# 마비노기 모바일 매크로 프로그램 탐지 우회 및 제작 계획 보고서

## 개요

이 보고서는 마비노기 모바일(Unity 기반 윈도우 프로그램)의 생활 콘텐츠 자동 클릭 매크로 프로그램 개발을 목표로, 넥슨의 매크로 탐지 시스템을 효과적으로 우회하기 위한 고급 전략과 기술적 고려사항을 제시한다. 사용자의 초기 제안(클릭 범위 중앙의 랜덤 오프셋, 각 클릭별 최소 200ms 이상의 랜덤 딜레이)을 바탕으로, 현대적인 게임 보안 시스템의 복잡성을 분석하고 이를 회피하기 위한 다층적인 접근 방식을 제안한다. 또한, 프로그램 개발 언어로서 C#과 Python의 적합성을 비교하고 최적의 개발 계획을 수립한다.

넥슨 인텔리전스랩스는 "사용자가 최적의 환경에서 더 재미있게 게임을 즐길 수 있도록" 연구하며, 게임 내외의 이상 상황을 빠르고 정확하게 탐지하여 조치하는 데 중점을 둔다.[1] 이는 단순한 매크로 방어를 넘어, 데이터 기반의 정교한 탐지 모델을 활용하고 있음을 시사한다.[1] 따라서 매크로 프로그램은 단순히 기존의 탐지 방식을 우회하는 것을 넘어, 인간의 행동과 구별하기 어려운 수준의 정교함을 갖춰야 한다.

## 넥슨의 안티 치트 생태계**: NGS/**블랙 사이퍼 이해

### 넥슨의 안티 치트 철학과 발전

넥슨 인텔리전스랩스는 게임 내 이상 상황을 신속하고 정확하게 탐지하여 사용자가 쾌적한 환경에서 플레이할 수 있도록 지원하는 데 주력한다.[1] 이들은 탐지된 어뷰저에 대한 제재 근거가 될 수 있는 데이터와 분석 결과를 제공하며, 이는 실시간 모니터링과 게임 내 신고 기능, 전용 창구를 통해 접수된 내용을 확인하여 이용을 제재하는 방식으로

이어진다.[1] 넥슨은 비인가 프로그램 사용 등 보안 이슈를 매우 중요하게 생각하고 강력하게 대응한다고 밝히며, AI를 활용한 시스템 구축을 통해 개인정보 도용 피해 건수와 금액을 90% 이상 줄이는 극적인 성과를 거두기도 했다.[1]

이러한 배경은 사용자의 초기 인식이 단순히 프로세스 이름 차단에 기반한 넥슨의 구식 안티 치트 시스템에 대한 것임을 보여준다. 넥슨의 인텔리전스랩스는 데이터 기반, AI/ML 기반의 "이상 상황" 및 "비정상 플레이" 탐지에 명확히 초점을 맞추고 있다.[3] 남윤우 연구원이 언급한 "신호 처리를 이용한 반복 매크로 탐지"는 단순한 정적 시그니처 검사를 넘어선 정교한 패턴 기반 접근 방식을 사용하고 있음을 확인시켜 준다.[1] 이는 안티 치트 개발이 "끝없는 전쟁"임을 의미하며, 어떤 정적 우회 기술도 결국 실패할 수 있음을 나타낸다.[1] 따라서 성공적인 매크로는 정적 시그니처를 피하는 것뿐만 아니라, 고급 통계 및 AI 기반 이상 탐지를 회피할 수 있도록 인간의 행동 패턴을 효과적으로 모방해야 한다. 이는 안티 치트가 지속적으로 학습하고 진화하고 있음을 의미하며, 매크로 개발 전략 또한 이에 맞춰 적응해야 한다.

**NGS/**블랙 사이퍼**:** 아키텍처 및 알려진 기능

넥슨 게임 시큐리티(NGS)는 블랙 사이퍼(BlackCipher)로도 알려진 넥슨의 주요 안티 치트 시스템이다.[6] 블랙 사이퍼는 커널 수준에서 작동하며 [7], 이는 운영 체제에 대한 깊은 접근 권한과 제어권을 부여하여 PC의 거의 모든 활동을 모니터링할 수 있게 한다.

블랙 사이퍼는 매우 침투적이며, 열린 프로그램, 방문한 웹사이트(HTTP 헤더 및 타임스탬프 포함), IP 주소, 워크그룹, 윈도우 버전, 비밀번호, 네트워크 자격 증명, 하드웨어, 호스트, 라이브러리, 현재 브라우저 탭, 윈도우 대화 상자, 파일 등 광범위한 사용자 데이터를 로깅할 수 있다.[9] 이러한 수준의 데이터 수집은 사용자 개인 정보 보호에 대한 심각한 우려를 제기한다. 블랙 사이퍼는 승인되지 않은 프로그램을 탐지하면 게임을 종료시키고 [11], 사용자가 프로세스를 종료하려고 하면 "파일 변조"로 계정을 플래그할 수 있다.[10]

이러한 정보는 사용자의 프로세스 이름 차단에 대한 가정이 너무 협소하다는 것을 보여준다. 블랙 사이퍼의 커널 수준 접근 권한은 모든 실행 중인 프로세스와 그 활동을 감시할 수 있음을 의미한다.[9] 이는 단순히 알려진 실행 파일 이름을 차단하는 것을 넘어, 사용자 행동과 시스템 상태에 대한 포괄적인 프로필을 구축하는 것이다. 만약 "모든 열린 프로그램" [9]과 "현재 브라우저 탭" [12]을 로깅한다면, 이는 게임 관련 프로세스 외에도 훨씬 더 광범위하게 모니터링하고 있음을 의미한다. 따라서 블랙 사이퍼를 우회하려면 매크로 실행 파일의 이름을 바꾸는 것을 넘어, 매크로의 흔적을 최소화하고, 의심스러운 시스템 상호 작용을 피하며, 저수준에서 그 존재와 행동을 난독화하는 전략이 필요하다. 수집된

데이터는 실시간 탐지를 우회하더라도 사후 행동 분석에 사용될 수 있다.

