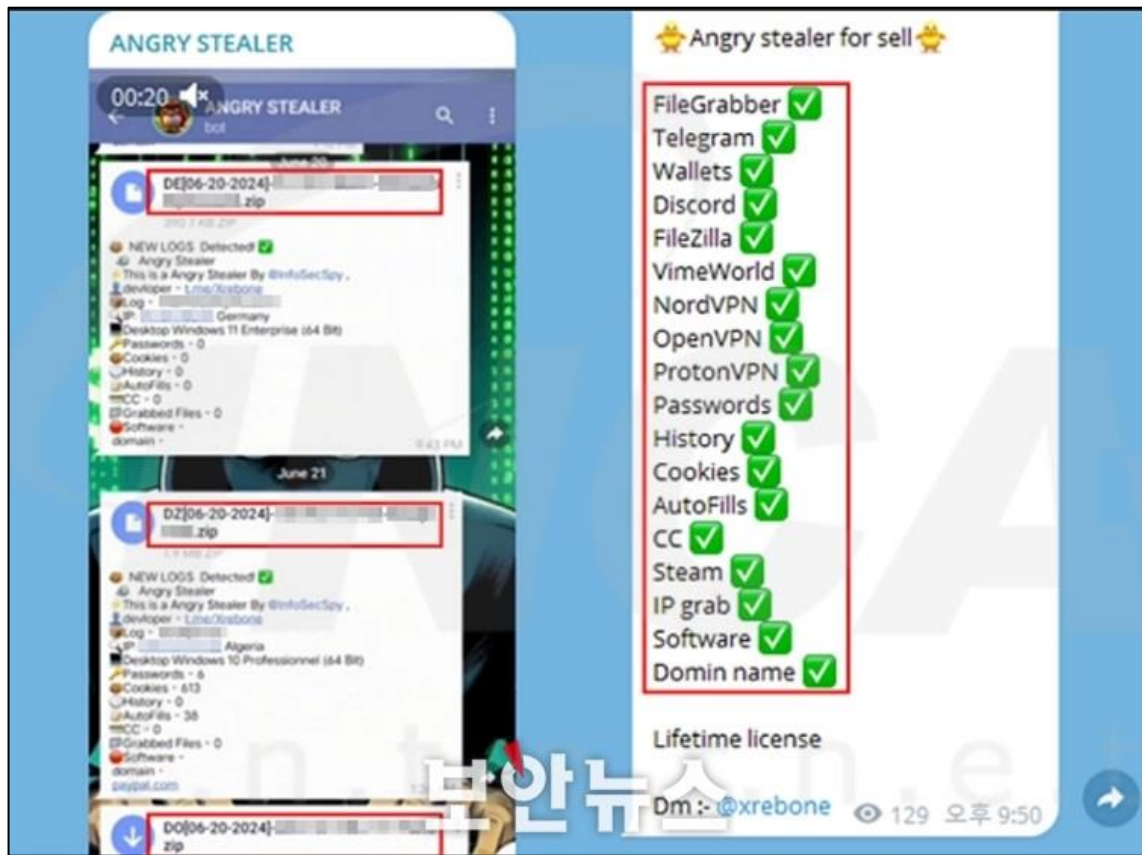
### 1.1 프로젝트 배경 및 목적

악성코드는 현대 사회에서 기업 및 개인의 정보 자산을 위협하는 주요 요인으로 자리 잡고 있으며, 사이버 공격의 빈도와 형태는 점점 더 정교화되고 다양해지고 있다. 이러한 위협 요소 중에서도 원격 접근 트로이 목마(RAT, Remote Access Trojan)는 특히 심각한 위협으로 평가받는다. RAT는 사용자의 시스템에 은밀히 침투하여 민감한 정보 탈취, 스파이 활동, 시스템 손상 등의 악의적 활동을 수행할 수 있다. 아랍에미리트 두바이에 위치한 컴퓨터 보안 서비스 회사인 ANY.RUN에서 발표한 자료에 따르면, [그림 1-1]과 같이 RAT는 세번째로 높은 위협 요소로 분류되고 있다.



[그림 1-1] ANY.RUN 2024년 3분기 악성코드 동향

최근에는 [그림 1-2]에서 확인할 수 있듯이 메신저를 매개로 한 악성코드와 원격 접근 트로이목마(RAT)의 유통이 활발히 이루어지고 있다. 이러한 환경에서는 컴퓨터 지식이 부족한 일반 사용자도 손쉽게 악성코드를 구매하고 이를 활용한 공격을 수행할 수 있는 상황이 조성되고 있다. 이로 인해 악성코드 공격의 진입 장벽이 크게 낮아졌으며, 사회 전반에 걸친 보안 위협의 심화로 이어지고 있다. 따라서 이러한 새로운 악성코드 유통 방식과 급증하는 보안 위협에 효과적으로 대응하기 위해서는, 악성코드에 대한 깊은 이해를 기반으로 한 실습 및 연구 환경의 구축이 필수적이다.



[그림 1-2] 텔레그램에서 유통되고 있는 악성코드

본 프로젝트는 악성코드의 구조와 동작 원리를 체계적으로 분석하고 이를 기반으로 보안 인식 향상과 실질적인 대응 역량을 강화할 수 있는 모의 악성코드를 개발하는 데 초점을 맞추고 있다. 이를 위해 개발된 모의 악성코드 테스트 플랫폼은 다양한 보안 위협 시나리오를 재현함으로써 사용자들이 안전한 환경에서 악성코드의 감염 경로와 작동 방식을 경험하고 분석할 수 있도록 설계되었다. 해당 플랫폼은 악성코드의 주요 동작을 시뮬레이션 하여 보안 위협의 심각성을 체감하게 함으로써 보안 관행 준수의 필요성을 강조 시키고, 사용자의 보안 인식을 한 층 강화하는 데 기여한다.

## 1.2 핵심내용

본 프로젝트는 원격 접근 트로이 목마(RAT, Remote Access Trojan)를 활용하여 공격자가 감염된 컴퓨터를 원격으로 제어할 수 있는 환경을 설계하였다. 프로젝트의 구현 과정에서는 Direct System Call, DLL Manual Mapping, Process Hollowing 과 같은 고급 기술을 적용하여 실제 악성코드의 동작을 정교하게 재현하였다. 또한, 감염된 컴퓨터의 다양한 정보를 C&C 서버로



수집하고, 이를 대시보드를 통해 실시간으로 가시화함으로써 감염된 시스템에서 발생하는 모든 행위를 명확히 파악할 수 있도록 하였다.

개발한 테스트 플랫폼에서 주요 기능은 다음과 같다.

- 키로깅: 사용자의 키보드 입력을 기록하여 민감한 데이터를 탈취하는데 활용된다.
- 파일 탈취: 중요 문서나 데이터 파일을 탈취하여 기밀성을 침해한다.
- 브라우저 자격 증명 탈취: 웹 브라우저에 저장된 로그인 정보나 쿠키를 수집한다.
- 화면 캡처: 사용자의 디스플레이 내용을 캡처하여 공격자 서버로 전송한다.
- DDoS: 특정 네트워크나 서버에 과도한 트래픽을 발생시킨다.

### 1.3 기대효과

본 프로젝트는 보안 훈련 도구를 개발하여 보안 인력의 실무 역량을 강화하고, 악성코드에 대한 대응 능력을 체계적으로 향상시키는 데 기여할 것으로 기대된다. 악성코드의 구조와 동작 원리를 심층적으로 분석하고 이를 기반으로 한 다양한 침투 시나리오와 모니터링 분석 환경을 제공함으로써, 실제 보안 위협 상황에 실질적이고 효과적인 훈련이 가능해질 것이다.

특히, 프로젝트에서 분석된 악성코드 데이터를 활용하여 향후 발생할 수 있는 새로운 공격에 대한 예방책을 마련함으로써, 보안 인프라의 안정성을 높이는 데 기여할 수 있다. 이는 악성코드 탐지 및 감염 예방 역량을 지속적으로 향상시키고, 새로운 위협에 신속하게 대응할 수 있는 기반을 제공할 것이다.

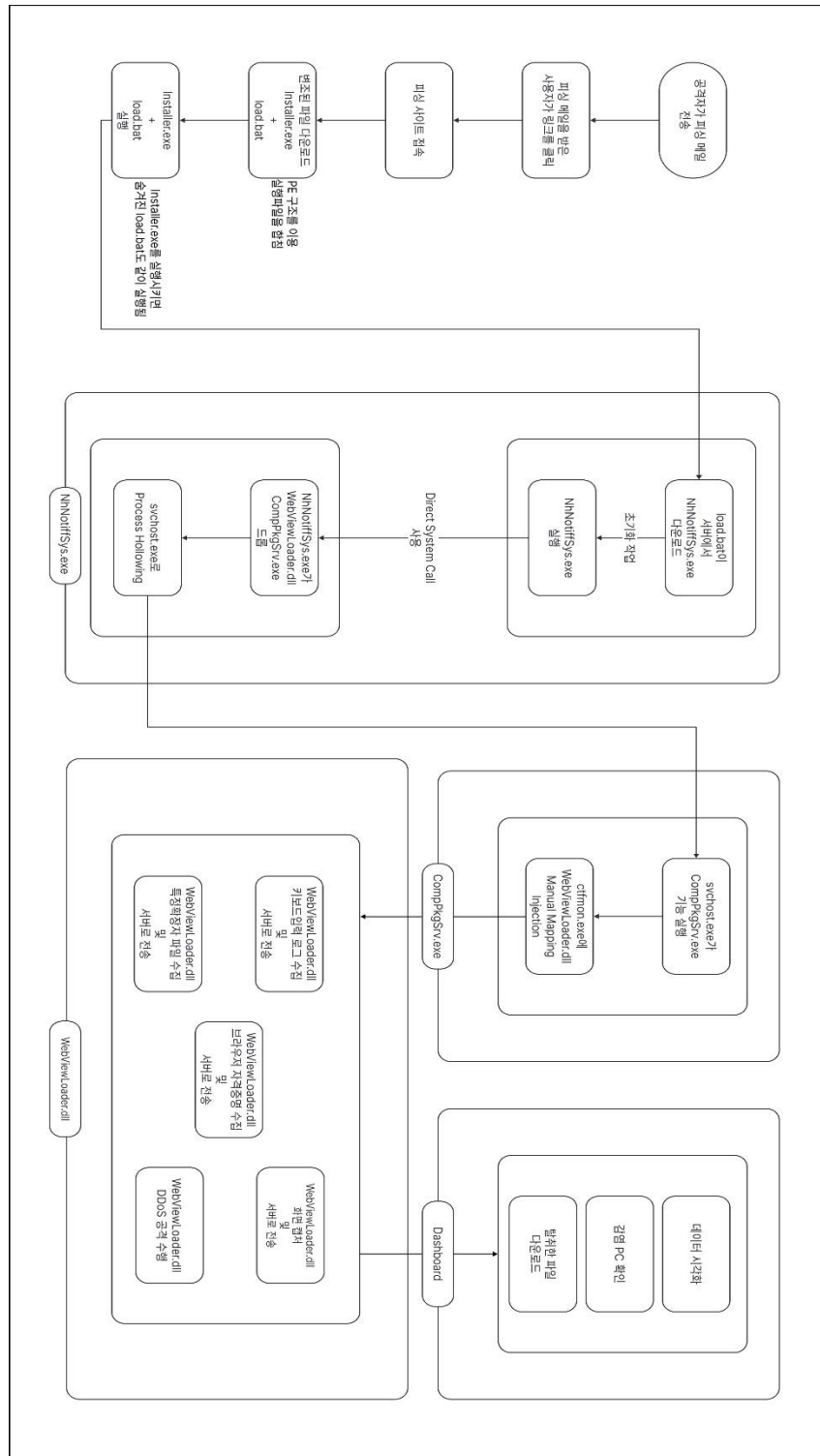
결과적으로, 본 프로젝트는 보안 인력의 인식을 강화하는 유용한 도구로 자리 잡을 뿐만 아니라, 조직의 사이버 보안 체계를 한층 더 견고히 하는 데 실질적인 도움을 줄 것으로 기대된다.

### 1.4 프로젝트 구성

본 프로젝트는 피싱 공격 시나리오를 기반으로 한 악성코드 감염 과정을 재현하여, 보안 위협에 대한 종합적인 이해를 돕고자 한다. 프로젝트의 구성은 다음과 같이 이루어진다.

우선, 공격자는 피싱 이메일을 통해 사용자가 피싱 사이트에 접속하도록 유도한다. 사용자가 피싱 사이트에 접속하면, 정상적인 파일로 위장된 키보드 보안 프로그램 설치를 권유 받게 된다.

이를 실행하는 순간 악성코드에 감염되며, 감염이 이루어지면 공격자와 감염된 시스템 간의 세션이 자동으로 연결된다. 이후 공격자는 원격으로 해당 시스템을 완벽히 제어할 수 있게 된다.



[그림 1-3] 전체 프로젝트 구조

감염된 시스템은 공격자의 대시보드에 연동되며, 대시보드는 실시간으로 감염자의 상태 및 정보를 시각적으로 제공한다. 공격자는 대시보드를 통해 탈취된 데이터를 확인하고 관리하며, 감염 시스템에서 발생하는 모든 활동을 실시간으로 모니터링할 수 있다.

이러한 구성은 피싱을 통해 감염이 이루어지는 단계부터 원격 제어와 데이터 수집까지의 과정을 종합적으로 재현하며, 실제 공격 상황을 재현함으로써 보안 위협의 구체적인 동작 방식을 분석하고 이해할 수 있는 환경을 제공한다.

## 2. 관련 연구

### 2.1 악성코드 재현 및 모델링

악성코드 자체를 개발하는 것은 법적, 윤리적 제약이 있어 직접적인 연구는 제한적이다. 이로 인해 기존 연구들은 악성코드의 분석이나 탐지에 초점이 맞춰져 있으며 실제 시스템 환경에서 악성코드의 동작을 재현하려는 시도는 상대적으로 드물다. 본 연구에서는 이러한 제약을 고려하여 실제 APT(Advanced Persistent Threat) 그룹들이 사용한 Trojan 도구를 예시로 관련 연구를 제시하고자 한다.

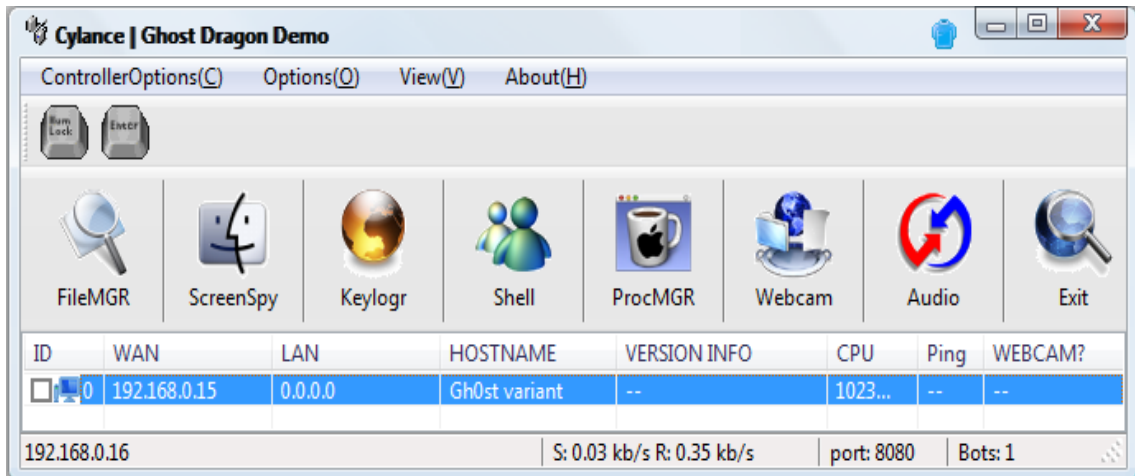
#### 2.1.1 njRAT



[그림 2-1] njRAT 로고

njRAT는 M38dHhM이라는 해킹 조직에 의해 만들어졌으며 강력한 기능과 편리한 사용자 인터페이스를 지원하므로 전 세계적으로 다양한 APT 그룹들에 의해 활용된다. 2013년 6월에 처음 발견되었으며 종종 중동 대상으로 사이버 공격에 사용되었다. njRAT의 주요 기능은 정보 탈취와 시스템 제어를 목적으로 파일 관리, 키로깅, 스크린샷 캡처, 원격 셸 실행, 웹 캠 및 마이크 액세스 등의 기능을 제공한다. 특히 공개적으로 유포된 소스코드와 사용자 가이드를 통해 커뮤니티에서 다양한 버전의 njRAT가 확산되었으며 난독화 및 동적 DNS 서비스를 활용한 C&C 서버 은닉 기술로 탐지를 어렵게 만드는 것이 특징이다.