## 현대 안티 치트가 사용하는 일반적인 탐지 벡터

현대적인 안티 치트 솔루션은 시그니처 탐지, 휴리스틱 분석, 행동 분석, 메모리 스캐닝, 서버 측 유효성 검사 등 다양한 기술을 조합하여 사용한다.[16]

- 시그니처 기반 탐지**:** 알려진 치트 파일 시그니처 또는 코드 패턴을 스캔한다.[16] 이는 치트를 재컴파일하거나 수정하면 쉽게 우회될 수 있다.[16]
- 휴리스틱 분석**:** 플레이어의 행동에서 치트를 나타낼 수 있는 의심스러운 패턴을 모니터링한다.[16] 여기에는 이동 속도, 점프 높이, 발사 속도, 그리고 전반적인 "비인간적 패턴" 분석이 포함된다.[17]
- 메모리 스캐닝**:** 게임 코드 또는 값에 대한 무단 변경을 찾는다.[23] 이는 직접적인 메모리 조작을 탐지하는 일반적인 방법이다.
- 행동 모니터링**(**런타임**):** 프로세스 생성, 파일 시스템 작업, 레지스트리 수정, 네트워크 통신 및 메모리 조작을 관찰한다.[22] 이는 휴리스틱 분석의 고급 형태이다.
- 머신러닝**/AI** 탐지**:** 방대한 플레이어 행동 데이터셋을 학습하여 인간 분석으로는 놓칠 수 있는 미묘한 패턴과 이상 징후를 식별한다.[17]
- 서버 측 유효성 검사**:** 클라이언트가 보낸 행동이 합법적인 게임 규칙 내에서 타당한지 확인한다. 이는 리소스 기반 치트 방지에는 효과적이지만, 맵 핵과 같은 "지식 기반" 치트에는 덜 효과적이다.[16]
- 네트워크 모니터링**:** 비정상적인 트래픽 급증, 무단 접근 시도, 잠재적인 패킷 주입을 탐지할 수 있다.[30]

사용자가 "패킷 캡처 프로그램을 활용한 프로그램도 나왔으니까 네트워크를 따로 모니터링하고 있지는 않을 거야"라고 언급한 것은 잘못된 가정이다. 클라이언트 측 패킷 캡처 도구의 존재가 서버 또는 커널 수준 안티 치트가 게임 클라이언트의 네트워크 활동을 모니터링하는 것을 배제하지 않는다. 실제로 많은 안티 치트 시스템은 네트워크 분석을 사용하여 봇을 탐지한다.[33] "딜미터기"가 프로세스 이름으로 차단되었다는 사용자의 주장은 과거의 단순한 접근 방식을 시사하지만, 현재는 복잡한 행동 및 AI 기반 탐지로 전환되었음을 명확히 보여준다.[17] 이는 정적 방법이 쉽게 우회되는 "군비 경쟁"이다.[16] 따라서 강력한 매크로는 단일 탐지 계층뿐만 아니라 여러 계층의 탐지를 회피해야 한다. 입력 무작위화에만 의존하는 것은 불충분하며, 프로세스, 메모리, 네트워크 흔적, 그리고 시간 경과에 따른 전반적인 행동 패턴까지 고려해야 한다.

넥슨의 정책 및 제재 이력

넥슨의 운영 정책은 "매크로 프로그램" 및 기타 비인가 소프트웨어 사용을 명시적으로 금지하며, 위반 시 "영구 게임 이용 제한"을 부과한다.[41] 이는 서류상으로는 엄격한 정책이다.

그러나 마비노기 자체의 과거 관찰에 따르면, 스킬 훈련을 위한 AFK 매크로를 사용하는 플레이어들이 의미 있는 처벌을 받지 않거나, 공개적인 사용을 피하라는 경고만 받는 등 관용적인 태도를 보인 사례가 있다.[42] 이는 공식 정책과 모순된다. 넥슨은 과거 스팸 봇을 대상으로 한 "시한폭탄 퀴즈"를 발행하기도 했는데, 이는 매크로 방지 챌린지의 한 형태일 수 있다.[42] 원격 프로그램(예: Sunshine/Parsec) 사용은 다른 넥슨 게임에서 제재 파동과 연관되어 있었으며 [43], 이는 합법적인 원격 데스크톱 도구도 게임 프로세스나 입력에 간섭하면 안티 치트를 유발할 수 있음을 시사한다. 넥슨의 고객 지원은 종종 도움이 되지 않는다고 묘사되며, 안티 치트 제재 항소는 자동 거부되어 입증 책임이 사용자에게 전가되는 경우가 많다.[30] 이는 오탐지(False Positive)의 위험이 크고 구제 수단이 거의 없음을 의미한다.

넥슨의 공식 정책은 엄격하지만 [41], 마비노기 커뮤니티의 보고서 [42]는 AFK 매크로에 대한 역사적인 관용을 시사한다. 그러나 이러한 관용은 오래된 마비노기 버전이나 덜 정교한 안티 치트 시스템에 국한될 수 있으며, 다른 넥슨 게임의 최근 제재 파동 [43]은 탐지 시스템의 엄격함과 업데이트를 나타낸다. 사용자의 질의는 "마비노기 모바일"에 대한 것이므로, PC 버전 마비노기나 다른 넥슨 타이틀과는 다른 안티 치트 구현이나 우선순위를 가질 수 있다. 원격 소프트웨어 사용의 위험 [43]은 겉보기에 무해한 백그라운드 프로세스도 안티 치트를 유발할 수 있음을 강조한다. 따라서 사용자는 과거의 관용에 의존해서는 안 된다. 업데이트된 안티 치트 시스템으로 인해 영구 제재의 위험은 현실적이다. 매크로는 공개적으로 "눈에 띄지 않는" 것을 넘어 완전한 탐지 불가능성을 목표로 해야 한다. "PC방 프리미엄 혜택" 및 "다수의 PC를 모니터링 및 제어하는 프로그램/기기"가 플래그될 수 있다는 언급 [45]은 넥슨이 다중 계정 또는 작업장과 같은 설정을 인지하고 적극적으로 표적화하고 있음을 시사한다.

넥슨 안티 치트 탐지 벡터 및 전략

| 탐지 유형 | 설명 | 지표/예시 | 넥슨의 알려진 사용 | 우회 과제 |
| --- | --- | --- | --- | --- |
| 시그니처 기반 | 알려진 치트 | 특정 파일 해시, | NGS/블랙 | 재컴파일, 코드 |

| | | | | |
|---|---|---|---|---|
| | 파일, 코드 패턴, 프로세스 이름 스캔 | 메모리 패턴, AutoMouseKey.exe와 같은 프로세스 이름 [18] | 사이퍼가 "불법 프로그램"을 탐지하고 게임을 종료시킴.[11] 과거 "딜미터기"가 프로세스 이름으로 차단된 사례 [사용자 질의]. | 난독화, 프로세스 이름 변경으로 쉽게 우회 가능.[16] |
| 휴리스틱/행동 기반 | 플레이어 행동의 비정상적인 패턴 분석 | 비정상적인 이동 속도, 점프 높이, 발사 속도, 반복적인 클릭 패턴, 비인간적인 반응 시간 [16] | "신호 처리를 이용한 반복 매크로 탐지".[1] AI/ML 기반 탐지.[3] | 인간의 불완전성을 모방하는 복잡한 입력 시뮬레이션 필요. 정교한 무작위화, 오류 주입, 자연스러운 지연. |
| 메모리 스캐닝 | 게임 메모리 내 무단 변경 또는 치트 코드 탐지 | 게임 값(예: 체력, 자원) 조작, DLL 주입, 코드 후킹 [23] | 블랙 사이퍼는 메모리 무결성을 확인하고 "파일 변조"를 플래그함.[10] | 직접적인 게임 메모리 조작 회피. 코드 및 데이터 난독화. 커널 수준 접근 시도 시 높은 탐지 위험.[47] |
| 네트워크 모니터링 | 게임 클라이언트와 서버 간 트래픽 패턴 분석 | 비정상적인 요청 빈도, 패킷 주입, 비정상적인 데이터 전송량 [30] | "패킷 주입"으로 인한 연결 끊김 사례 보고.[30] AI 기반 봇 탐지.[33] | 인간과 유사한 트래픽 패턴 모방. 요청 타이밍 무작위화, 비정상적인 시퀀스 회피. |
| 서버 측 유효성 검사 | 클라이언트 행동의 게임 규칙 준수 여부 확인 | 불가능한 이동 속도, 비정상적인 대미지, 비정상적인 자원 획득 [16] | "모든 입력은 악이다"라는 원칙.[17] 클라이언트의 비정상적인 상태는 동기화 해제 및 연결 끊김으로 이어짐.[29] | 게임 내 물리적/논리적 규칙을 위반하는 행동 금지. 서버가 궁극적인 권한을 가짐. |
| 커널 수준 모니터링 | 운영 체제 최하위 수준에서 시스템 | 모든 실행 중인 프로세스, 파일 | 블랙 사이퍼는 커널 수준에서 | 매크로의 시스템 흔적 최소화. |