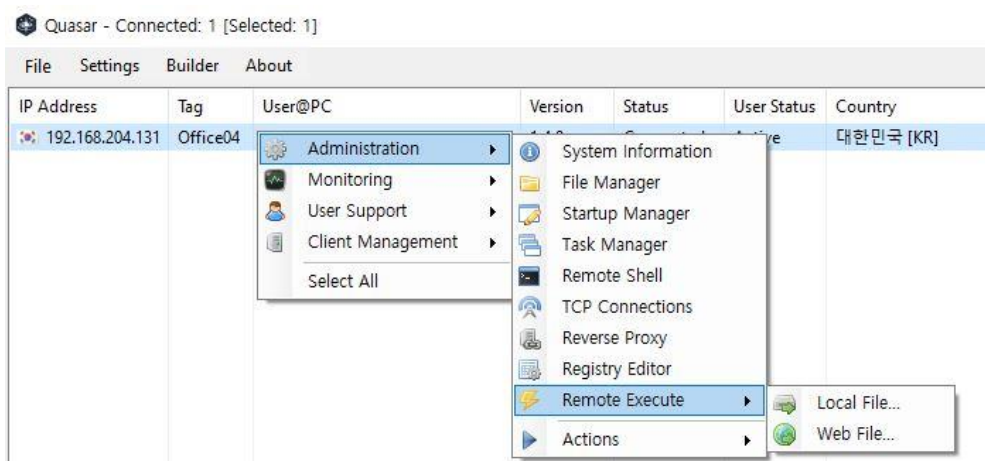
### 2.1.2 Gh0st RAT



[그림 2-2] Gh0st RAT 이미지

Gh0st RAT는 2008년경 중국 기반 해커 그룹에 의해 처음 발견되었다. 이 악성코드는 키로깅, 파일 전송, 화면 및 오디오 캡처, 시스템 정보 수집, 원격 셸 실행 등의 기능을 제공하여 공격자가 피해 시스템을 완전히 제어할 수 있도록 설계되었다. 주로 정부 기관, 기업, 비영리 단체를 대상으로 한 스파이 활동과 사이버 첩보 공격에서 사용되었으며 고급 난독화 기술을 통해 탐지를 회피하는 것이 특징이다. Gh0st RAT는 공개된 소스코드로 인해 다수의 변종이 존재하며 현재까지도 공격자 그룹에 의해 활용되고 있다.

### 2.1.3 Quasar



[그림 2-3] Quasar RAT 이미지

Quasar는 2014년경 Github 사용자 MaxXor가 작성한 C# 기반 오픈소스 원격 관리 도구이다. 합법적인 목적으로 사용될 수 있지만 사이버 범죄 및 사이버 스파이 활동에도 사용된다. Quasar는 사용자 친화적인 인터페이스와 비교적 가벼운 설계로 빠르고 효율적인 원격제어 기능을 제공한다. 주요 기능은 파일 관리, 키로깅, 프로세스 관리, 원격 데스크톱, 네트워크 스니핑, 암호 탈취 등의 기능을 포함한다. 특히 오픈소스 형태로 공개되어 있기 때문에 누구나 쉽게 다운로드하고 수정할 수 있어 공격자들 사이에서 빠르게 확산되었다. 단순한 구성으로 인해 보안 탐지를 우회하고 공개된 소스코드를 수정하여 난독화 및 암호화 기술을 통해 탐지를 어렵게 하는 것이 특징이다.

#### 2.1.4 Cobalt Strike

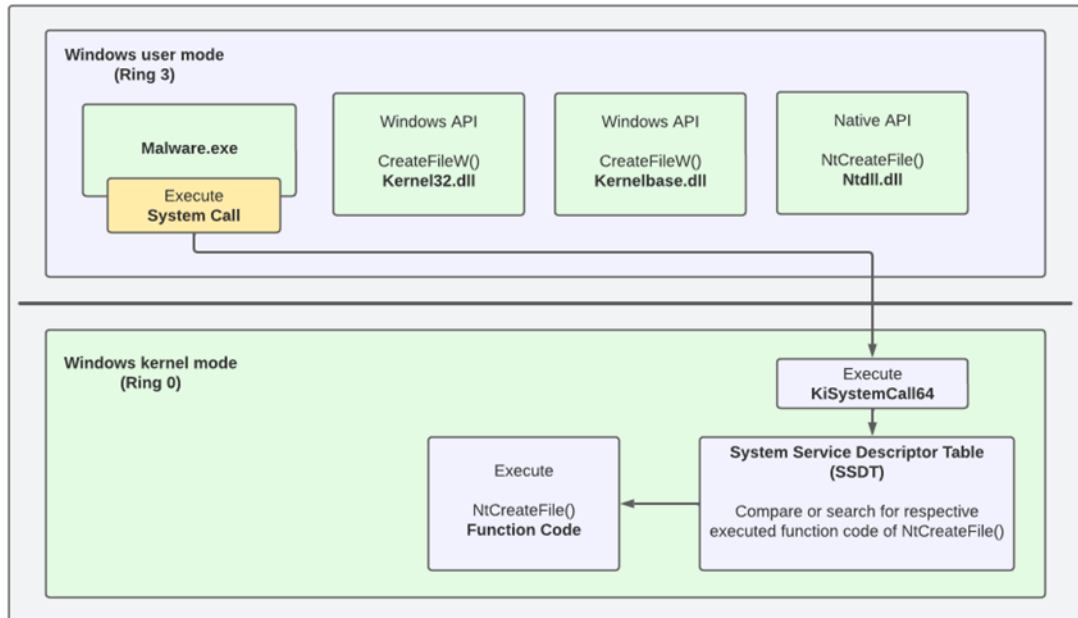


[그림 2-4] Cobalt Strike 로고

Cobalt Strike는 2012년에 Raphael Mudge가 개발한 사용 침투 테스트 도구이다. 2020년 11월 크랙 된 Cobalt Strike 4 버전이 GitHub를 통해 공개된 이후 내부 시스템을 장악하기 위한 목적으로 사용되기 시작했다. 기업들을 대상으로 하는 랜섬웨어 공격 시도에서 공격자들이 내부 시스템 장악을 위한 중간 단계로 Cobalt Strike를 악용하는 사례가 다수 발견되었으며 주요 기능으로는 내부 네트워크 내에서 다른 시스템으로 통제권을 확장할 수 있으며 권한 상승, 자격증명 수집, 파일 탐색, 데이터 수집, 스크린샷 캡처, 키로깅 등이 있다. 특징으로는 C&C 서버와의 통신 데이터를 암호화하여 네트워크 트래픽 분석 도구 탐지를 회피하며 페이로드와 스크립트를 난독화 하여 안티 바이러스 및 EDR 탐지를 우회한다. 주요 랜섬웨어 조직에서 Cobalt Strike를 초기 진입 및 배포 도구로 사용되고 있다.

### 3. 관련 기술

#### 3.1 Direct System Call



The figure shows the transition from Windows user mode to kernel mode in the context of executing malware with implemented direct system calls

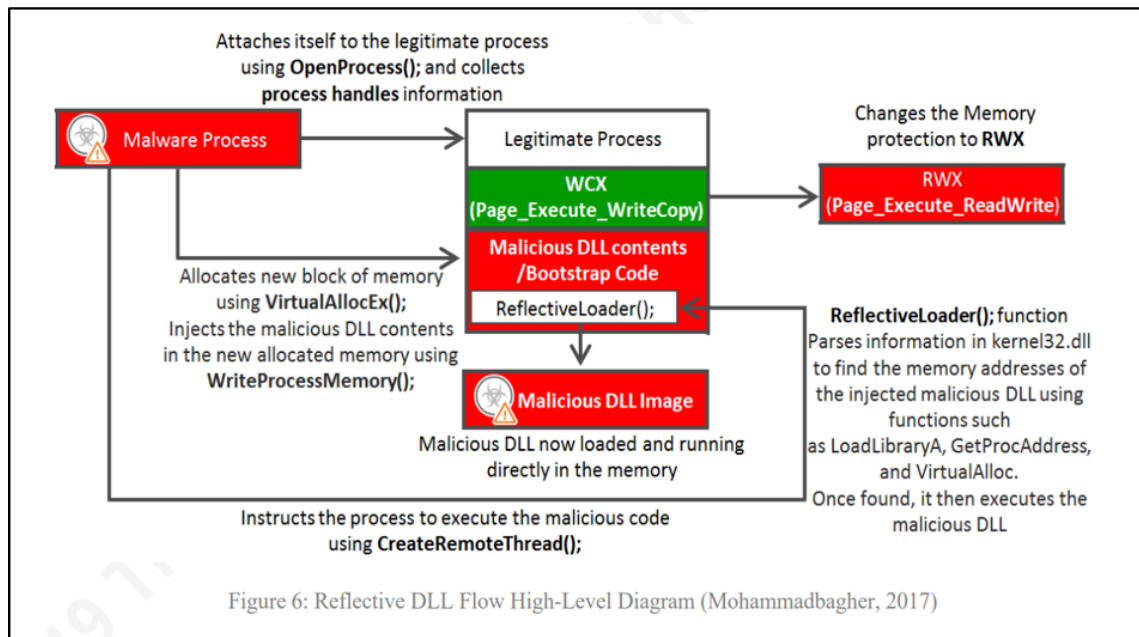
[그림 3-1] Direct System Call 동작 원리

보통 윈도우 운영체제에서는 직접적인 시스템 리소스에 접근하기 위해선 시스템 호출을 통해 커널 자원에 접근하는데 이를 직접 하려면 매우 복잡하며 안정성이 떨어진다. 마이크로소프트에서는 응용 프로그램이 운영 체제의 다양한 기능을 손쉽게 사용할 수 있도록 Windows API를 제공한다. Windows API는 윈도우 운영체제에서 사용자 애플리케이션이 시스템 리소스에 직접 접근하지 않고 운영체제의 보호 아래에서 자원에 접근할 수 있도록 제공되는 API를 말한다. 이때, Windows API를 통해 수행되는 작업 중 커널의 기능을 필요로 하는 작업들은 시스템 콜(System Call)을 수행해 커널 인터페이스와 연결된다.

일부 고급 애플리케이션 특히 성능 최적화나 보안 연구를 목적으로 하는 프로그램은 Windows API를 거치지 않고 시스템 콜을 직접 호출하는 Direct System Call 방식을 사용하기도 한다. Direct System Call은 API 호출을 오버헤드를 줄이거나 보안 소프트웨어가 모니터링하는 API Hooking을 우회하기 위해 시스템 콜을 호출하여 커널의 기능을 수행한다. 해당 기법은 주로 특정 커널 함수의 주소와 시스템 콜 번호(SSN)를 구하고 어셈블리 명령을 사용해 커널로 진입하게 한다. 본

프로젝트에서는 보안 솔루션의 API Hooking을 회피하고 은밀하게 시스템 리소스를 조작 및 자원을 얻기 위함으로써 해당 기법을 채택하기로 했다.

### 3.2 DLL(Dynamic Link Library) Manual Mapping



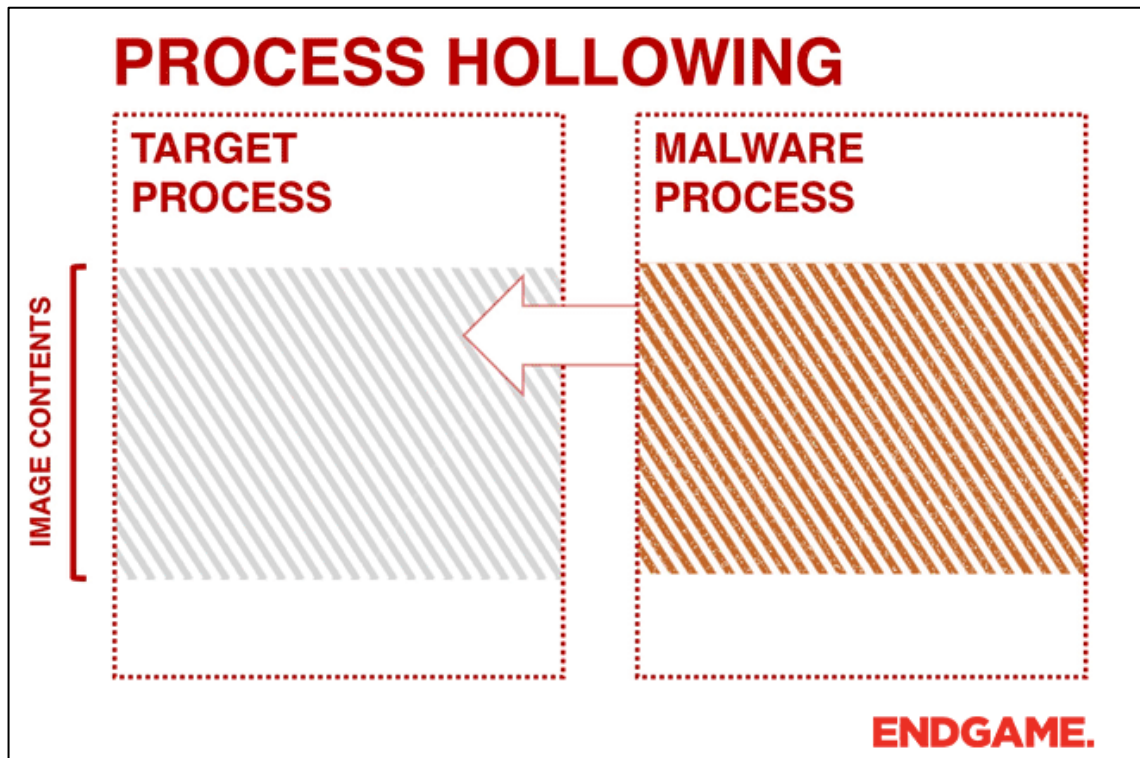
[그림 3-2] DLL Manual Mapping 동작 원리

DLL Manual Mapping은 운영 체제의 표준 DLL(Dynamic Link Library, 동적 링크 라이브러리) 불러오기 메커니즘을 우회하여 악성코드가 메모리 내 특정 DLL을 수동으로 불러오고 실행할 수 있게 하는 기법이다. 일반적으로 윈도우 운영 체제에서는 DLL을 로드할 때 유저모드(ring3)에서 `LoadLibrary()`를 호출하여 파일 시스템을 거쳐서 로드 된다.

[그림 2-2]와 같이 DLL Manual Mapping은 타겟 프로세스에 `VirtualAllocEx()`로 타겟 프로세스에 가상 메모리 공간을 할당하고 각 Section과 IAT(Import Address Table), EAT(Export Address Table) 테이블 셋업을 직접 처리하고 `WriteProcessMemory()`를 사용하여 DLL 바이너리를 타겟 프로세스에 적재 시킨다. 메모리에 적재되면 `CreateRemoteThread()`로 DLL OEP(Original Entry Point)를 실행하여 DLL 내용이 동작하게 된다. 기존 DLL Injection 기법과는 다르게 표준 라이브러리 호출 없이 수동으로 DLL을 로드 한다는 점에서 차이가 있다. 이처럼 해당 기법은 표준 라이브러리를 사용하지 않고 DLL을 직접 프로세스 메모리에 인젝션 하는 특징에서 인젝션 된 모듈이 바인드 되기 때문에 보안 소프트웨어가 탐지하기 어려운 방식으로 동작할 수 있게 된다.