| | | | | |
|---|---|---|---|---|
| | 활동 감시 | 접근, 브라우저 탭, 네트워크 연결 등 광범위한 데이터 수집 [7] | 작동하며, 매우 침투적인 로깅 시스템을 가짐.[9] | 프로세스 난독화, 가상 머신/원격 플레이 탐지 회피 (그러나 완벽하지 않음).[45] |

## 인간과 유사한 입력 생성: 기본 무작위화 그 이상

사용자의 랜덤 오프셋과 최소 200ms 딜레이 아이디어는 좋은 출발점이지만, 현대 안티 치트 시스템을 우회하기 위해서는 훨씬 더 정교한 접근 방식이 필요하다. 인간의 반응 시간은 평균 약 250ms이며, 가장 빠른 경우 100-120ms이다.[53] 고정된 최소 200ms는 여전히 너무 일관적으로 보일 수 있다.

안티 치트 시스템은 행동 분석 [3]과 신호 처리 [1]를 사용하여 반복적이고 비인간적인 패턴을 탐지한다. 고정된 딜레이나 완벽하게 균일한 랜덤 딜레이는 여전히 패턴으로 인식될 수 있다. 인간의 반응 시간은 균일하지 않으며, 변동성과 가끔의 오류를 포함한다.[26] 따라서 매크로는 이러한 불균일성과 불완전성을 시뮬레이션해야 한다. 이는 단순히 딜레이를 무작위화하는 것을 넘어, 딜레이와 클릭 지속 시간에 가우시안 분포와 같은 분포를 사용하고, 약간의 오클릭이나 일시적인 멈춤과 같은 미묘하고 현실적인 "실수"를 의도적으로 주입하는 것을 포함한다.

### 마우스 움직임 시뮬레이션

- 고급 궤적 생성: 베지어 곡선 및 스플라인 보간: 이러한 알고리즘은 인간의 행동을 모방하는 부드럽고 자연스러운 커서 궤적을 생성하여, 단순한 봇의 "들쭉날쭉"하거나 순간적인 움직임을 피한다.[57] human_mouse (Python)와 같은 라이브러리는 "초현실적인" 움직임을 위해 명시적으로 베지어 곡선과 스플라인 보간을 사용한다.[57]
- 자연스러운 노이즈 주입: 가우시안 및 퍼린 노이즈: 마우스 움직임에 가우시안 또는 퍼린 노이즈와 같은 임의의 노이즈를 추가하면 현실적인 흔들림과 가변성을 도입하여 자동화된 움직임을 덜 로봇처럼 보이게 할 수 있다.[59] 가우시안 노이즈는 정규 분포를 따르며 실제 변동을 시뮬레이션하는 데 사용된다.[62] 퍼린 노이즈는 부드럽고 유기적이며 겉보기에 무작위적인 패턴을 생성한다.[64]
- 무작위 오프셋 및 미세 움직임: 초기 클릭 지점 외에도, 최종 목표에 도달하기 전에

작고 무작위적인 미세 움직임 또는 "오버슈트" 및 "재조정"을 추가하면 현실감을 더욱 높일 수 있다.[59] 인간의 마우스 움직임은 거의 완벽하게 직선적이지 않다.[60]

단순한 랜덤 오프셋과 딜레이는 신호 처리 [1] 또는 "비인간적 패턴"을 찾는 AI/ML 모델 [17]에 의해 쉽게 탐지될 수 있다. 인간은 동일한 픽셀을 반복적으로 클릭하지 않으며 [46], 완벽하게 직선으로 움직이지도 않는다.[60] 따라서 매크로는

제어된 불완전성을 도입해야 한다. 베지어 곡선과 스플라인 [57]은 부드러운 곡선 경로를 제공하며, 가우시안/퍼린 노이즈 [59]는 완벽한 알고리즘과 인간 움직임을 구별하는 미묘하고 자연스러운 "흔들림"을 추가한다. "오버슈트" 및 "재조정" 행동 [59]은 핵심적인 인간의 특징이다. 매크로는 베지어 곡선이나 스플라인을 사용하여 마우스 경로를 생성하고, 경로를 따라 좌표에 가우시안 또는 퍼린 노이즈를 추가해야 한다. 최종 클릭 목표 전에 약간의 오버슈트와 교정 미세 조정을 구현해야 한다. 목표 클릭 지점 자체도 정확한 중심에서 약간 무작위 오프셋을 가져야 한다.

클릭 패턴 및 타이밍 인간화

- 가변 클릭 지속 시간 **(**키 다운**/**키 업 딜레이**):** 인간은 키를 즉시 누르거나 떼지 않는다. 각 키 누름에는 "누르는 시간"이 있다.[66] 매크로는 마우스 버튼이 해제되기 전에 "눌려 있는" 시간을 현실적인 범위 내에서 무작위화하여 시뮬레이션해야 한다.[67]
- 클릭 간 비균일 딜레이 분포**:** 사용자의 최소 200ms 딜레이는 시작점이지만, 인간의 반응 시간은 다양하다. 평균 인간 반응 시간은 약 250ms이며, (프로 게이머와 같은) 더 빠른 반응은 100-120ms 정도이다.[53] 행동 간 딜레이는 고정되거나 균일하게 무작위화되어서는 안 되며, 인간의 가변성을 모방하는 분포를 따라야 한다. 복잡한 행동에는 더 긴 딜레이를, 단순하고 반복적인 행동에는 더 짧은 딜레이를 주는 경향이 있을 수 있다.
- 자연스러운 멈춤 및 유휴 시간 도입**:** 실제 플레이어는 휴식을 취하거나, 다른 화면을 보거나, 단순히 멈춘다. 매크로에 무작위적이고 더 긴 멈춤(예: 몇 분마다 또는 일련의 행동 후) 및 유휴 시간 [70]을 구현하면 매크로가 더 인간적으로 보일 수 있다. 일부 매크로 도구는 명시적인 "#pause" 명령을 제공하기도 한다.[71]
- 인간 오류 시뮬레이션**:** 가끔씩 미묘한 "오류"를 의도적으로 주입하는 것은 강력한 우회 기술이 될 수 있다. 여기에는 약간의 오클릭(목표에 정확히 클릭하지 않고 근처를 클릭), 다시 시도해야 하는 가끔의 클릭 실패, 또는 약간 지연된 반응이 포함될 수 있다.[26] AI 봇은 더 사실적으로 보이기 위해 "고의적으로" 실수를 저지르도록 훈련되고 있다.[26]

딜레이는 인간 평균(예: 250ms)을 중심으로 하는 가우시안 분포를 사용하여 행동 간에

구현되어야 하며, 자연스러운 분산을 위해 표준 편차를 포함해야 한다. 클릭 다운 지속 시간 또한 무작위화되어야 한다(예: 50-150ms). 주기적으로 더 길고 무작위적인 "유휴" 기간을 도입해야 한다. "오클릭"이 발생하여 재타겟팅이 필요한 매우 낮은 확률을 추가하는 것도 고려할 수 있다.

키보드 입력 시뮬레이션 **(**생활 콘텐츠에 해당되는 경우**)**

생활 콘텐츠에 타이핑이나 특정 키 누름이 포함된다면, 유사한 원칙이 적용된다.

- 가변 키 누름**/**해제 시간**:** 마우스 클릭과 유사하게 개별 키 누름에 대한 가변 "누르는 시간"을 시뮬레이션해야 한다.[66]
- 자연스러운 타이핑 속도 및 멈춤**:** 문자열을 입력할 때, 인간의 타이핑 속도 변동과 단어 또는 문장 사이의 자연스러운 멈춤을 모방하여 키 입력 간에 가변적인 딜레이를 도입해야 한다.[74] 키스트로크 다이내믹스는 알려진 생체 인식 정보이다.[66]

게임이 행동 분석의 일환으로 키보드 입력을 모니터링한다면 (이는 안티 치트에서 흔하다[74]), 단순하고 균일한 키 입력은 탐지될 것이다. 목표는 모든 시뮬레이션된 입력을 인간의 입력과 구별할 수 없게 만드는 것이다. 따라서 모든 키보드 입력에도 마우스 클릭과 마찬가지로 무작위 딜레이와 지속 시간을 적용해야 한다.

인간과 유사한 입력 매개변수 및 권장 범위

| 입력 유형 | 매개변수 | 권장 범위/알고리즘 | 정당성 |
| --- | --- | --- | --- |
| 마우스 움직임 | 궤적 알고리즘 | 베지어 곡선 또는 스플라인 보간 | 인간의 움직임은 직선적이지 않고 부드러운 곡선을 그림.[57] |
|  | 노이즈 유형 | 가우시안 또는 퍼린 노이즈 | 인간의 움직임에는 미묘한 떨림과 불규칙성이 존재함.[59] |
|  | 오프셋 및 미세 움직임 | 목표 주변의 작은 랜덤 오프셋, 오버슈트 및 재조정 | 인간은 정확한 픽셀을 반복적으로 클릭하지 않으며, 목표에 |

| | | | |
|---|---|---|---|
| | | | 도달하기 전에 미세 조정을 함.[46] |
| 클릭 타이밍 | 클릭 다운 지속 시간 | 50ms ~ 150ms (랜덤화) | 인간은 키를 누르는 데 일정한 "누르는 시간"을 가짐.[66] |
| | 행동 간 딜레이 | 200ms 이상 (비균일 분포, 예: 가우시안) | 인간의 반응 시간은 평균 250ms이며, 가변성이 존재함.[53] 고정된 딜레이는 패턴으로 탐지될 수 있음. |
| | 유휴 시간 | 주기적으로 길고 무작위적인 멈춤 (예: 1분~5분마다 5초~30초) | 실제 플레이어는 게임 플레이 중 휴식을 취하거나 다른 활동을 함.[70] |
| 키보드 입력 | 키 누름/해제 시간 | 마우스 클릭과 유사하게 랜덤화 | 키보드 입력에도 인간적인 "누르는 시간"이 존재함.[66] |
| | 타이핑 속도 및 멈춤 | 키스트로크 간 가변 딜레이, 단어/문장 간 자연스러운 멈춤 | 인간의 타이핑 속도는 일정하지 않으며, 자연스러운 멈춤이 발생함.[74] |
| 일반적인 행동 | 오류 주입 | 매우 낮은 확률로 오클릭, 클릭 실패, 지연된 반응 | AI 봇은 더 사실적으로 보이기 위해 고의적으로 실수를 저지르도록 훈련됨.[26] |

## 마비노기 모바일을 위한 고급 우회 전략

### 프로세스 수준 우회

안티 치트 시스템은 알려진 치팅 소프트웨어와 실행 중인 프로세스를 스캔한다.[18] 블랙 사이퍼는 특히 "열린 모든 프로그램"을 로깅한다.[9]

- 프로세스 이름 난독화**:** "MabinogiMacro.exe"와 같이 명백한 이름을 피해야 한다. 대신 svchost.exe나 notepad.exe와 같이 일반적이고 합법적인 이름으로 위장하는 것이 도움이 될 수 있다.[48] 그러나 이는 행동 분석에 의해 비정상적인 활동으로 플래그될 수 있으므로 표면적인 방어에 불과하다.[48]
- 의심스러운 **API** 호출 회피**:** 안티 치트 시스템은 API 호출을 모니터링한다.[22] Windows API 함수( SendInput 등)를 사용하여 직접 입력을 조작하는 것은 매크로에서 흔한 일이다.[75] SendInput은 합법적인 기능이지만, 빠르고 반복적인 사용이나 비정상적인 시퀀스는 플래그될 수 있다.
- 프로세스 할로잉**/**주입 **(**고위험**):** 매크로 코드를 합법적인 프로세스(예: explorer.exe 또는 브라우저)에 주입하여 실행을 신뢰할 수 있는 프로세스 아래에 숨겨 일부 보안 제품의 탐지를 회피할 수 있다.[47] DLL 주입, 리플렉티브 DLL 주입(메모리에서 로드), 스레드 실행 하이재킹과 같은 기술이 포함된다.[47] 그러나 프로세스 주입 자체는 매우 의심스러운 활동이며 EDR/안티 치트 솔루션에 의해 모니터링된다.[47] 이는 커널 수준의 이해를 필요로 하며, 프로세스 접근, 명령줄(또는 부재), 모듈 로드 등을 모니터링하여 탐지될 수 있다.[48]