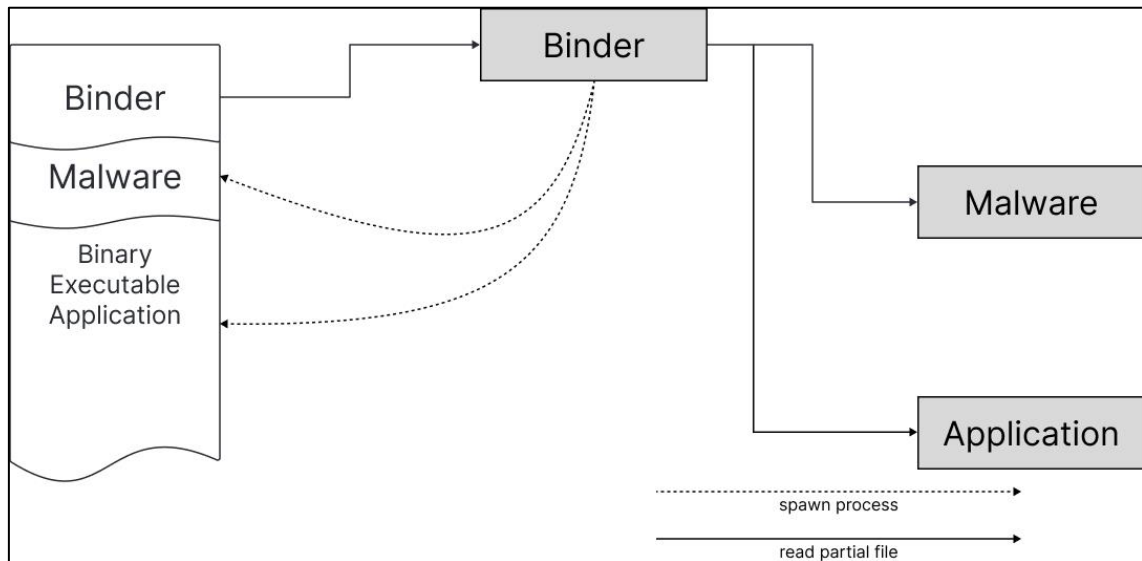
### 3.3 Process Hollowing



[그림 3-3] Process Hollowing 동작 원리

Process Hollowing은 악성코드가 주로 사용하는 기술로 대상 프로세스의 이미지를 언매핑하고 자신의 이미지를 매핑하는 기술이다. 타겟 프로세스를 svchost.exe와 같은 프로세스를 SUSPEND모드로 CreateProcess()를 호출하여 프로세스를 생성한다. 생성된 프로세스에 UnmapViewOfSection()를 사용하여 이미지를 언매핑 시켜 아무것도 동작하지 않는 빈 프로세스로 만든다. VirtualAllocEx()를 호출하여 프로세스에 가상 메모리 공간을 할당한 뒤 자신의 이미지를 WriteProcessMemory()로 매핑하고 이미지 재배포 작업을 수행한다. 이후 SetThreadContext()로 새로운 코드 섹션으로 가리키게 하고 ResumeThread()로 스레드의 실행을 재개한다. 이 기법을 통해 악성코드는 시스템 내에서 합법적인 프로세스의 외형을 유지하면서도 실제로는 악의적인 동작을 수행하게 된다. 이 특징으로 인해 탐지 솔루션의 감시에서 벗어나게 되고 컴퓨터 사용자 입장에서도 정상 프로세스로 식별 되기 때문에 악성코드를 은폐하는 수단으로 주로 사용된다.

### 3.4 Executable Binder



[그림 3-4] Executable Binder 동작 원리

Executable Binder는 두 개 이상의 실행 파일(.exe)을 하나의 실행 파일로 결합하는 도구를 의미한다. 주로 악성코드 제작자들이 정상적인 프로그램과 악성코드를 합쳐 컴퓨터 사용자가 정상적인 파일은 의심 없이 실행한다는 점을 노려 사용자가 의심 없이 직접 악성코드를 실행할 수 있도록 하는데 Executable Binder가 사용된다. 주요 방식은 실행 파일의 섹션 구조를 분석하여 파일의 데이터를 서로 병합한다. 실행 시 하나의 스레드에서 정상 파일이 동작하도록 하고 또 다른 스레드에서 악성코드가 백그라운드로 실행되도록 설정한다. 이는 사용자가 정상 프로그램 실행 화면을 인지하도록 만들어 악성코드가 백그라운드에서 은밀히 실행되도록 설계된다. 악성 파일이 독립적으로 존재하지 않기 때문에 사용자에게 있어 의심을 피할 수 있고 난독화, 암호화, 파일리스(fileless) 실행 등과 함께 적용될 경우 백신 소프트웨어에서 탐지가 힘들어진다. 주로 소셜 엔지니어링 기법과 결합하여 사용자를 속이는 데 활용된다.

## 4. 프로젝트 구현

### 4.1 테스트 환경 구성

본 프로젝트에서 개발 및 실행된 환경은 다음과 같다.

구성	서버
Product Name	VMware Virtual Platform
OS	Ubuntu 22.04.4 LTS
CPU	Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz
Memory	15GB
HDD	196GB

[표 4-1] C2서버 구동 환경

구성	클라이언트
Product Name	Virtual Machine Hyper-V
OS	Microsoft Windows 10 Pro 10.0.19045
CPU	11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz
Memory	2GB
HDD	130GB

[표 4-2] 클라이언트 구동 환경

구성	개발 환경
IDE	Visual Studio Community 2019
installation Version	16.11.34931.43
Build	Release – x64
Language	C++17
Version Control	Git, GitHub

[표 4-3] 모의 악성코드 개발 환경

구성	대시보드
Python	3.10.12
Pip	22.0.2
Libraries	Flask 3.0.3 JinJa2 3.1.4
Database	SQLite 3.37.2
Docker	27.3.1
Docker Compose	2.29.7

[표 4-4] 대시보드 환경

C2서버는 Linux 환경에서 동작하도록 설계되었으며, 이를 위해 Ubuntu 가상 서버에 C2서버와 피싱 사이트, 웹 대시보드 서버를 구축 하였다.

각각의 구성 요소는 다음과 같은 기술 스택을 사용하여 개발되었다.

- (1) C2 서버: 악성코드 제어 및 명령 전송을 위해 C/C++로 구현하였으며, 서버-클라이언트 간 효율적인 데이터 교환을 지원하도록 설계되었다,
- (2) 웹 대시보드 서버: Python 기반 Flask와 Jinja2 템플릿 엔진을 활용하여 구현되었으며, 관리자가 악성코드의 세션, 파일 및 시스템 활동을 직관적으로 모니터링 하고 제어할 수 있도록 기능을 제공한다.
- (3) 피싱 사이트: 사용자 유인을 위한 페이지를 구현하기 위해 Nginx를 기반으로 개발하였다.

#### 4.1.1 환경 구축

본 프로젝트는 Docker 환경을 활용하여 C2서버와 관련된 개발 및 실행 환경을 컨테이너로 구성하였다. 이를 통해 사용자는 다른 시스템에서도 별도의 복잡한 설정 없이 Docker를 통해 환경을 간편하게 구축하고 실행을 진행할 수 있도록 설계하였다. 이러한 환경 구성은 다양한 운영체제와 하드웨어 환경에서도 일관된 실행 환경을 제공한다.

```
wget http://apty.leong.kr/project.tar.gz -O project.tar.gz
```

```
testv@DESKTOP-JDU01B9:~$ wget http://apty.leong.kr/project.tar.gz -O project.tar.gz
--2024-11-26 05:43:23-- http://apty.leong.kr/project.tar.gz
Resolving apty.leong.kr (apty.leong.kr)... 106.240.41.77
Connecting to apty.leong.kr (apty.leong.kr)|106.240.41.77|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9762519 (9.3M) [application/octet-stream]
Saving to: 'project.tar.gz'

project.tar.gz      100%[=====] 9.31M  2.84MB/s  in 3.3s
2024-11-26 05:43:27 (2.84 MB/s) - 'project.tar.gz' saved [9762519/9762519]

testv@DESKTOP-JDU01B9:~$ tar -xzf project.tar.gz
testv@DESKTOP-JDU01B9:~$ ls
Desktop  project.tar.gz
testv@DESKTOP-JDU01B9:~$ cd Desktop/
testv@DESKTOP-JDU01B9:~/Desktop$ ls
Dockerfile  app  docker-compose.yml  html  nginx.conf  start.sh  supervisord.conf
```

[표 4-5] wget을 이용하여 도커 파일 설치

app	2024-11-26 오전 4:50	파일 폴더	
html	2024-11-26 오전 4:32	파일 폴더	
.dockerignore	2024-11-24 오전 3:00	DOCKERIGNORE ...	1KB
docker-compose	2024-11-26 오전 4:29	Yaml 원본 파일	1KB
Dockerfile	2024-11-26 오전 4:41	파일	2KB
nginx.conf	2024-11-26 오전 4:07	CONF 파일	1KB
start	2024-11-26 오전 4:29	SH 원본 파일	2KB
supervisord.conf	2024-11-26 오전 4:07	CONF 파일	1KB

[그림 4-1] 도커 파일의 구조

```
sudo ./start.sh
```

```
testv@DESKTOP-JDU01B9:~/Desktop$ sudo ./start.sh
[sudo] password for testv:
Docker is not installed. Installing now...
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20240203~22.04.1).
ca-certificates set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
```

[표 4-6] start.sh 스크립트 실행(1)

```
sudo ./start.sh
```

```
testv@DESKTOP-JDU0189:~/Desktop$ sudo ./start.sh
Docker is already installed.
Docker Compose is already installed.
Starting Docker containers...
WARN[0000] /home/testv/Desktop/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Building 46.8s (17/17) FINISHED                                docker:default
=> [app internal] load build definition from Dockerfile            0.0s
=> => transferring dockerfile: 1.28kB                               0.0s
=> [app internal] load metadata for docker.io/library/ubuntu:22.04 2.3s
=> [app internal] load .dockerignore                               0.0s
=> => transferring context: 103B                                       0.0s
=> [app 1/11] FROM docker.io/library/ubuntu:22.04@sha256:0e5e4a57c2499249aafc3b40fcd541e9a456aab72966 2.2s
=> => resolve docker.io/library/ubuntu:22.04@sha256:0e5e4a57c2499249aafc3b40fcd541e9a456aab7296681a399 0.0s
=> => sha256:0e5e4a57c2499249aafc3b40fcd541e9a456aab7296681a3994d631587203f97 6.69kB / 6.69kB 0.0s
=> => sha256:3d1556a8a18cf5307b121e0a98e93f1dddf1f3f8e092f1ddfd941254785b95d7 424B / 424B 0.0s
=> => sha256:97271d29cb7956f0908c-fb1449610a2cd9cb46b004ac8af25f0255663eb364ba 2.30kB / 2.30kB 0.0s
=> => sha256:6414378b647780fee8fd903ddb9541d134a1947ce092d08bdeb23a54cb3684ac 29.54MB / 29.54MB 1.3s
=> => extracting sha256:6414378b647780fee8fd903ddb9541d134a1947ce092d08bdeb23a54cb3684ac 0.8s
=> [app internal] load build context                               0.1s
=> => transferring context: 25.00MB                                    0.1s
=> [app 2/11] RUN sed -i 's|http://archive.ubuntu.com|http://mirror.kakao.com|g' /etc/apt/sources.lis 3.3s
=> [app 2/11] RUN apt-get install -y mailutils union mount2 authn3 authn3-pia calico3 supervisor 30.2s
```

[표 4-7] start.sh 스크립트 실행(2)

```
[+] Running 2/2
✔Network desktop_default Created 0.2s
✔Container APTY_app Started 0.3s
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
41a832e75fbd desktop-app "/usr/bin/supervisor..." Less than a second ago Up Less than a second 0.0.0.0
:80->80/tcp, ::80->80/tcp, 0.0.0.0:443->443/tcp, ::443->443/tcp, 0.0.0.0:8888->8888/tcp, ::8888->8888/tcp
APTY_app
```

[그림 4-2] 도커 실행 및 컨테이너 적용

```
start.sh
```

```
#!/bin/bash
```

```
# Docker 설치 여부 확인
```

```
if ! command -v docker &> /dev/null
```

```
then
```

```
    echo "Docker is not installed. Installing now..."
```

```
    # 필수 도구 설치
```

```
    sudo apt-get update
```

```
    sudo apt-get install -y ca-certificates curl gnupg lsb-release
```

```
    # Docker GPG 키 추가
```

```
    sudo mkdir -p /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg

# Docker 리포지토리 추가

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null


# Docker 최신 버전 설치

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin


# Docker 권한 설정

echo "Adding current user to the Docker group..."

sudo usermod -aG docker $USER


# Docker 서비스 시작 및 활성화

echo "Starting and enabling Docker service..."

sudo systemctl start docker

sudo systemctl enable docker


echo "Docker installation complete. Please log out and log back in for group changes
to take effect."

exit 0

else

echo "Docker is already installed."

fi


# Docker Compose 설치 여부 확인

if ! docker compose version &> /dev/null

then

echo "Docker Compose plugin is not installed. Installing now..."
```

```
# Docker Compose Plugin 설치 (Docker 설치 과정에서 이미 포함)
sudo apt-get install -y docker-compose-plugin

echo "Docker Compose installation complete."
else
    echo "Docker Compose is already installed."
fi

# Docker Compose로 컨테이너 빌드 및 실행
if [ -f "docker-compose.yml" ]; then
    echo "Starting Docker containers..."
    docker compose up --build -d

    # 실행 중인 컨테이너 목록 표시
    docker ps
else
    echo "docker-compose.yml not found in the current directory. Please ensure the file exists."
fi
```

위 스크립트는 Docker 와 Docker Compose 가 설치되어 있지 않은 경우 이를 자동으로 설치하고, 현재 디렉토리에 있는 docker-compose.yml 파일을 기반으로 Docker 컨테이너를 빌드 및 실행하는 자동화 스크립트다.

먼저, Docker 가 설치되어 있는지 확인하고, 설치되지 않았다면 Docker 의 GPG 키와 저장소를 추가한 후 필수 패키지를 설치한다. 그 후, Docker 서비스를 활성화하고 현재 사용자를 Docker 그룹에 추가하고 Docker Compose 플러그인의 설치 여부를 확인하고, 필요 시 이를 설치한다. 마지막으로 docker-compose.yml 파일이 있는 경우 이를 이용해 컨테이너를 빌드 및 실행하며, 실행 중인 컨테이너 목록을 출력한다.



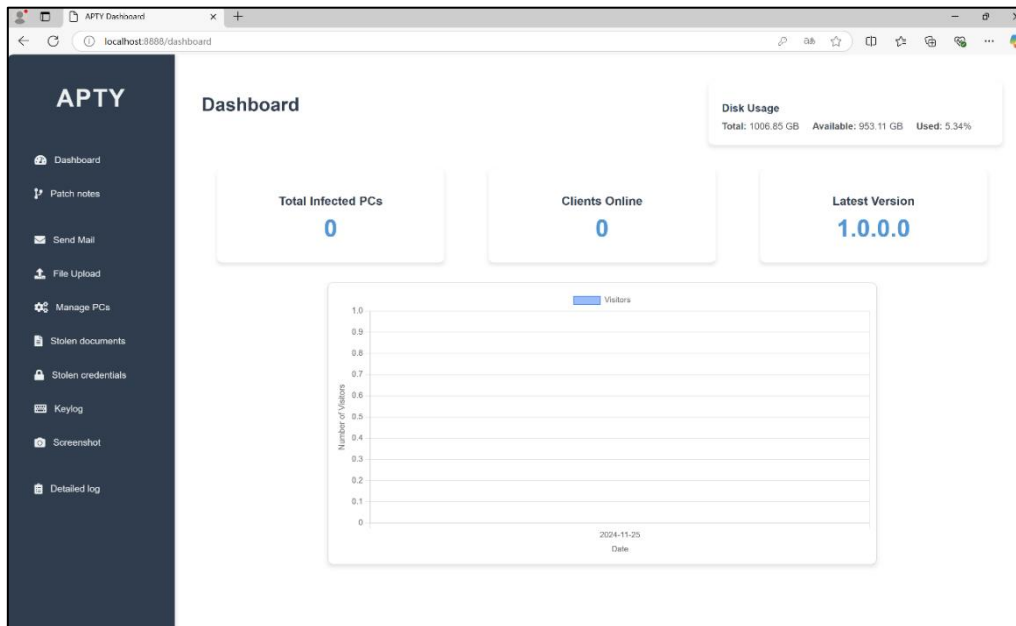
## docker-compose.yml

```
version: '3.8'

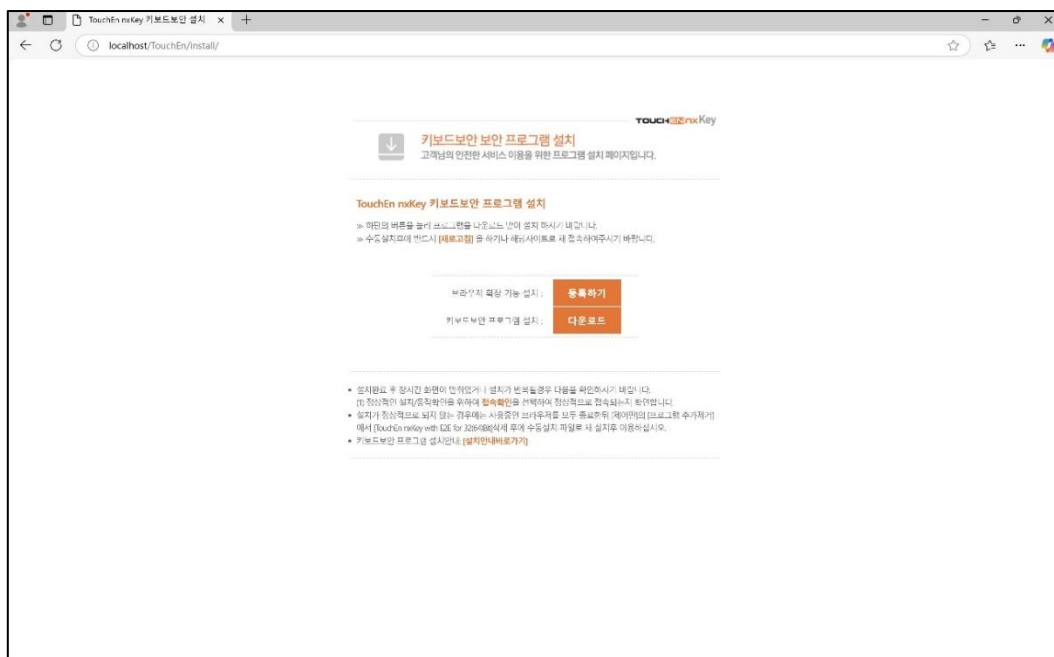
services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: APTY_app
    hostname: google # hostname 설정
    ports:
      - "80:80"
      - "443:443"
      - "8888:8888"
    volumes:
      - ./html:/var/www/html # 정적 파일 공유
      - ./app:/home/Microsoft/app # Flask 애플리케이션 공유
    environment:
      - USERNAME=Microsoft
    restart: always
```

위 파일은 Docker Compose 를 사용해 Flask 애플리케이션 환경을 설정한다. app 이라는 이름의 서비스는 현재 디렉토리의 도커 파일을 기반으로 이미지를 빌드하며, 컨테이너 이름을 APTY\_app, 호스트 이름을 google 로 설정한다. 80, 443, 8888 포트를 호스트와 연결하고, 정적 파일(/html)과 Flask 애플리케이션 코드(/app)를 각각 컨테이너의 /var/www/html 과 /home/Microsoft/app 경로에 공유한다.