사용자는 NSG(넥슨 게임 시큐리티)가 자신의 프로그램을 관찰하지 못할 것이라고 언급했다. 이는 시그니처나 단순한 프로세스 이름으로는 프로그램이 플래그되지 않았기 때문일 가능성이 높다. 그러나 블랙 사이퍼의 커널 수준 접근 권한 [9]은 모든 실행 중인 프로세스와 그 활동을 볼 수 있음을 의미한다. 프로세스 주입은 알려진 우회 기술이지만 [47], 안티 치트 시스템은 이를 탐지하도록 설계되어 있다. 프로세스 주입에 대한 탐지 위험은 높으며, 이는 매우 의심스러운 시스템 수준 작업을 포함하기 때문이다.[48] 이는 고급 안티 치트가 이러한 기술의 탐지 능력을 지속적으로 향상시키는 "숨바꼭질" 게임이다. 따라서 실행 파일의 이름을 바꾸는 것은 기본적인 단계이지만, 더 깊은 프로세스 수준의 우회는 복잡하고 위험이 높다. 프로세스 주입은 고급 사용자에게 최후의 수단으로 고려되어야 하며, 이는 탐지 및 심각한 제재 가능성을 크게 높인다. 매크로의 프로세스를 숨기기보다는, 매크로의

행동이 합법적으로 보이도록 하는 데 중점을 두어야 한다.

메모리 수준 우회

안티 치트 시스템은 게임 코드나 값에 대한 무단 변경을 탐지하기 위해 메모리 스캐닝을

수행한다.[23]

- 메모리 스캐닝 우회**:**
  - 동적 메모리 할당**:** 매크로 내의 중요한 데이터에 정적 메모리 주소를 사용하지 않아야 한다.
  - 값 난독화**:** 예측 가능한 내부 상태 변수와 같은 민감한 값은 메모리에서 난독화되거나 암호화된 방식으로 저장해야 한다.[21] 그러나 이는 결심한 리버스 엔지니어에 의해 쉽게 우회될 수 있다.[24]
  - 직접 메모리 접근**(DMA) (**하드웨어 기반**,** 극심한 위험**):** DMA 치트는 외부 하드웨어를 사용하여 게임 메모리를 읽고 쓰며, CPU에서 실행되는 기존 안티 치트 소프트웨어를 우회한다.[78] 이는 매우 정교하고 비용이 많이 들며, 일반적으로 단순한 매크로의 범위를 벗어난다.
- 코드 및 데이터 난독화 기술**:**
  - 이름 난독화**:** 클래스, 메서드, 변수 이름을 모호하게 변경하여 목적을 숨긴다.[22]
  - 제어 흐름 난독화**:** 코드의 실행 경로를 기능 변경 없이 변경하여 추적을 어렵게 만든다.[22]
  - 데이터 난독화**:** 매크로 코드 내의 중요한 정보를 인코딩하거나 암호화한다.[21]
  - 관리되는 심볼 제거**/**디버깅 비활성화**:** C# Unity 애플리케이션의 경우, 심볼을 제거하고 스크립트 디버깅을 비활성화하면 리버스 엔지니어링이 더 어려워진다.[21]

메모리 스캐닝은 핵심적인 안티 치트 기능이다.[23] 난독화 [21]는 "초보 치터"를 막고 리버스 엔지니어링 속도를 늦출 수 있지만 [21], 결심한 공격자를 막지는 못한다. 이는 복잡성을 추가하지만 완벽하지는 않다. 사용자의 목표는 매크로이지, 본격적인 핵이 아니므로 깊은 메모리 조작은 과도하고 불필요한 위험을 초래할 수 있다. 매크로 실행 파일에 기본적인 코드 난독화(예: 변수 이름 변경, 기본적인 제어 흐름 변경)를 적용하여 단순한 정적 분석을 방해해야 한다. 직접적인 게임 메모리 읽기/쓰기는 안티 치트의 주요 표적이므로, 절대적으로 필요한 경우가 아니라면 피해야 한다. 게임 상태를 수정하기보다는 합법적인 입력을 시뮬레이션하는 데 집중해야 한다.

네트워크 수준 우회

사용자의 "패킷 캡처 프로그램이 있으니 네트워크 모니터링은 하지 않을 것"이라는 가정은 잘못되었다. 안티 치트 시스템은 네트워크 트래픽에서 이상 징후를 모니터링할 수 있다.[33]

- 인간 네트워크 트래픽 패턴 모방**:** 봇은 종종 반복적이거나 비정상적으로 높은 양의 요청을 생성한다.[33] 네트워크 트래픽을 인간화하려면 다음이 필요하다.

    - 무작위 요청 타이밍: 게임 동작을 서버로 보내는 딜레이를 다양하게 변경해야 한다.[40]
    - 반복적인 시퀀스 회피: 매크로가 고정된 일련의 동작을 수행한다면, 생성되는 네트워크 패킷이 너무 균일할 수 있다. 약간의 변형을 도입해야 한다.
    - 속도 제한: 매크로가 인간이 할 수 있는 것보다 빠르게, 또는 서버가 예상하는 것보다 빠르게 동작을 보내지 않도록 해야 한다.[21]
- 서버 측 유효성 검사 이해: 게임 서버는 클라이언트 동작의 유효성을 검사한다. 클라이언트가 합법적인 플레이어에게는 불가능하거나 매우 있을 법하지 않은 동작(예: 너무 빠르게 이동, 벽을 통과하여 클릭, 자원 즉시 생성)을 보내면 서버가 이를 탐지할 것이다.[16]

사용자의 "패킷 캡처 프로그램이 있으니 네트워크를 따로 모니터링하고 있지는 않을 거야"라는 주장은 논리적 오류이다. 클라이언트 측 패킷 캡처 도구의 존재가 서버 또는 커널 수준 안티 치트가 게임 클라이언트의 네트워크 활동을 모니터링하는 것을 배제하지 않는다. 실제로 많은 안티 치트 시스템은 네트워크 분석을 사용하여 봇을 탐지한다.[33] 서버 측 유효성 검사는 클라이언트가 우회할 수 없는 강력한 안티 치트 계층이다.[16] 따라서 매크로의 입력이 로컬에서 인간과 유사하게 보이더라도, 서버는 합법성 여부를 최종적으로 판단하는 주체이다. 매크로는 게임 규칙 내에서 인간 플레이어에게 물리적으로 가능하고 그럴듯한 동작만 시도해야 한다. 네트워크 트래픽 패턴은 자동화된 봇으로 탐지되지 않도록 무작위화되어야 한다.

환경적 고려사항

- 가상화 및 원격 플레이 탐지: 일부 안티 치트 시스템은 가상화된 환경(VM, Hyper-V)을 탐지하고 게임 실행을 차단할 수 있다.[45] 이는 VM이 치트를 호스트 시스템으로부터 격리하거나 안티 치트 후킹을 우회하는 데 사용될 수 있기 때문이다.[52] 원격 플레이 소프트웨어(예: Sunshine/Parsec)도 넥슨 게임에서 제재와 연관되어 있었다.[43]
- 하드웨어 기반 매크로 탐지: 내장 매크로 기능이 있는 게이밍 키보드/마우스는 소프트웨어에 의해 탐지하기 어렵지만 (합법적인 하드웨어 입력으로 보이기 때문에), 과도한 속도나 일관성은 행동 분석에 의해 여전히 플래그될 수 있다.[45]

사용자는 "가상 머신을 이용하여 접속/플레이하는 경우"가 넥슨의 "권장하지 않는 플레이 방식"이라고 언급했다.[45] 이는 VM이 완전한 안전을 제공한다는 생각과 직접적으로 모순된다. VM은 어느 정도의 격리를 제공하지만 [52], 안티 치트 시스템은 이를 탐지할 수 있다.[51] 마찬가지로 하드웨어 매크로는

프로세스 수준에서 탐지하기 어렵지만, 그 행동적 출력(완벽한 타이밍, 일관된 클릭)은 여전히 플래그될 수 있다.[45] 따라서 VM이나 원격 데스크톱을 통해 매크로를 실행하는 것은 탐지를 보장하는 방법이 아니며, 오히려 탐지 위험을 높일 수 있다. 매크로의 실행 환경과 관계없이 매크로의

출력을 인간화하는 데 초점을 맞춰야 한다.

## 프로그래밍 언어 선택: 매크로 개발을 위한 **C# vs. Python**

### 저수준 **Windows API** 상호 작용 및 성능을 위한 **C#**

C#은 정적으로 타입이 지정된 객체 지향 언어로, 특히.NET 생태계 내에서 강력한 성능을 제공한다.[82]

Windows Input Simulator( SendInput 사용)와 같이 키보드 및 마우스 입력을 시뮬레이션하기 위한 저수준 Windows API 상호 작용을 위한 훌륭한 라이브러리를 보유하고 있다.[75] 이는 입력 이벤트에 대한 정밀한 제어를 가능하게 한다. C#은 Windows 운영 체제와 직접 상호 작용하는 애플리케이션 개발에 적합하며, 독립 실행형 실행 파일로 컴파일될 수 있다. 마비노기 모바일의 게임 엔진인 Unity는 주로 C# 기반이므로, 직접적인 게임 클라이언트 상호 작용(예: 메모리 읽기, 고위험)을 시도한다면 이론적인 이점을 제공할 수 있지만, 매크로 우회에는 일반적으로 권장되지 않는다.

### 신속한 프로토타이핑 및 고수준 인간 유사 입력 라이브러리를 위한 **Python**

Python은 더 쉬운 학습 곡선과 높은 가독성의 구문을 제공한다.[82] 인간과 유사한 입력 시뮬레이션을 포함한 다양한 작업을 위한 풍부한 라이브러리 생태계를 자랑한다.

HumanCursor [83] 및

OxyMouse [59]와 같은 라이브러리는 현실적인 마우스 움직임을 위한 베지어 곡선,

스플라인 보간, 가우시안/퍼린 노이즈와 같은 고급 기능을 제공한다.

pynput [84]은 마우스 및 키보드 이벤트를 직접 제어할 수 있게 한다. Python은 신속한 프로토타이핑과 행동 모델 반복에 탁월하다. 그러나 Python은 일반적으로 더 큰 런타임 종속성(Python 인터프리터)을 가지며, 극도로 성능이 중요한 작업에서는 더 느릴 수 있지만, 단순한 클릭 매크로에는 큰 문제가 되지 않는다.

사용자는 이미 C#에 익숙하고 Python을 학습하고 있다. C#은 정밀한 입력 시뮬레이션에 좋은 직접적인 저수준 Windows API 제어를 제공한다.[75] 그러나 Python은 베지어 곡선 및 노이즈 생성과 같은 수학적 복잡성을 추상화하는

인간과 유사한 입력 시뮬레이션을 위해 특별히 설계된 고수준 라이브러리를 쉽게 사용할 수 있다.[57] 선택은 사용자의 우선순위에 따라 달라진다: 정밀한 제어(C#) 또는 복잡한 인간 유사 패턴 구현의 용이성(Python).

인간 유사 입력에 대한 강조와 사용자의 Python 학습 관심사를 고려할 때, Python이 원하는 행동 복잡성을 달성하는 데 더 효율적일 수 있다.

사용자의 특정 요구 사항 및 우회 복잡성을 기반으로 한 권장 사항

- 핵심 과제가 인간과 유사한 행동 시뮬레이션이며, 베지어 곡선 및 퍼린/가우시안 노이즈와 같은 복잡한 알고리즘을 추상화하는 정교한 Python 라이브러리 [57]의 가용성을 고려할 때,
  **Python**이 핵심 매크로 개발에 강력히 권장된다**.** 이는 더 빠른 반복과 현대 안티 치트를 우회하는 데 필요한 행동 복잡성을 더 쉽게 구현할 수 있게 한다.
- C#은 Python 라이브러리가 불충분하거나 사용자가 절대적인 제어 및 성능을 선호하는 경우 Windows 특정 저수준 상호 작용에 여전히 강력한 선택이지만, 인간 유사 알고리즘을 더 수동으로 구현해야 한다.

## 제안된 매크로 개발 계획

**1**단계**:** 핵심 입력 시뮬레이션 **(**인간과 유사한 마우스**/**키보드에 중점**)**

- 게임 창에 마우스 클릭 및 키 누름을 보낼 수 있는 Python(또는 C#)의 기본 매크로 프레임워크를 개발한다.
- 정의된 목표 영역 내에서 무작위 클릭 위치를 구현한다(사용자의 x,y 중심에 랜덤 오프셋).
- 마우스 움직임 경로에 베지어 곡선 또는 스플라인 보간을 통합한다(Python의 HumanCursor 또는 OxyMouse 라이브러리 사용, 또는 C#에서 사용자 정의 구현).
- 마우스 궤적에 가우시안 또는 퍼린 노이즈를 추가하여 자연스러운 흔들림을 구현한다.
- 최소 200ms 이상, 500-1000ms 이상까지 다양하게 변화하는 비균일(예: 가우시안) 분포를 따르는 행동 간 가변 딜레이를 구현한다.
- 클릭/키 누름에 대한 무작위 키 다운/키 업 지속 시간을 도입한다.

**2**단계**:** 안티 치트 상호 작용 및 우회 계층 **(**프로세스**,** 메모리**,** 행동**)**

- 기본적인 프로세스 수준 난독화(예: 매크로 실행 파일에 일반적인 프로세스 이름 사용)를 구현한다.
- 메모리 스캐닝 및 프로세스 주입 탐지를 유발하지 않도록 매크로가 게임 메모리를 직접 읽거나 쓰거나 게임 프로세스에 코드를 주입하지 않도록 한다.
- 반복적인 패턴을 깨고 인간의 휴식을 시뮬레이션하기 위해 주기적으로 더 긴 "유휴" 멈춤을 구현한다.
- "인간 오류"(예: 재타겟팅이 필요한 약간의 오클릭)의 발생 확률을 매우 낮게 추가하는 것을 고려한다.
- 매크로 자체의 리소스 사용량(CPU, 메모리)을 모니터링하여 눈에 띄지 않도록 한다.