또한 환경 변수로 USERNAME=Microsoft 를 설정하고, 컨테이너가 종료되더라도 자동으로 다시 시작되도록 restart: always 정책을 적용한다. 이 설정은 Flask 애플리케이션 실행에 필요한 파일 공유와 포트 매핑을 제공하며, 안정적인 개발 환경을 구축한다.



[그림 4-3] 컨테이너로 생성된 웹 서버

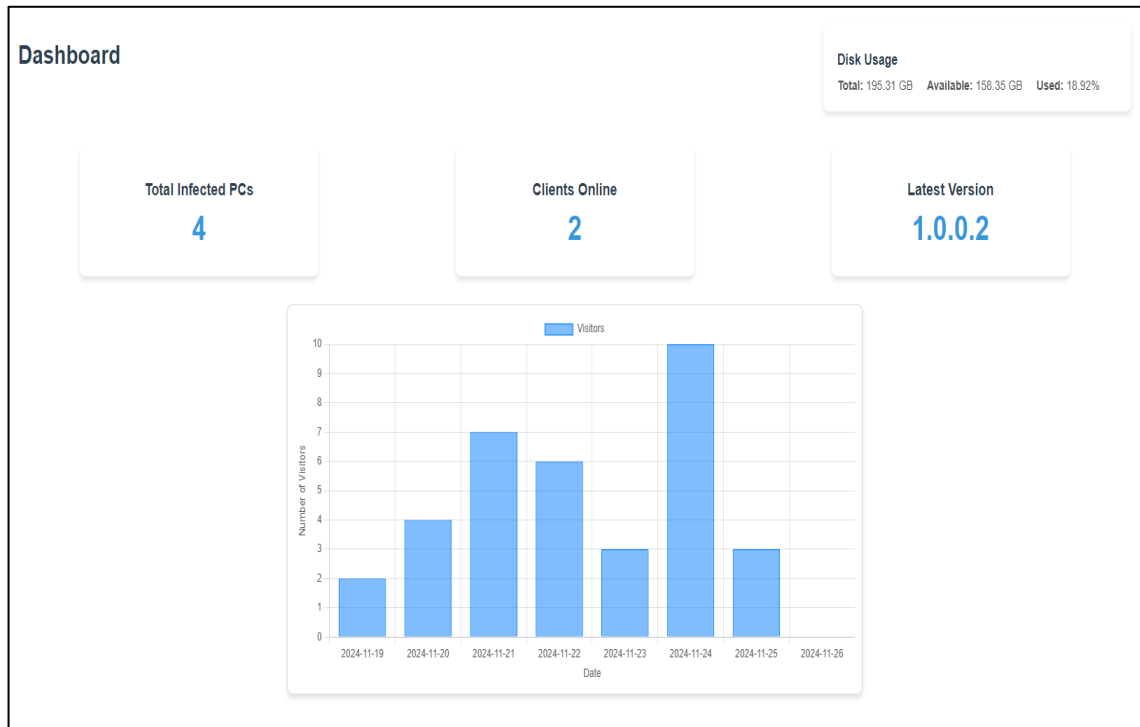


[그림 4-4] 컨테이너로 생성된 피싱 사이트

start.sh 수행이 완료되면 localhost:8888(웹 대시보드), localhost:80(피싱 사이트)에서 각 웹 사이트들이 동작하고, 공격자측 서버 환경을 구성 할 수 있게 된다.

## 4.2 관리용 플랫폼 개발

### 4.2.1 대시보드 메인 페이지



[그림 4-5] 대시보드 메인 페이지

웹 대시보드는 네트워크 상태와 악성 파일 배포 현황을 직관적으로 제공하는 관리 도구로 설계되었다. 대시보드에는 총 감염된 클라이언트 수, 서버와 연결된 활성 클라이언트 수, 악성 파일 배포 버전, 서버 저장소 사용량 등의 주요 지표가 포함된다. 이러한 정보는 관리자가 감염 현황과 자원 사용 상태를 실시간으로 확인할 수 있도록 돕는다.

피싱 사이트 접속자 수는 일별 막대 차트로 시각화 되어 접속 추세를 한눈에 파악할 수 있도록 구성되었으며, 피싱 공격의 효과 및 특정 기간의 활동 패턴을 분석하는 데 유용하다.

#### 4.2.2 피싱 메일 전송 페이지

[그림 4-6] 피싱 메일 전송 페이지

해당 페이지에서는 이메일 본문에 HTML 코드를 첨부하여 사용자 지정 메시지를 작성할 수 있으며, 송신자 이메일 주소를 변경하여 이메일을 전송할 수 있는 기능을 제공한다. 이를 통해 실제 피싱 공격에서 활용되는 이메일 형태를 보다 정밀하게 모사할 수 있다. 특히, HTML을 활용한 이메일 본문 작성 기능은 피싱 메일의 주요 특징 중 하나인 링크 위장과 시각적 설득 요소를 구현하는 데 사용된다.

그러나 메일 서버의 보안 정책에 따라 송신자 이메일 변경 기능은 제한적이다. 예를 들어, 공공 SMTP 서버인 Naver 나 Gmail 과 같은 서비스는 엄격한 인증 절차(SPF, DKIM, DMARC 등)를 요구하므로, 송신자 이메일 주소를 변경하여 발송하는 것이 불가능하다. 이에 따라 테스트

목적의 이메일 전송에는 사용자 정의 SMTP 서버를 사용하거나 보안 설정이 완화된 테스트 환경이 필요하다.

또한, 플랫폼의 악용 가능성을 방지하기 위해 전송된 이메일은 지정된 테스트 계정으로만 발송되도록 제한을 두었다. 이를 통해 플랫폼의 사용 목적이 교육 및 연구를 위한 모의 환경 조성임을 명확히 하고, 실제 악성 행위로 악용될 가능성을 최소화하였다.

#### 피싱 메일 전송

```
@app.route('/send_mail', methods=['GET', 'POST'])

@login_required

def send_mail():

    if request.method == 'POST':

        subject = sanitize_input(request.form['subject'])

        sender = sanitize_input(request.form['sender'])

        recipient = sanitize_input(request.form['recipient'])

        message_body = request.form['message']

        attachment = request.files.get('attachment')

        # Prepare attachment if provided

        attachment_path = None

        if attachment:

            filename = secure_filename(attachment.filename)

            attachment_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
```

```
attachment.save(attachment_path)

# Encode the subject in UTF-8

subject_encoded = base64.b64encode(subject.encode('utf-8')).decode('utf-8')

subject_formatted = f"?UTF-8?B?{subject_encoded}?="

# Build mail command

cmd = [

    "mail",

    "-s", subject_formatted,

    "-r", sender,

    "-a", "Content-Type: text/html; charset=UTF-8",

    recipient

]

# Add attachment option if available

if attachment_path:

    cmd.extend(["-A", attachment_path])

# Execute command

try:

    process = subprocess.Popen(cmd, stdin=subprocess.PIPE, text=True)

    process.communicate(input=message_body)
```

```
if process.returncode == 0:

    flash("Email sent successfully!", "success")

else:

    flash("Failed to send email.", "error")

except Exception as e:

    flash(f"Error: {str(e)}", "error")

# Clean up uploaded file

if attachment_path:

    os.remove(attachment_path)

return redirect(url_for('send_mail'))

return render_template('send_mail.html')
```

### 4.2.3 악성 파일 업로드 페이지

[그림 4-7] 악성 파일 업로드 페이지

해당 페이지는 악성파일을 최신화 하기 위한 업로드 페이지다. 이곳에 악성 파일을 업로드 하게 되면 서버에는 NhNotiffSys.exe로 이름을 변경한 상태로 저장한다. 이로인해 악성파일을 최신화 및 관리 하기 용이하도록 구성하였다.

#### 최신 악성 파일 업로드

```
@app.route('/upload', methods=['GET', 'POST'])
@login_required
def upload_file():
    if request.method == 'GET':
        logs = read_logs()
        return render_template('upload.html', logs=logs)
    elif request.method == 'POST':
        if 'file' not in request.files:
            write_log(request.remote_addr, "No file", "실패")
            return jsonify({"status": "error", "message": "No file part"}), 400
        file = request.files['file']
        if file.filename == '':
            write_log(request.remote_addr, "No file", "실패")
            return jsonify({"status": "error", "message": "No selected file"}), 400
```



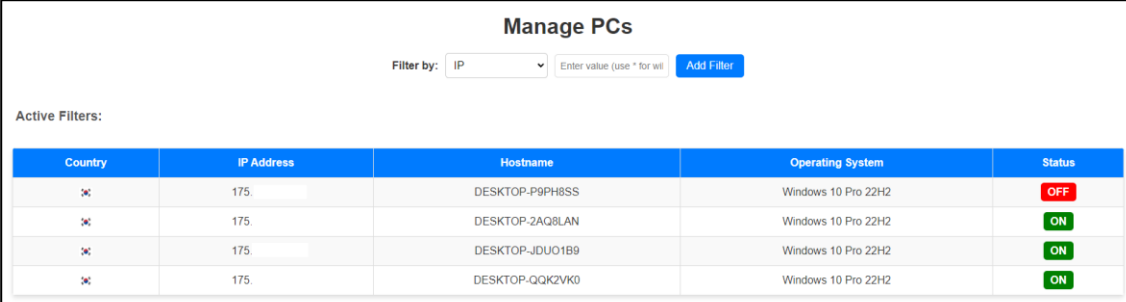
```

# 원래 파일명을 저장
original_filename = file.filename

# 저장될 파일명은 고정
save_path = os.path.join(app.config['UPLOAD_FOLDER'], "NhNotiffSys.exe")
try:
    file.save(save_path)
    write_log(request.remote_addr, original_filename, "완료")
    return jsonify({"status": "success", "message": f"File '{original_filename}' uploaded successfully!"})
except Exception as e:
    write_log(request.remote_addr, original_filename, f"실패: {str(e)}")
    return jsonify({"status": "error", "message": f"Failed to save file: {str(e)}"}, 500

```

#### 4.2.4 클라이언트 관리 페이지

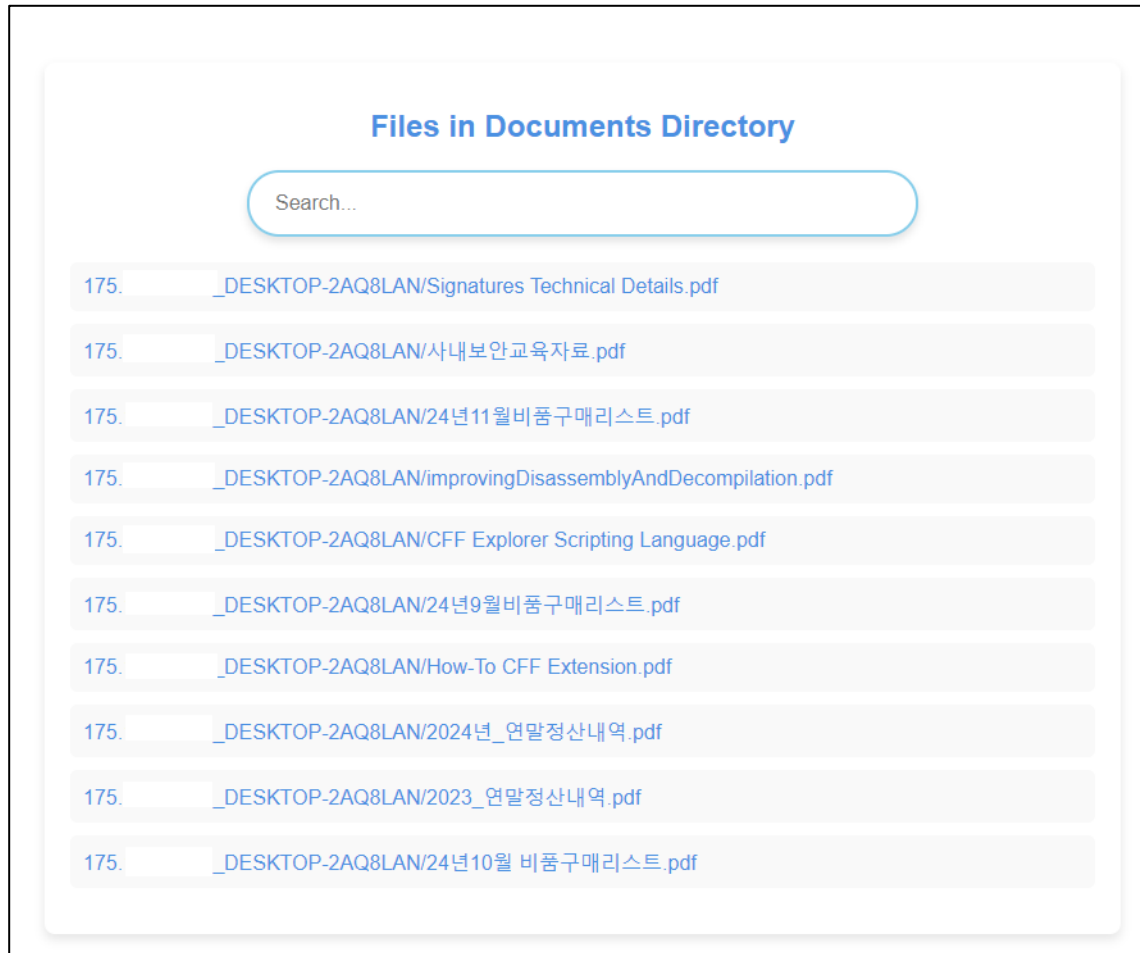


Country	IP Address	Hostname	Operating System	Status
🇰🇷	175.	DESKTOP-P9PH8SS	Windows 10 Pro 22H2	OFF
🇰🇷	175.	DESKTOP-2AQ8LAN	Windows 10 Pro 22H2	ON
🇰🇷	175.	DESKTOP-JDUO1B9	Windows 10 Pro 22H2	ON
🇰🇷	175.	DESKTOP-QQK2VK0	Windows 10 Pro 22H2	ON

[그림 4-8] 클라이언트 관리 페이지

해당 페이지는 감염된 클라이언트들의 정보를 출력해주는 관리 페이지로, ip, 호스트네임, OS 버전, 현재 연결 상태와 ip를 기반으로 위치정보를 확인하여 각 클라이언트가 속한 국가를 시각적으로 출력한다. 또한 페이지는 실시간으로 연결 상태를 갱신하여 클라이언트의 연결이 종료되면 이것을 즉각적으로 반영한다.

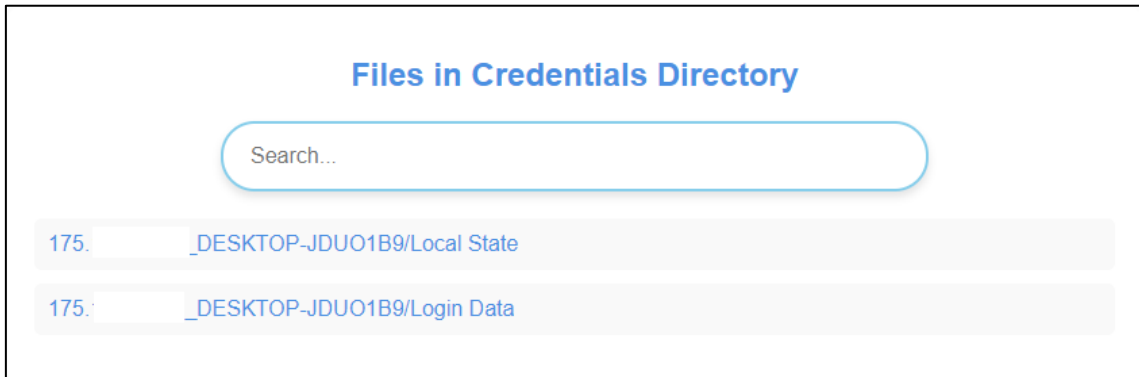
#### 4.2.5 탈취 파일 관리 페이지



[그림 4-9] 탈취 파일 관리 페이지

해당 페이지는 감염된 클라이언트로부터 탈취한 파일 목록들을 확인하고, 특정 파일을 다운로드할 수 있다. 페이지에는 <ip\_호스트네임/파일명> 형태로 나타내며 이것으로 어떤 클라이언트에서 탈취한 파일인지 확인 할 수 있다.

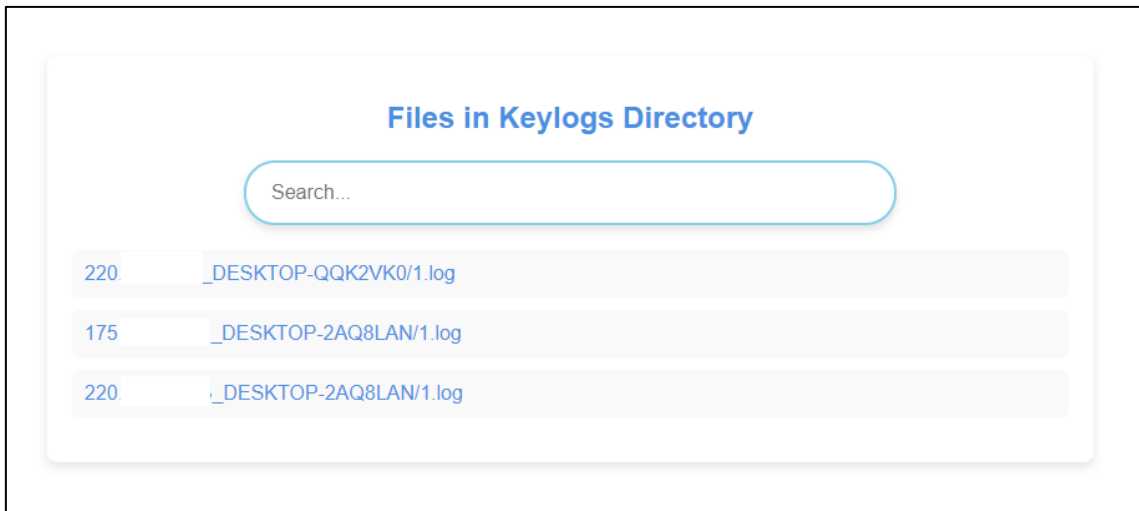
#### 4.2.6 브라우저 자격 증명 탈취 페이지



[그림 4-10] 자격 증명 탈취 파일 관리 페이지

해당 페이지는 클라이언트들의 Chrome browser에서 수집된 자격 증명 파일을 관리하는 페이지이다. 이 페이지에서는 파일들을 <ip\_호스트네임/Login Data>, <ip\_호스트네임/Local State>파일로 나열하여 어떤 클라이언트의 자격 증명 파일인지 확인 할 수 있다.

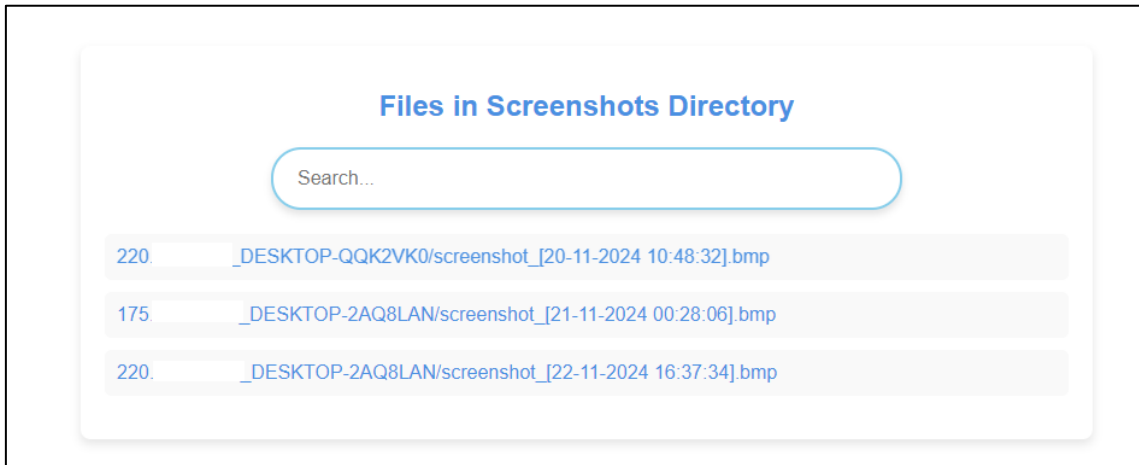
#### 4.2.7 키로깅 페이지



[그림 4-11] 키로그 파일 관리 페이지

해당 페이지는 클라이언트에서 수집된 키로그 파일들의 목록을 표시하며, 다운로드 할 수 있다. 페이지에는 <ip\_호스트네임/파일명>형태로 나타내며 이것으로 어떤 클라이언트에서 기록된 로그 파일인지 확인 할 수 있다.

#### 4.2.8 화면 캡처 페이지



[그림 4-12] 화면 캡처 파일 관리 페이지

해당 페이지는 클라이언트의 화면을 캡처한 파일들의 목록을 표시하며, 열람 및 다운로드 할 수 있다. 페이지에는 <ip\_호스트네임/[타임스태프]> 형태로 나타내며 이것으로 어떤 클라이언트에서 기록된 로그 파일인지 확인 할 수 있다.

#### 4.2.9 상세 로그 페이지



**[그림 4-13] 상세 로그 출력 페이지**

해당 페이지는 클라이언트와 서버 간의 상호작용, 데이터 전송 및 명령 실행 결과와 같은 다양한 활동 기록을 시간 순으로 출력해준다.

**상세 로그 출력**

```
@app.route('/logs')
@login_required
def view_logs():
    return render_template('logs.html')

@app.route('/api/logs')
def get_logs():
    if os.path.exists(LOG_FILE_PATH):
        with open(LOG_FILE_PATH, 'r', encoding='utf-8', errors='ignore') as f:
            logs = f.readlines() # 최근 로그 가져오기
    else:
        logs = ["No logs available."]
    return jsonify(logs)

<script>
// 주기적으로 로그 데이터를 가져오는 함수
function fetchLogs() {
    fetch('/api/logs')
        .then(response => response.json())
        .then(data => {
            const logContainer = document.getElementById('log-container');
            const formattedLogs = data
                .filter(log => log.trim() !== '')
                .map(log => `<div class="log-entry">${log}</div>`)
                .join("");
            logContainer.innerHTML = formattedLogs;
            logContainer.scrollTop = logContainer.scrollHeight;
```

```
// 항상 스크롤을 맨 아래로 유지

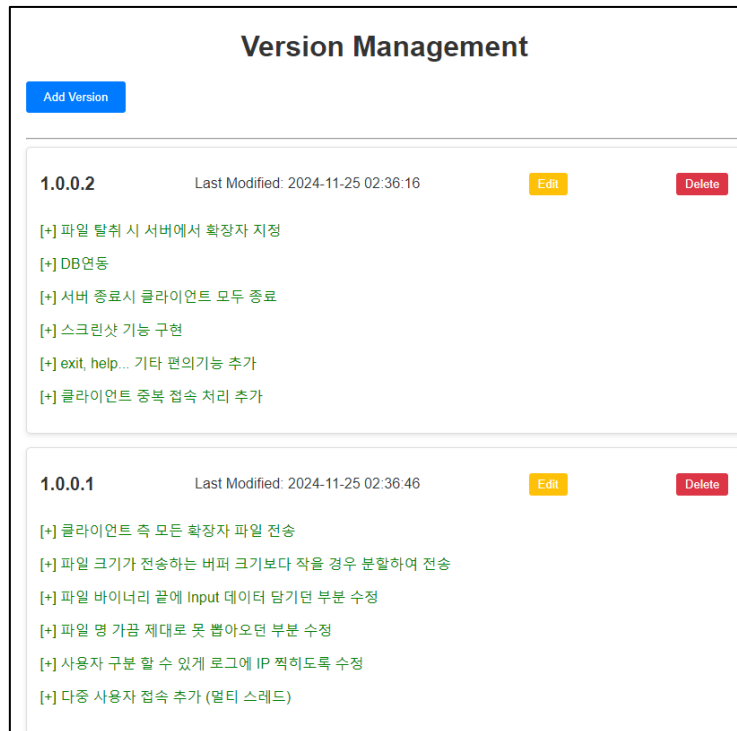
    })

    .catch(error => console.error('Error fetching logs:', error));

}

// 페이지 로드 후 로그 갱신
fetchLogs();
</script>
```

#### 4.2.10 패치노트 페이지



[그림 4-14] 패치노트 페이지

해당 페이지는 앞서 설명한 악성 파일 업로드 페이지와 더불어 악성 파일의 버전 별로 어떠한 패치가 진행 되었는지 나타내주는 페이지다.

패치노트

@app.route('/logs')

```

@login_required
def view_logs():
    return render_template('logs.html')

@app.route('/api/logs')
def get_logs():
    if os.path.exists(LOG_FILE_PATH):
        with open(LOG_FILE_PATH, 'r', encoding='utf-8', errors='ignore') as f:
            logs = f.readlines() # 최근 로그 가져오기
    else:
        logs = ["No logs available."]
    return jsonify(logs)

<script>
// 주기적으로 로그 데이터를 가져오는 함수
function fetchLogs() {
    fetch('/api/logs')
        .then(response => response.json())
        .then(data => {
            const logContainer = document.getElementById('log-container');
            const formattedLogs = data
                .filter(log => log.trim() !== '')
                .map(log => `<div class="log-entry">${log}</div>`)
                .join("");
            logContainer.innerHTML = formattedLogs;
            logContainer.scrollTop = logContainer.scrollHeight; // 항상 스크롤을 맨 아래
로 유지
        })
        .catch(error => console.error('Error fetching logs:', error));
}

// 페이지 로드 후 로그 갱신

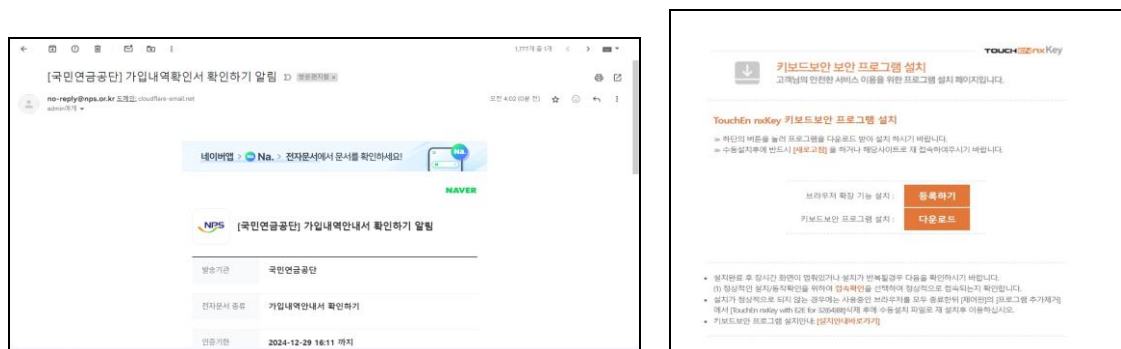
```

```
fetchLogs();
</script>
```

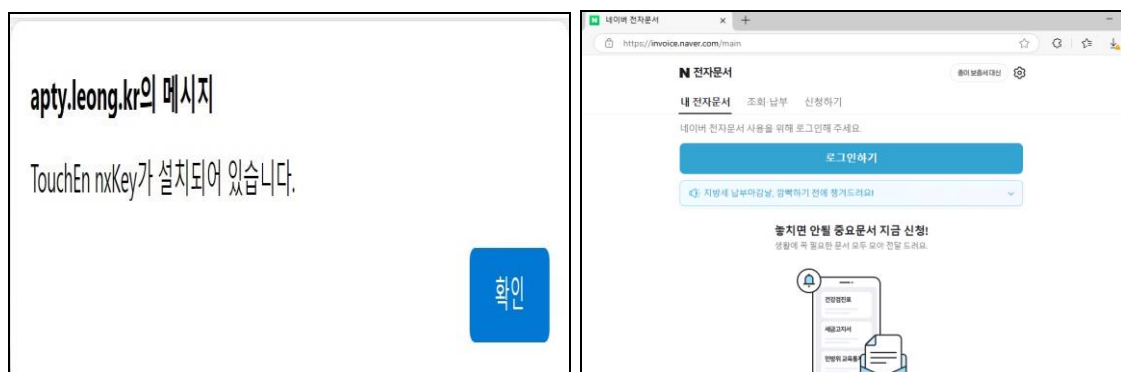
### 4.3 악성코드 세부 동작

앞서 언급했듯이, Cobalt Strike는 원래 합법적인 침투 테스트 도구로 설계되었으나, 크랙된 버전이 유출된 이후 공격자들에게 널리 악용되면서 주요 사이버 공격 도구로 변질되었다. 이는 보안 연구에서 악성 코드의 상세 코드나 기술적 세부 사항을 공개할 경우 예상치 못한 악용 사례를 초래할 수 있음을 보여준다.

본 프로젝트는 이러한 악용 가능성을 방지하고, 보안 윤리와 책임을 준수하기 위해 악성 코드의 상세 코드를 논문에 기재하지 않는다. 대신, 플랫폼 설계의 개념적 원리와 작동 방식에 중점을 두어, 프로젝트 결과가 안전하게 활용될 수 있도록 구성하였다.



[그림 4-15] 피싱 메일을 통해 피싱 사이트 접속



[그림 4-16] 악성프로그램 설치 유무 확인 후 리다이렉션



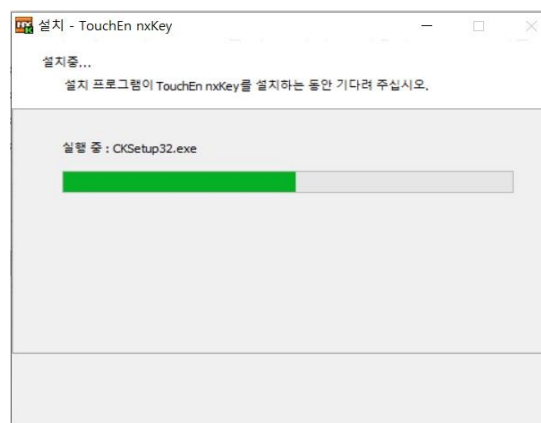
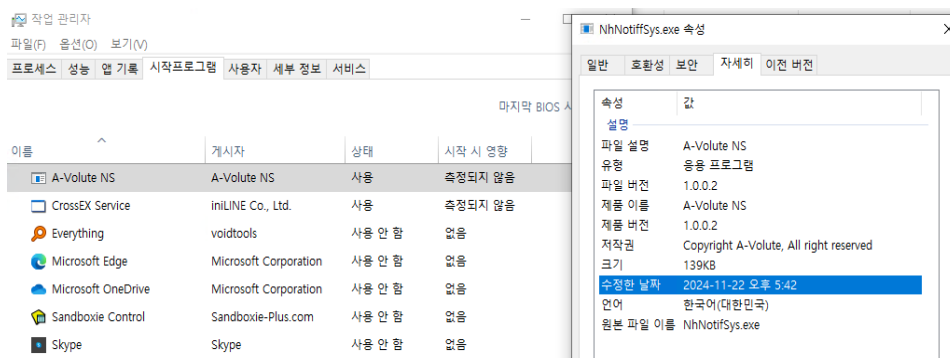
**apty.leong.kr의 메시지**

설치 확인 시간이 초과되었습니다.