**3**단계**:** 테스트**,** 모니터링 및 반복 **(**지속적인 적응의 중요성**)**

- 내부 테스트**:** 제어된 환경(예: 보조 계정 또는 가능한 경우 안전하고 격리된 게임 모드)에서 매크로를 실행하여 동작을 관찰하고 인간과 유사한 매개변수를 미세 조정한다.
- 행동 로깅**:** 매크로가 생성하는 마우스 움직임, 클릭 타이밍 및 딜레이를 로깅한다. 이러한 로그를 분석하여 AI/ML에 의해 탐지될 수 있는 "로봇 같은" 패턴이 남아 있는지 식별한다.
- 적응형 개발**:** "군비 경쟁"에 대비해야 한다.[16] 안티 치트 시스템은 지속적으로

업데이트된다.[43] 이는 매크로가 시간이 지남에 따라 지속적인 조정과 새로운 우회 기술을 필요로 할 수 있음을 의미한다.

- 위험 관리: 모든 노력에도 불구하고 탐지 및 제재 가능성은 남아 있음을 강조해야 한다. 사용자는 "무관용 원칙"[1]과 영구 제재의 위험[41]을 인지해야 한다.

## 결론: 끝나지 않는 군비 경쟁

"탐지 불가능한" 매크로를 만드는 것은 안티 치트 개발의 역동적인 특성으로 인해 일회성 해결책이 아니라 지속적인 도전 과제이다.[16] 가장 효과적인 우회는 공격적인 시스템 수준의 해킹보다는 정교한 인간 유사 입력 시뮬레이션과 최소한의 합법적인 시스템 흔적을 결합하는 데 달려 있다.

궁극적으로, 매크로의 목표는 시스템을 압도하는 것이 아니라, 게임 환경 내에서 인간 플레이어와 구별할 수 없을 정도로 자연스럽게 "섞여 들어가는" 것이다. 이를 위해서는 기술적 정교함뿐만 아니라 인간 행동의 미묘한 복잡성에 대한 깊은 이해가 필요하다. 지속적인 테스트, 모니터링, 그리고 안티 치트 시스템의 진화에 따른 적응은 성공적인 매크로 유지에 필수적이다. 사용자는 영구 제재의 위험을 분명히 인지하고, 이러한 복잡한 시스템을 개발하고 사용하는 데 따르는 잠재적인 결과를 이해해야 한다.

참고 자료


1. 인텔리전스랩스 탐지솔루션실을 소개합니다., 6월 27, 2025에 액세스, https://www.intelligencelabs.tech/1ecf2410-ef45-4d2f-8b32-81788d46cea7
2. 넥슨 인텔리전스랩스 테크블로그, 6월 27, 2025에 액세스, https://www.intelligencelabs.tech/
3. Securing Your Game: Anti-Cheating Strategies - Number Analytics, 6월 27, 2025에 액세스, https://www.numberanalytics.com/blog/securing-your-game-anti-cheating-strategies
4. AI in Mobile Games: The Future of Anti-Cheat Innovation, 6월 27, 2025에 액세스, https://intl.anticheatexpert.com/resource-center/content-70.html
5. 넥슨이 AI를 활용하는 방법 (Feat. 실제 사례), 6월 27, 2025에 액세스, https://www.intelligencelabs.tech/6adc3e4b-646a-4b6b-81b6-63b60b71330e
6. BlackCipher AntiCheat - SteamDB, 6월 27, 2025에 액세스, https://steamdb.info/tech/AntiCheat/BlackCipher/
7. No one cares about NGS/Blackcipher anti-cheat? :: The First Descendant General Discussions - Steam Community, 6월 27, 2025에 액세스, https://steamcommunity.com/app/2074920/discussions/0/4408543304944411881/
8. Nexon Guard Black Cipher. - MapleStory - GameFAQs, 6월 27, 2025에 액세스,

https://gamefaqs.gamespot.com/boards/924697-maplestory/62389088
9. Game Details for Counter-Strike Nexon - ProtonDB, 6월 27, 2025에 액세스, https://www.protondb.com/app/273110
10. If you kill blackcipher.exe anything bad happen? - MapleStory - GameFAQs - GameSpot, 6월 27, 2025에 액세스, https://gamefaqs.gamespot.com/boards/924697-maplestory/62394676
11. I received an "Illegal Program has been detected" message. - The First Descendant, 6월 27, 2025에 액세스, https://global.support.tfd.nexon.com/hc/en-001/articles/32896271887001-I-received-an-Illegal-Program-has-been-detected-message
12. 67-6f-64/rebecca: Reverse Engineering BlackCipher - GitHub, 6월 27, 2025에 액세스, https://github.com/67-6f-64/rebecca
13. The insanity of EA's anti-cheat system by a Kernel Dev : r/gaming - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/gaming/comments/xf1cwr/the_insanity_of_eas_anticheat_system_by_a_kernel/
14. According to experts on kernel level anticheat, two things are abundantly clear: 1) It's not perfect and 2) It's not going anywhere | PC Gamer, 6월 27, 2025에 액세스, https://www.pcgamer.com/according-to-experts-on-kernel-level-anticheat-two-things-are-abundantly-clear-1-its-not-perfect-and-2-its-not-going-anywhere/
15. How does Nexon Game Security works in the game? :: MapleStory 2 综合讨论, 6월 27, 2025에 액세스, https://steamcommunity.com/app/560380/discussions/0/3145094199295878852/?l=schinese
16. Anti-Cheat, An Analysis | Games | Maddy Miller, 6월 27, 2025에 액세스, https://madelinemiller.dev/blog/anticheat-an-analysis/
17. Anti-Cheat implementation - How does it work? : r/gamedev - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/gamedev/comments/1bkw49t/anticheat_implementation_how_does_it_work/
18. How does Anticheat implementation in Games work? : r/computerscience - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/computerscience/comments/1bkw3gq/how_does_anticheat_implementation_in_games_work/
19. Bot Detection - Programming - Linus Tech Tips, 6월 27, 2025에 액세스, https://linustechtips.com/topic/1337028-bot-detection/
20. In the Fight Against Ransomware, is Signature-Based or Behavior-Based Detection Best?, 6월 27, 2025에 액세스, https://www.soterosoft.com/blog/in-the-fight-against-ransomware-is-signature-based-or-behavior-based-detection-best/
21. Securing Game Code in 2025: Modern Anti-Cheat Techniques and Best Practices - Medium, 6월 27, 2025에 액세스, https://medium.com/@lzysoul/securing-game-code-in-2025-modern-anti-cheat-techniques-and-best-practices-e2e0f6f14173
22. How to Evade Antivirus with Custom Payloads in Python | by Maxwell Cross -