확인

**[그림 4-17] 악성프로그램 설치하지 않을 경우**

공격자는 대시보드의 이메일 전송 기능을 이용하여 발신자를 변조하고 사용자에게 위조된 피싱 이메일을 발송한다. 사용자가 해당 이메일을 열람하고 피싱 사이트에 접속하면, 악성코드가 포함된 키보드 보안 프로그램 설치를 유도하는 페이지로 이동한다. 사용자가 설치를 진행하면, [그림4-2], 와 같이 피싱 사이트는 프로그램 설치 완료 여부를 확인한 후, 설치가 완료되면 정상적인 문서 확인 페이지로 이동하도록 설계되어 있으며, 설치하지 않을 경우 [그림4-3]과 같이 알림 메시지를 출력하고 다시 피싱 사이트로 재접속한다.

**[그림 4-18] 위조된 보안프로그램 설치 화면****[그림 4-19] 악성코드 지속 메커니즘 등록**

설치된 프로그램을 실행하면 사용자에게는 정상적인 보안 프로그램 설치 화면이 표시되지만, 백그라운드에서는 악성코드가 임시 폴더 경로 C:\Users\<username>\AppData\Local\Temp 에 사용자 몰래 설치된다. 동시에, 저장된 악성 로더(NhNotiffSys.exe) 파일이 악성 DLL 인젝터 (CompPkgSrv.exe), 악성 DLL (WebViewLoader.dll)를 Drop하고 [그림 4-4]와 같이 악성 로더 (NhNotiffSys.exe) 파일은 시스템의 시작 프로그램에 등록되어 지속적인 메커니즘을 수행한다.

```
sudo ./server
```



#### DB 테이블 구조

```
CREATE TABLE clients (hostip TEXT NOT NULL,hostname TEXT NOT NULL,osver TEXT NOT NULL,active BOOLEAN NOT NULL);
```

```
sqlite> PRAGMA table_info(clients);
0|hostip|TEXT|1||0
1|hostname|TEXT|1||0
2|osver|TEXT|1||0
3|active|BOOLEAN|1||0
```


[표 4-8] 악성코드 지속 메커니즘 등록

C&C(Command and Control) 서버는 클라이언트와의 연결을 대기하며, 연결이 이루어지면 DB에 사용자의 IP 주소, 호스트 네임, OS 버전, 연결 상태를 저장한다. 또한, 다양한 명령어를 통해 기능을 수행할 수 있다.

### 4.3.1 Keylogger

명령어	
키로깅 시작	<호스트 네임> keyon
키로깅 종료	<호스트 네임> keyoff

[표 4-9] 키로깅 명령어

Terminal	Client action
<호스트 네임> keyon Start Keylogging	

[표 4-10] 키로깅 시작 및 클라이언트 입력

공격자는 C&C 서버를 통해 keyon 명령어를 입력함으로써 클라이언트에서 키로깅 기능을 활성화 할 수 있다. 키로깅이 시작되면 클라이언트의 키보드 입력을 가상 키코드로 변환하여 C:\Windows\Temp 경로의 key.qccc 파일에 기록한다. 기록되는 키보드 입력은 특수 키, 특수 문자, 알파벳 등 다양한 입력 유형을 포함한다.

Terminal
<호스트 네임> keyoff Stop keylogging and receive log file. File Size received File saved Log file received Log file received:/files/keylogs/<IP>_<호스트 네임>/1.log

Result
<pre>[2024-11-19 03:50:22] ADMIN1234[shift][shift]2NAVER.COM[tab] [2024-11-19 03:50:31] PASSWORD1234 </pre>

[표 4-11] 키로깅 종료 및 C&amp;C 서버에 로그 파일 전송

keyoff 명령어를 입력하면 사용자의 키로깅이 종료되고 기록된 로그 파일은 C&C 서버로 전송된다. 전송된 파일은 서버의 <사용자 ip\_호스트 네임> 디렉토리에 저장된다. 저장된 로그 파일은 이후 대시보드에서 다운로드 할 수 있다.

#### 4.3.2 파일 탈취

명령어	
파일 탈취	<호스트 네임> steal
확장자 입력	Please enter the file extension to steal: <확장자>

[표 4-12] 파일 탈취 명령어

Terminal
<pre>&lt;호스트 네임&gt; steal Please enter the file extension to steal: pdf Steal command received for extension: pdf Receive files with extension: pdf File name received. : ./files/documents/&lt;ip&gt;_&lt;호스트 네임&gt;/사내보안교육자료.pdf File size received File data received File received : ./files/documents/&lt;ip&gt;_&lt;호스트 네임&gt;/사내보안교육자료.pdf ...</pre>
Result

```

apty@google:~/Desktop/files/documents/175. [REDACTED]_DESKTOP-L4I9B07$ ls
2023년_연말정산내역.pdf  24년10월비품구매리스트.pdf  24년9월비품구매리스트.pdf
2024년_연말정산내역.pdf  24년11월비품구매리스트.pdf  사내보안교육자료.pdf

```

[표 4-13] 파일 탈취

공격자는 steal 명령어 입력 시 목표 확장자를 설정하며, 클라이언트는 해당 명령어에 따라 시스템 내 지정된 경로에서 파일을 검색하고, 대상 파일을 식별한다. 식별된 문서 파일은 악성코드에 의해 C&C 서버로 전송된다. 전송된 파일은 서버의 <사용자 ip\_호스트 네임> 디렉토리에 저장된다. 이후 저장된 파일은 대시보드에서 다운로드 할 수 있다.

### 4.3.3 브라우저 자격증명 탈취

명령어	
자격증명 탈취 시작	<호스트 네임> credentials

[표 4-14] 브라우저 자격증명 탈취 명령어

Terminal
<호스트 네임> credentials Receive the credentials file... File name received. : ./files/credentials/<ip>_<호스트 네임>/Login Data File size received File data received File received. : ./files/credentials/<ip>_<호스트 네임>/Login Data File name received. : ./files/credentials/<ip>_<호스트 네임>/Local State File size received File data received File received. : ./files/credentials/<ip>_<호스트 네임>/Local State All files received.

Result
<pre> pty@google:~/Desktop/files 'Local State' 'Login Data' </pre>

[표 4-15] 탈취된 브라우저 자격증명 파일

Result
<pre> DESKTOP-P9PH8SS credentials Receive the credentials file... [175. ██████████ _DESKTOP-P9PH8SS] Credentials file doesn't exist </pre>

[표 4-16] 브라우저 자격증명 파일이 존재하지 않을 경우

공격자는 credentials 명령어 입력 시 Chrome Browser를 대상으로 자동 로그인 시 로컬에 저장되는 계정과 패스워드에 대한 정보가 담긴 파일인 Login Data와 암호화 키가 저장되어 있는 Local State 파일을 탈취한다. 공격자는 탈취한 파일에 저장된 데이터를 기반으로 복호화를 수행하여 사용자의 계정과 패스워드에 대한 평문 데이터를 획득할 수 있다. 사용자가 Chrome Browser를 사용하지 않는 경우에는 자격증명 탈취 명령이 실패하게 된다.

#### 4.3.4 DDoS

명령어	
DDos 명령시작	ddos

[표 4-17] DDoS 명령어

Terminal
<pre> ddos Enter domain: &lt;도메인&gt; Enter repeat count : &lt;요청할 횟수&gt; Sending DDoS command to &lt;클라이언트index&gt; Sent target domain: &lt;도메인&gt; Sent repeat count: &lt;요청할 횟수&gt; DDoS command successfully sent to all clients. </pre>

[illegible]

**[표 4-18] 공격을 통한 요청 로고**

공격자는 ddos 명령어 입력 시 타겟 도메인과 요청할 횟수를 입력받고 연결된 모든 클라이언트에게 공격 명령을 전달한다. 전달받은 클라이언트들은 입력받은 데이터를 기반으로 HTTP Flood 방식의 DDoS 공격을 타겟 웹 서비스에 수행한다. 클라이언트에서 ddos 명령은 멀티 스레드로 생성되어 작업 되기 때문에 DDoS 공격을 수행 중에도 명령을 처리할 스레드에 영향을 끼치지 않고 병렬적으로 다른 커맨드 명령을 처리 하여 작업 할 수있다.

### 4.3.5 화면 캡처

명령어	
화면 캡처 명령 시작	screenshot

**[표 4-19] 화면 캡처 명령어**

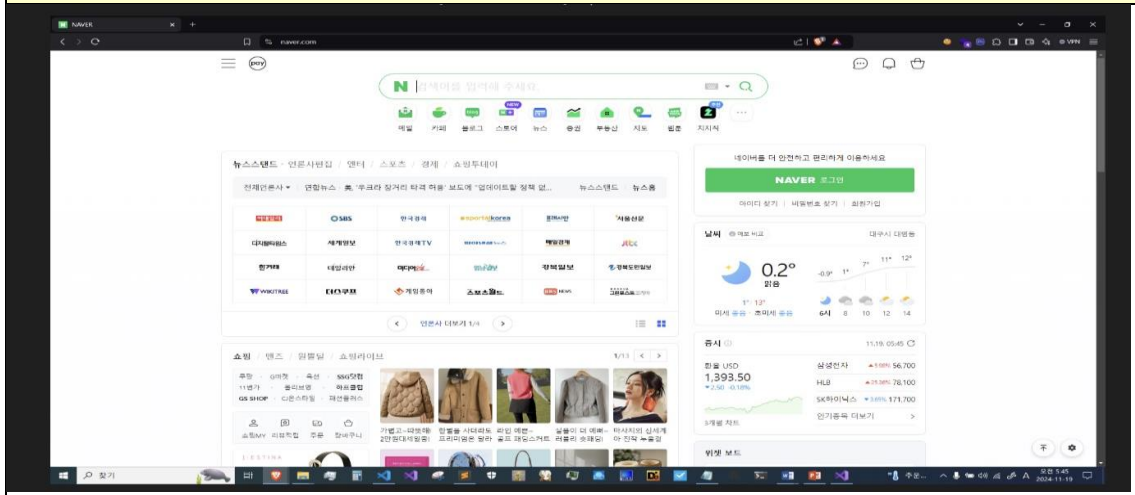
Terminal
<pre>&lt;호스트 네임&gt; screenshot Receive a screenshot file ... [&lt;ip&gt;_&lt;호스트 네임&gt;] Send the screenshot command and start receiving fileReady to receive</pre>

screenshots

Screenshot file size received: <파일크기>

Screenshot file received: ./files/screenshots/<ip>\_<호스트 네임>/screenshot\_<타임스탬프>.bmp

### Result



[표 4-20] 전송된 클라이언트 화면 캡처 파일

공격자는 screenshot 명령어 입력 시 사용자의 현재 실시간 화면을 bmp 확장자로 캡처하고 공격자 서버로 전송한다. 이때 사진 데이터는 메모리 버퍼에 저장되어 전송되고 전송 후에는 저장된 버퍼가 메모리 상에서 해제되기 때문에 스크린 샷에 대한 파일 리소스가 클라이언트 컴퓨터의 저장 공간에 남지 않는다. 전송된 사진은 C&C 서버에 저장되며 대시보드에서 열람 및 다운로드가 가능하다. 이에 따라 공격자는 감염된 사용자의 실시간 현황을 화면으로 확인할 수 있으며 화면에 기록되는 민감한 개인 정보들을 몰래 엿볼 수 있다.

### 4.3.6 연결된 클라이언트 정보 출력

명령어	
연결된 클라이언트 정보 출력	list

[표 4-21] 연결된 클라이언트 정보 출력 명령어



Result	
<pre>list [1]: DESKTOP-2AQ8LAN(175. [REDACTED]) - Windows 10 Pro 22H2 [2]: DESKTOP-QQK2VK0(175. [REDACTED]) - Windows 10 Pro 22H2 [3]: DESKTOP-P9PH8SS(175. [REDACTED]) - Windows 10 Pro 22H2 [4]: DESKTOP-JDU01B9(175. [REDACTED]) - Windows 10 Pro 22H2 Connected Clients 4</pre>	

[표 4-22] 연결된 클라이언트 정보

list 명령 시 연결된 클라이언트에 대한 호스트 네임, ip, OS 정보를 출력해준다.

#### 4.3.7 연결된 클라이언트 세션 삭제

명령어	
연결된 클라이언트 세션 삭제	<호스트 네임> delete

[표 4-23] 연결된 클라이언트 세션 삭제 명령어

Result	
<pre>DESKTOP-JDU01B9 delete Client info deleted from database: DESKTOP-JDU01B9</pre>	

[표 4-24] 연결된 클라이언트 세션 삭제

delete 명령 시 연결된 클라이언트의 세션을 끊고, 데이터베이스에 저장된 클라이언트 정보들도 함께 삭제한다.

#### 4.3.8 서버 종료

명령어	
서버 종료	exit
서버 종료 유무	y / n

[표 4-25] 서버 종료 명령어

Result
<pre>exit Are you sure you want to shut down the server? [y/n] : y  BYE  goodbye~!</pre>

[표 4-26] exit (y 입력시)

Result
<pre>exit Are you sure you want to shut down the server? [y/n] : n Does not shut down the server.</pre>

[표 4-27] exit (n 입력시)

exit 명령 시 서버 종료에 대해 확인하는 문구가 나오며 "y" 입력 시 C&C 서버를 종료하며, "n" 입력 시 서버를 종료하지 않는다는 문구가 나온다.

## 4.3.9 도움말 기능

명령어	
도움말	help

[표 4-28] 도움말 명령어

Result
<pre> help -----       help      : Command Help       list      : Check connected clients       exit      : Shutdown server       ddos      : Start DDoS       keyon     : Start keylogging       keyoff    : Stop keylogging and send log file       steal     : Files steal       credentials : Steal browser credentials       screenshot : Get screenshot       delete    : Delete client info (session) ----- </pre>

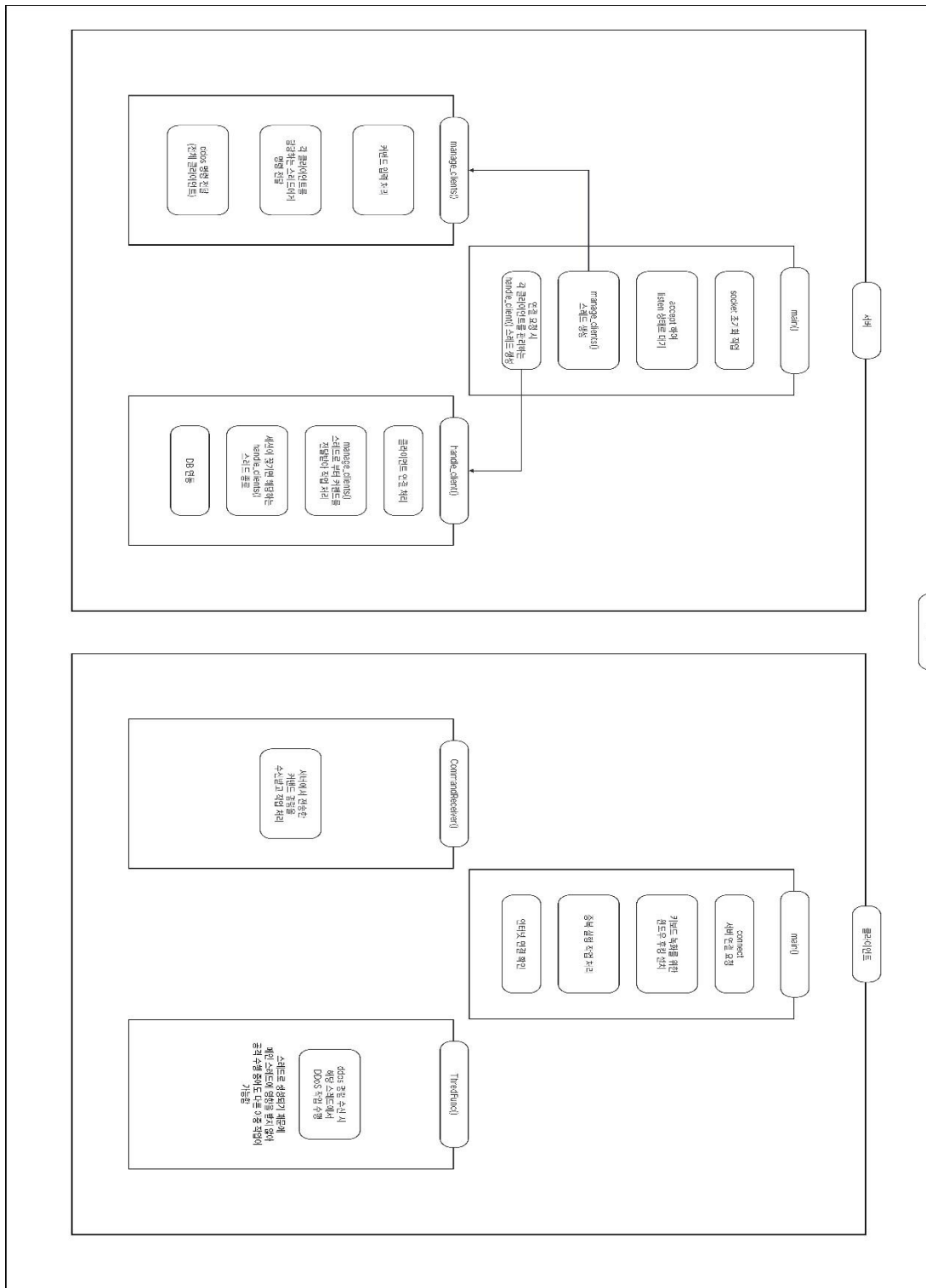
[표 4-29] 도움말

help 명령 시 공격 및 추가적인 명령어에 대한 상세 내용을 출력한다.

상세 내용은 다음과 같다.

- help: 도움말 출력
- list: 연결된 클라이언트 정보 출력
- delete: 연결된 클라이언트 세션 삭제
- exit: 서버를 종료.
- ddos: 모든 클라이언트에게 DDoS 명령 전달
- keyon: 키로깅 시작 명령 전달
- keyoff: 키로깅 중지 및 로그파일 전송 명령 전달
- steal: 파일 탈취 명령 전달
- credentials: 브라우저 자격증명 파일 탈취 명령 전달
- screenshot: 화면 캡처 및 파일 전송 명령 전달

#### 4.3.10 스레드 구조



[그림 4-20] 스레드 구조 구성도

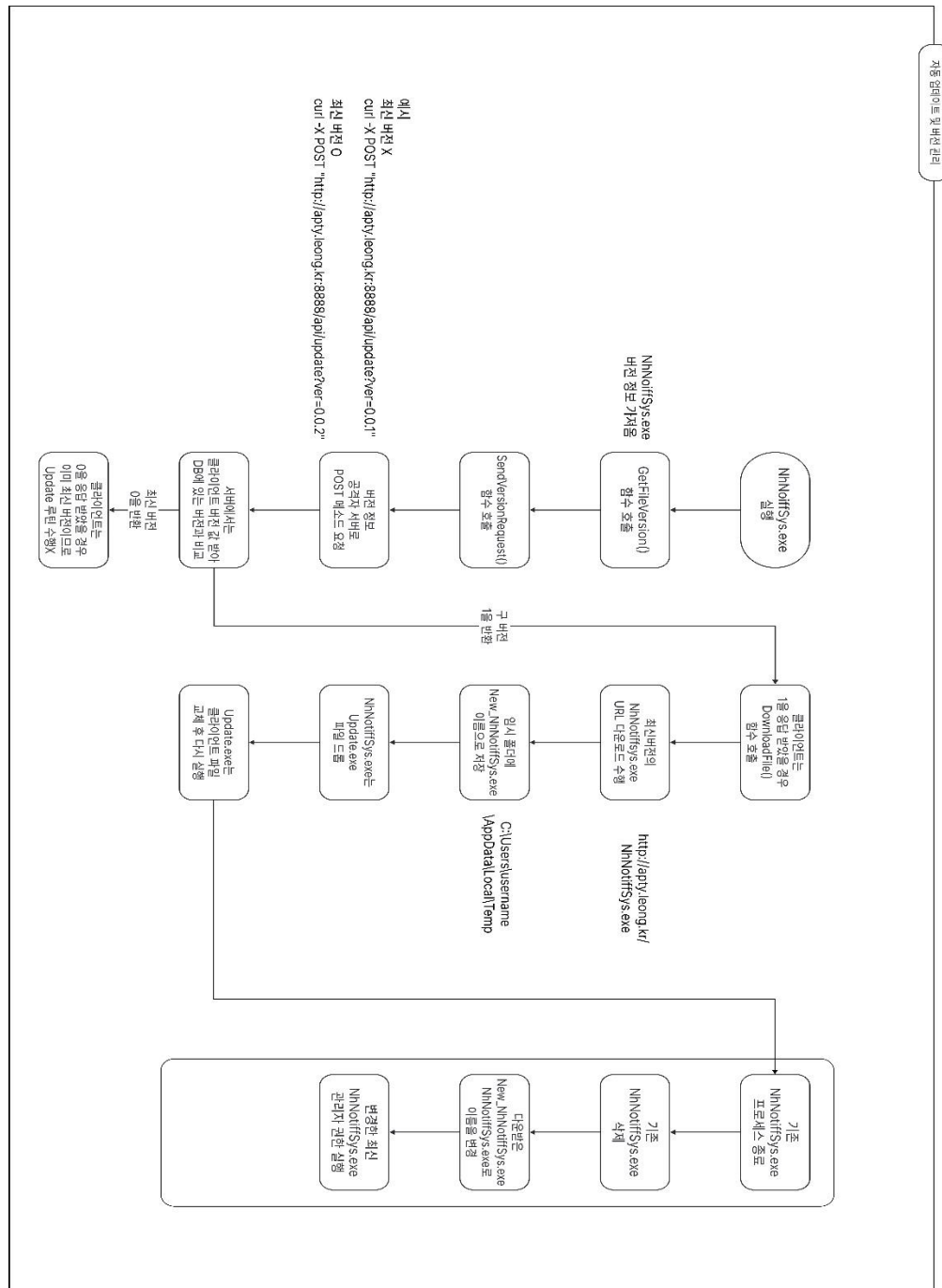
서버 구성 요소 표		
구성요소	설명	주요 작업
main()	서버의 핵심 진입점	소켓 초기화 및 설정 클라이언트 연결 대기 (accept 및 listen 상태) manage_clients()관리 스레드 생성
manage_clients()	클라이언트 관리 및 명령 전달을 처리하는 스레드	커맨드 입력 처리 특정 클라이언트에 명령 전달 ddos 명령 수행시 전체 클라이언트 대상으로 명령 실행
handle_client()	각 클라이언트와 통신을 담당하는 독립적인 스레드	클라이언트 연결 요청 처리 DB 와 연동하여 클라이언트 정보관리 manage_clients()스레드로부터 명령 전달 및 작업 수행 클라이언트 연결 종료 시 스레드 종료

클라이언트 구성 요소 표		
구성요소	설명	주요 작업
main()	클라이언트의 초기 설정 및 서버 연결 작업	서버 연결 요청(connect) 키보드 녹화를 위한 윈도우 후킹 설치 중복 실행 방지 처리 인터넷 연결 확인
commandReceiver()	서버에서 전송된 명령을 수신 및 실행	명령 수신 명령 실행 결과를 서버로 반환
threadFunc()	ddos 명령 수신 시 스레드를 통해 작업 처리	ddos 명령 수신 시 스레드 생성 병렬 작업 수행

[표 4-30] 스레드 구성 요소

클라이언트마다 독립적인 스레드를 생성하여 명령을 처리함으로써 다수의 클라이언트를 동시에 안정적으로 관리할 수 있으며, 특정 클라이언트에 명령을 전달하거나 전체 클라이언트를 대상으로 명령을 실행할 수 있는 유연성을 제공한다.

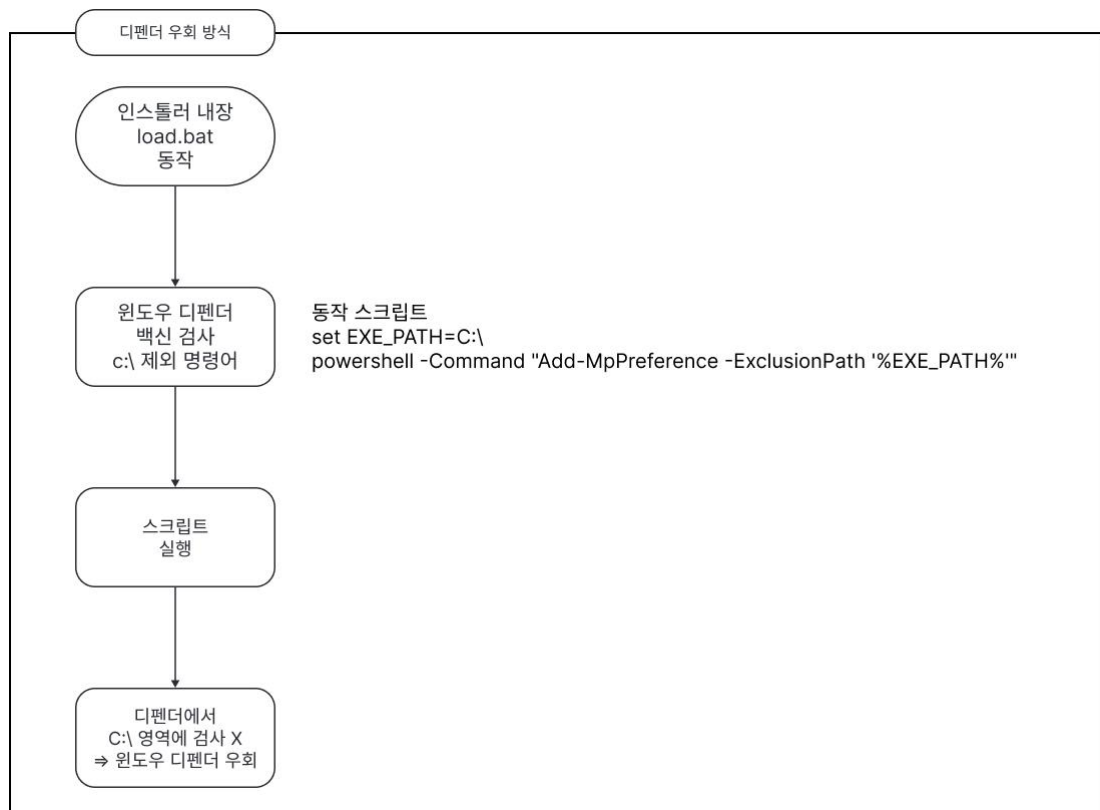
#### 4.3.11 자동 업데이트 및 버전 관리



[그림 4-21] 자동 업데이트 및 버전 관리 구성도

악성 로더(NhNotiffSys.exe)가 실행되면 GetfileVersion()를 호출하여 프로그램의 리소스 영역에서 버전 정보를 가져온다. 이후 SendVersionRequest()를 호출하여 POST 메소드로 공격자 서버에 클라이언트의 버전 정보를 전송한다. 서버에서는 version.db에 있는 버전 정보 데이터와 클라이언트가 전송한 버전 정보를 비교하여 같을 경우 0을 반환하고 다를 경우 1을 반환한다. 클라이언트는 0을 응답받았을 경우 이미 최신 버전이기 때문에 Update루틴을 수행하지 않고 기존 구문으로 넘어간다. 1을 응답받았을 경우는 DownloadFile()를 호출하여 공격자 서버에 있는 최신 악성 로더를 다운로드 하고 임시 폴더 경로에 New\_NhNotiffSys.exe로 저장한다. 이후 최신 버전으로 교체 작업을 하기위한 Update.exe 파일을 Drop하고 실행한다. Update.exe는 구버전 클라이언트와 신버전 클라이언트의 파일을 교체하고 다시 실행하는 작업을 수행하기 위해서 기존 악성 로더를 종료하고 파일을 삭제한다. 이후 New\_NhNotiffSys.exe를 NhNotiffSys.exe 이름으로 교체하고 다시 실행하여 공격자는 원할 때 감염자가 최신 버전의 악성 로더를 유지할 수 있도록 설계되어 있다.

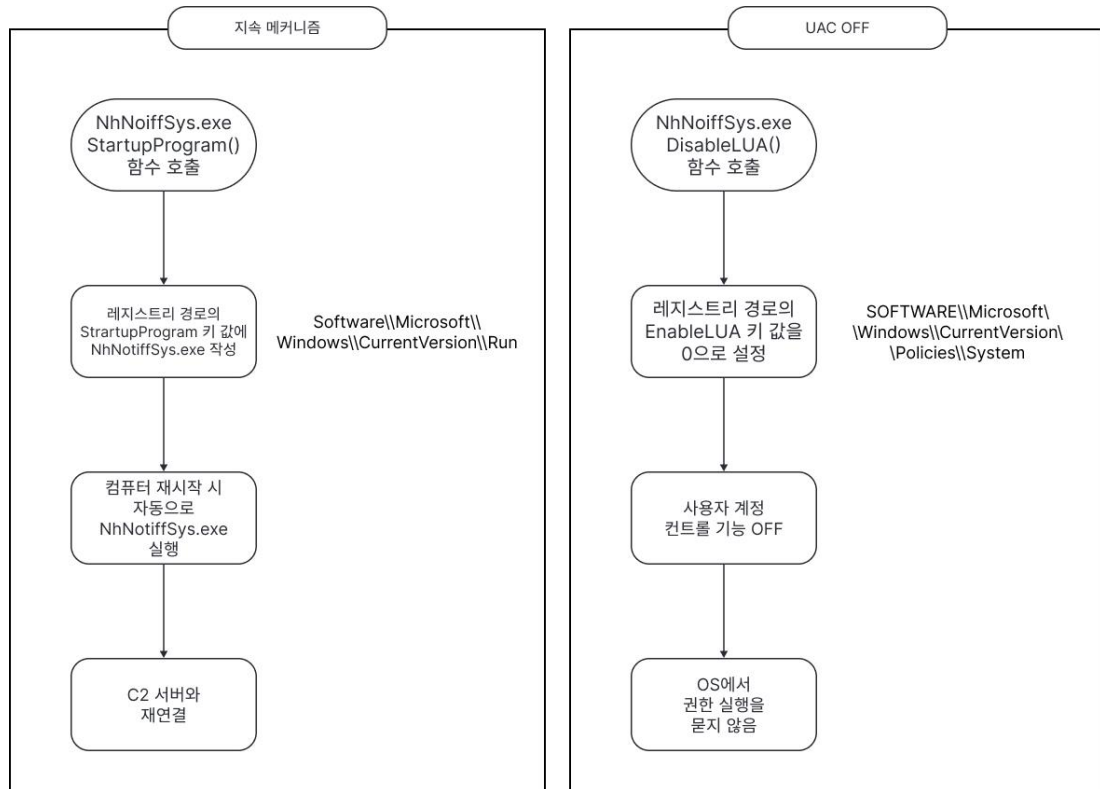
#### 4.3.12 Defender 설정 조작



[그림 4-22] Defender 우회 구성도

위조된 보안 설치 프로그램이 실행되면 내장되어 있는 load.bat(배치 스크립트)에서 윈도우 디펜더 백신 검사를 우회하기 위한 스크립트가 실행된다. C:\w 영역 구간은 백신 검사에서 제외될 수 있도록 파워 셸 명령을 수행하여 악성 로더와 악성 DLL 파일에 대한 백신 검사를 제외시킨다.

#### 4.3.13 지속 메커니즘 및 사용자 계정 컨트롤



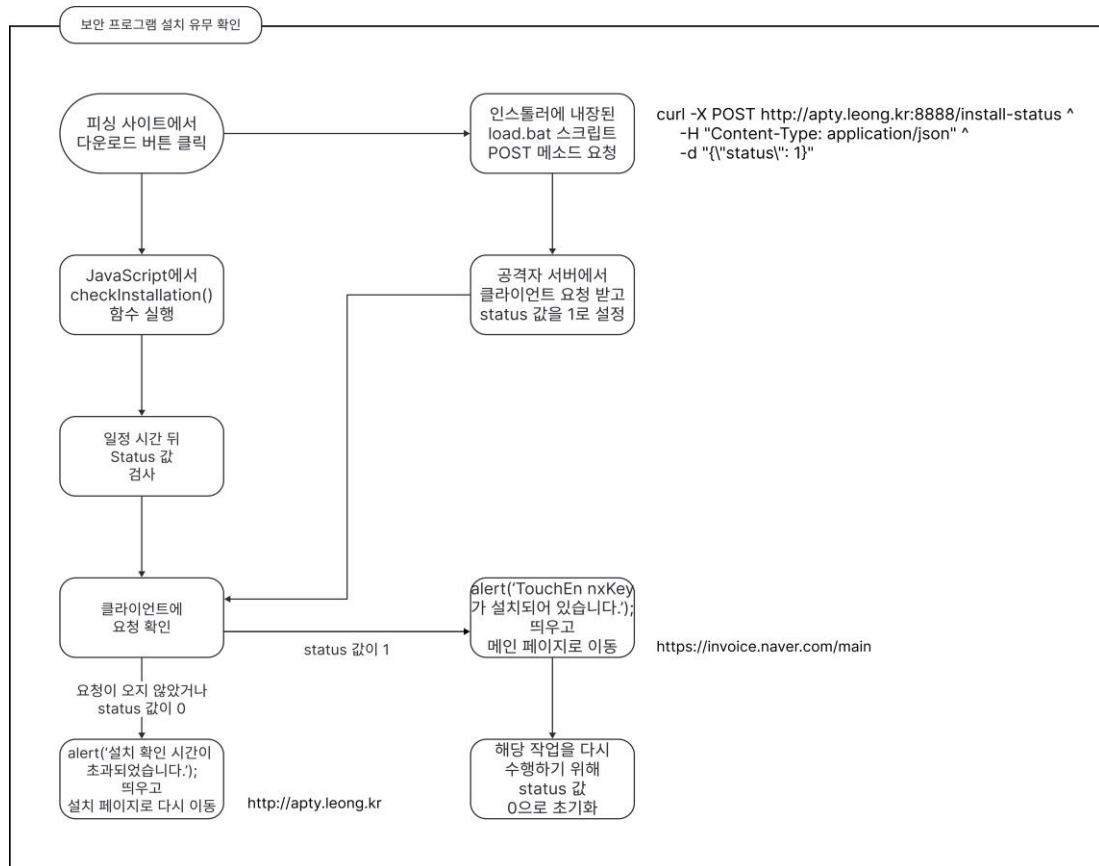
[그림 4-23] 지속 메커니즘 및 UAC 구성도

악성 로더 (NhNotiffSys.exe)는 공격자 서버와 접속을 계속 유지하기 위해 지속 메커니즘에 등록된다. 악성 로더는 StartupProgram()를 호출하여 레지스트리 경로의 StartupProgram 키 값에 NhNotiffSys.exe를 작성한다. 이로 인해 로더는 컴퓨터가 재시작되어도 악성 로더가 실행되기 때문에 지속적으로 공격자 서버와 세션을 유지할 수 있다.

또한 레지스트리 EnableLUA 키 값을 0으로 설정하여 관리자 권한으로 실행될 때 OS에서 경고 팝업 창을 띄우지 않도록 사용자 계정 컨트롤 기능을 강제로 수정한다.



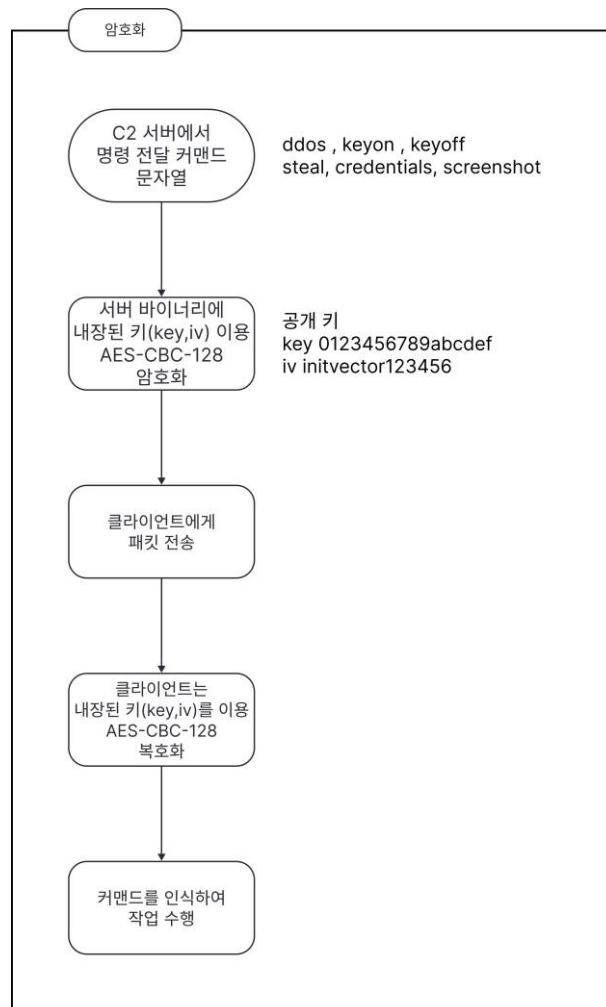
## 4.3.14 키보드 보안 프로그램 설치 유무



[그림 4-24] 피싱 사이트 리다이렉션 구성도

본 구성도는 피싱 사이트에서 악성파일 설치 유무를 판단하여 정상 페이지로 리다이렉션 하는 동작 원리를 나타낸다. 피싱 사이트에서 정상 파일인 TouchEn\_nxKey 로 위장한 악성파일을 설치하기 위해 다운로드 버튼을 누르게 되면 JavaScript 에서 checkinstallaion()를 실행하며, checkinstallation()를 통해 일정 시간 뒤에 status 값을 검사하여 클라이언트에게 요청 수신 유무를 검사한다. 악성파일을 설치하면 정상 파일에 숨겨진 악성 스크립트(load.bat)가 실행되면서 C&C 서버 POST 요청을 전송한다. 이에 C&C 서버는 클라이언트의 요청을 받아 status 값을 1로 설정하고, "TouchEn\_nxKey 가 설치되어 있습니다." 라는 알림 메시지를 띄운 뒤 정상페이지로 이동한다. 요청이 오지 않았거나 status 값이 0 이라면 "설치 확인 시간이 초과되었습니다."라는 알림 메시지를 띄우고 다시 설치페이지로 이동하게 된다. 이 모든 작업이 끝나게 되면 재화성화를 위해 status 값을 0으로 초기화한다.

## 4.3.15 암호화



[그림 4-25] 암호화 구성도

본 구성도는 C&C 서버와 클라이언트 간의 명령 전달 및 실행 과정을 AES-CBC-128 암호화 알고리즘을 활용하여 구현한 시스템 동작 원리를 나타낸다. C&C 서버는 커맨드 문자열을 서버 바이너리에 하드 코딩 되어있는 키(key)와 벡터(iv)를 사용하여 AES-CBC-128 암호화 알고리즘을 적용한다.

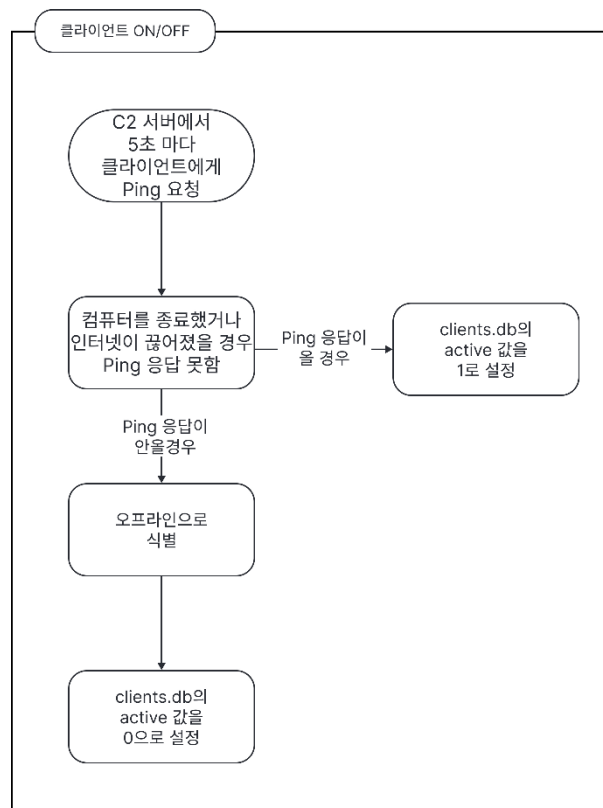
암호화되는 커맨드 명령 문자는 다음과 같다.

- ddos
- keyon
- keyoff
- steal

- credentials
- screenshot

C&C 서버는 암호화된 명령어를 클라이언트로 전송하고 클라이언트는 수신된 패킷을 처리하여 서버와 동일하게 저장된 key 와 iv 값을 사용하여 AES-CBC-128 복호화를 수행한다. 복호화된 명령을 인식하고 해당 명령에 따라 작업을 수행한다. 본 시스템은 악성 코드의 동작 원리를 학습하고, 보안 취약점을 분석하기 위한 연구 환경으로 설계되었다. AES-CBC-128 암호화 알고리즘을 활용하여 명령 전달 과정의 보안성을 강화하였으며, 이를 통해 실제 악성 코드와 유사한 환경을 제공한다.

#### 4.3.16 클라이언트 ON/OFF

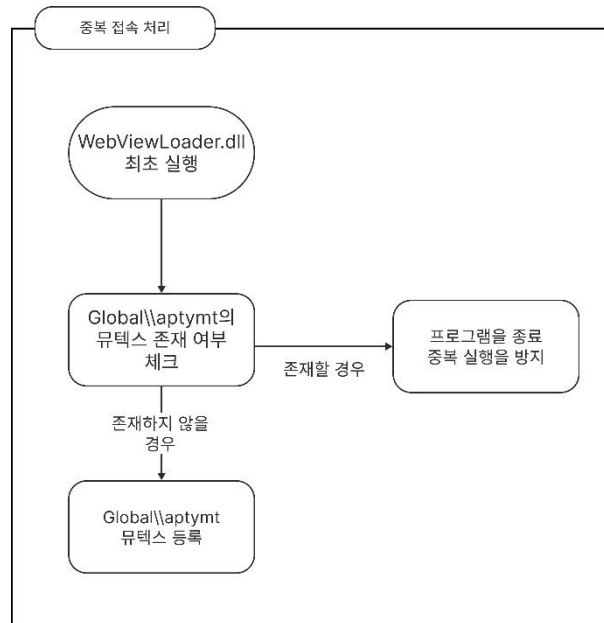


[그림 4-26] 클라이언트 ON/OFF 구성도

C&C 서버는 일정 간격마다 클라이언트에게 ping 요청을 전송하여 네트워크가 연결되어 있는지 확인한다. ping 요청에 대한 응답이 올 경우 클라이언트의 네트워크가 연결되었다고 판단하여 C&C 서버는 clients.db의 active 값을 1로 설정하여 클라이언트가 활성화 상태임을

나타낸다. 만약 ping 요청에 응답이 오지 않았을 경우, 네트워크가 연결되어 있지 않거나, 컴퓨터가 종료되었다고 판단하여 C&C 서버는 clients.db 의 active 값을 0 으로 설정하여 클라이언트가 비활성화 상태임을 나타낸다.

#### 4.3.17 클라이언트 중복 접속 처리



[그림 4-27] 클라이언트 중복 접속 처리 구성도

본 구성도는 C&C 서버와 클라이언트 간의 중복접속 여부 판단 및 중복접속 방지에 대한 동작 원리를 나타낸다. 시스템이 처음 시작할 때 WebViewLoader.dll 파일이 최초로 실행되며, 시스템은 Global\\wptymt 라는 이름의 뮤텍스가 존재하는지 확인한다. 만약 뮤텍스가 존재한다면 동일 프로세스의 인스턴스가 실행되고 있음을 의미하며, 새로운 인스턴스는 실행이 종료되며, 뮤텍스가 존재하지 않을 경우 뮤텍스를 새로 생성하여 프로세스의 고유한 뮤텍스로 등록한다. C&C 서버와 클라이언트 간의 중복접속 여부를 판단 및 방지하였다.

## 5. 프로젝트 결론

본 프로젝트는 피싱 이메일을 통해 악성코드 설치를 유도하고 피해자와 공격자 간 상호작용을 기반으로 하는 모의 악성코드 감염 플랫폼을 개발하였다. 프로젝트의 핵심은 실질적인 악성코드 동작 시나리오를 구현하고 실제 악성코드의 기능을 정교하게 재현함으로써 보안 인력들과 악성코드 동작 및 분석에 대한 교육이 필요한 대상으로 이를 학습하고 분석할 수 있는 환경을 제공하는 데 있다.

본 프로젝트는 다음과 같은 장점을 제공한다.

- (1) 현실적인 피싱 이메일과 악성코드의 동작 방식을 학습할 수 있는 시뮬레이션 환경을 제공함으로써 악성코드의 세부적인 동작 과정에 대한 분석 및 대응 전략 수립에 기여한다.
- (2) Windows와 Linux 환경 간의 상호작용을 통해 다양한 운영 체제의 보안 메커니즘을 실습하고 연구할 수 있도록 한다.
- (3) 공격자 대시보드에서 데이터 흐름을 실시간으로 확인 할 수 있는 기능을 제공하는 인터페이스를 구현하였다.

그러나 본 프로젝트는 몇 가지 한계를 가진다. 현재 플랫폼은 Windows의 사용자 모드에서 동작하는 악성코드에 중점을 두고 있어, 시스템의 더 깊은 레벨에서 이루어지는 고급 공격 시나리오를 다루는 내용은 없기에 아쉬움이 있다. 또한, 다양한 기능을 포함한 악성코드 테스트 과정에서 예상치 못한 동작이 발생할 가능성이 있으며, 실제 환경에서의 실행 시 발생할 수 있는 법적·윤리적 이슈 또한 충분히 고려해야 한다. 대규모 데이터를 처리하거나 구현하는 데 있어 성능 최적화 역시 해결해야 할 과제로 남아 있다.

추후에는 Windows 내부 구조에 대한 심화 연구를 통해 고급 공격 시나리오를 구현하고, 루트킷이나 커널(Ring 0)을 참조하는 악성코드 및 시스템 취약점에 대한 연구도 진행할 계획이다. 이를 통해 시스템의 더 깊은 레벨에서 이루어지는 악성코드 동작과 취약점 악용 방식을 이해하고, 이에 대한 대응책을 마련하는 데 기여하는 것을 목표 과제로 삼을 것이다.

본 프로젝트는 보안에 관심이 있는 학생부터 전문 보안 인력에 이르기까지 악성코드의 동작 원리를 학습하고 직접 실습해 보면서 대응 방안을 연구하는 데 있어 유용한 도구로 활용될 수 있으며, 다양한 악성코드 동작 시나리오를 추가하여, 보안 연구와 방어 전략 개발에 기여할 수 있는 확장성 높은 플랫폼으로 활용될 수 있을 것으로 기대된다.

## 참고 문헌

- [1] <https://boannews.com/media/view.asp?idx=133525&page=1&kind=1>  
(텔레그램으로 유통되는 악성코드)
- [2] <https://en.wikipedia.org/wiki/NjRAT>  
(njRAT)
- [3] <https://asec.ahnlab.com/en/47283>  
(quasar RAT)
- [4] <https://blogs.blackberry.com/en/2016/04/the-ghost-dragon>  
(Gh0st RAT)
- [5] <https://www.cobaltstrike.com/>  
(Cobalt Strike)
- [6] <https://redops.at/en/blog/direct-syscalls-a-journey-from-high-to-low>  
(Direct System Call)
- [7] <https://www.igloo.co.kr/security-information/악성코드로-알아보는-reflective-dll-injection/>  
(DLL Manual Mapping)
- [8] <https://www.elastic.co/kr/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process>  
(Process Hollowing)

## Abstract


This project focuses on developing a simulated malware infection platform designed to replicate real-world attack scenarios for security research and training purposes. The platform employs phishing emails as the initial attack vector to deliver and execute malware on a victim's Windows environment, with the attacker operating from a Linux-based control system. The core of the system is a Remote Access Trojan (RAT) implemented in C/C++ programming languages, offering functionalities such as keylogging, screenshot capture, file theft, credential extraction, and DDoS attacks.

A dashboard-based Command and Control (C&C) server enables centralized management, allowing administrators to monitor sessions, manage files, and execute commands in real-time. The platform simulates realistic malware behaviors, providing researchers and analysts with an environment to study the impact of malware infections and develop defense mechanisms.

By integrating various attack techniques into a single platform, the system offers a hands-on learning experience for both beginners and advanced researchers. While the platform is a powerful tool for understanding malware operations, ethical considerations and legal constraints are emphasized to ensure responsible use.

The developed platform is expected to contribute to security education and research by enabling users to analyze malware behavior, refine incident response strategies, and ultimately strengthen cyber security defenses.

## 팀원 소개

	<p>이 름: 전 동 혁</p> <p>전 공: 사이버보안과</p> <p>학 번: 2237021</p> <p>졸업 프로젝트 역할: 프로젝트 팀장 및 총괄, 논문 제작, 발표 자료 제작, 주요 시스템 설계 및 개발</p>
	<p>이 름: 최 우 영</p> <p>전 공: 사이버보안과</p> <p>학 번: 2237025</p> <p>졸업 프로젝트 역할: 웹 대시보드 제작(프론트/백엔드), 환경구축, 피싱 사이트 제작</p>
	<p>이 름: 강 세 현</p> <p>전 공: 사이버보안과</p> <p>학 번: 2237027</p> <p>졸업 프로젝트 역할: 악성코드 기능 구현, 발표 자료 제작, 논문 제작</p>
	<p>이 름: 서 혁 준</p> <p>전 공: 사이버보안과</p> <p>학 번: 2237037</p> <p>졸업 프로젝트 역할: 악성코드 기능 구현, 논문 제작, 판넬 제작, 발표 자료 제작, 자료수집, 이미지 제작, 발표</p>