Medium, 6월 27, 2025에 액세스, https://medium.com/maxwell-cross-python-for-red-teaming/how-to-evade-antivirus-with-custom-payloads-in-python-2ff08fa23d2d
23. Memory Scanner - Game Hacking Academy, 6월 27, 2025에 액세스, https://gamehacking.academy/pages/7/03/
24. Has anyone tested/implemented Anti-Cheat Toolkit in Unity? : r/gamedev - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/gamedev/comments/1igb2qr/has_anyone_testedimplemented_anticheat_toolkit_in/
25. AI-Powered Anti-Cheat and Anti-Fraud Solutions for Gaming, 6월 27, 2025에 액세스, https://www.anybrain.gg/
26. How Multiplayer AI Bots Mimic Human Behaviour - TechRound, 6월 27, 2025에 액세스, https://techround.co.uk/tech/how-multiplayer-ai-bots-mimic-human-behaviour/
27. arxiv.org, 6월 27, 2025에 액세스, https://arxiv.org/html/2501.00078v1
28. Cybersecurity | Black Cipher | United States, 6월 27, 2025에 액세스, https://www.blackcipher.com/
29. Unbeatable anti-cheat. - Factorio Forums, 6월 27, 2025에 액세스, https://forums.factorio.com/viewtopic.php?t=127448
30. You have been kicked by Nexon Game Security for suspicious game or client behavior. Led to second temporary ban in span of 1 month. : r/Maplestory - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/Maplestory/comments/1i0n2pw/you_have_been_kicked_by_nexon_game_security_for/
31. I think I'm more interested in Anti-Cheat than GameDev - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/gamedev/comments/1kns80a/i_think_im_more_interested_in_anticheat_than/
32. Continuous Security Monitoring: A NetBrain Approach to Threat Detection, 6월 27, 2025에 액세스, https://datahubanalytics.com/continuous-security-monitoring-a-netbrain-approach-to-threat-detection/
33. What is a Traffic Bot? - Anura.io, 6월 27, 2025에 액세스, https://www.anura.io/fraud-tidbits/what-is-a-traffic-bot
34. Bunny Shield Bot Detection for Modern Web Traffic, 6월 27, 2025에 액세스, https://bunny.net/blog/introducing-bunny-shield-bot-detection-smart-silent-and-built-for-the-modern-web/
35. Bot Traffic Overtakes Human Activity as Threat Actors Turn to AI - Infosecurity Magazine, 6월 27, 2025에 액세스, https://www.infosecurity-magazine.com/news/bot-traffic-human-activity-threat/
36. Bot Traffic Surpasses Humans Online—Driven by AI and Criminal Innovation - SecurityWeek, 6월 27, 2025에 액세스, https://www.securityweek.com/bot-traffic-surpasses-humans-online-driven-by-ai-and-criminal-innovation/
37. Bot Detection 101: How to Detect (and Beat) Bot Traffic - Stytch, 6월 27, 2025에

액세스, https://stytch.com/blog/bot-detection-how-to-detect-bot-traffic/
38. Bot Protection & Prevention Software - Anura.io, 6월 27, 2025에 액세스, https://www.anura.io/bots
39. Addressing Network Packet-based Cheats in Multiplayer Games: A Secret Sharing Approach - arXiv, 6월 27, 2025에 액세스, https://arxiv.org/html/2501.10881v1
40. Best Web Scraping Detection Avoidance Libraries for Python | ScrapingAnt, 6월 27, 2025에 액세스, https://scrapingant.com/blog/python-detection-avoidance-libraries
41. 마비노기 모바일 게임 운영정책, 6월 27, 2025에 액세스, https://mabinogimobile.nexon.com/Support/Policy/2753857
42. PSA: Nexon will not punish you for using a bot/macro to AFK train a skill : r/Mabinogi - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/Mabinogi/comments/5mv6a8/psa_nexon_will_not_punish_you_for_using_a/
43. Can Nexon please address whats going on with these banwaves? : r/Maplestory - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/Maplestory/comments/1kspbix/can_nexon_please_address_whats_going_on_with/
44. Addressing the Recent Bans - MapleStory | News, 6월 27, 2025에 액세스, https://www.nexon.com/maplestory/news/general/27123/addressing-the-recent-bans
45. 권장하지 않는 플레이 방식 안내 - 아카이빙 센터 | 마비노기, 6월 27, 2025에 액세스, https://mabinogi.nexon.com/page/archive/guide_view.asp?id=4889849&num=7&playtarget=1
46. RuneScape - How macros are detected by RuneScape? - Offtopic ..., 6월 27, 2025에 액세스, https://www.autohotkey.com/board/topic/20049-runescape-how-macros-are-detected-by-runescape/
47. What is Process Injection? Techniques & Preventions - SentinelOne, 6월 27, 2025에 액세스, https://www.sentinelone.com/cybersecurity-101/cybersecurity/process-injection/
48. Process Injection - Red Canary Threat Detection Report, 6월 27, 2025에 액세스, https://redcanary.com/threat-detection-report/techniques/process-injection/
49. Process Injection Techniques using Windows Thread Pools, Defense Evasion & Privilege Escalation - Abusing Windows Internals : TryHackMe Walkthrough | by RosanaFSS | Medium, 6월 27, 2025에 액세스, https://medium.com/@RosanaFS/process-injection-techniques-using-windows-thread-pools-defense-evasion-privilege-escalation-ed0e85a07baa
50. Basic Anti-Cheat Evasion - sh3n, 6월 27, 2025에 액세스, https://mark.rxmsolutions.com/basic-anti-cheat-evasion/
51. learn.microsoft.com, 6월 27, 2025에 액세스, https://learn.microsoft.com/en-us/answers/questions/1530705/hyper-v-data-security-kernel-access-programs#:~:text=Some%20anti%2Dcheat%20systems%20might,a%20VM%20with%20GPU%20passthrough.

52. Hyper-V data security kernel access programs - Microsoft Q&A, 6월 27, 2025에 액세스, https://learn.microsoft.com/en-us/answers/questions/1530705/hyper-v-data-security-kernel-access-programs
53. How Fast is Human Reaction Time? Brain & Perception - PubNub, 6월 27, 2025에 액세스, https://www.pubnub.com/blog/how-fast-is-realtime-human-perception-and-technology/
54. Factors influencing the latency of simple reaction time - PMC - PubMed Central, 6월 27, 2025에 액세스, https://pmc.ncbi.nlm.nih.gov/articles/PMC4374455/
55. Macro Modeling of V-Shaped Electro-Thermal MEMS Actuator with Human Error Factor, 6월 27, 2025에 액세스, https://www.mdpi.com/2072-666X/12/6/622
56. (PDF) Micro- and Macro-Level Validation in Agent-Based Simulation: Reproduction of Human-Like Behaviors and Thinking in a Sequential Bargaining Game - ResearchGate, 6월 27, 2025에 액세스, https://www.researchgate.net/publication/5140549_Micro-_and_Macro-Level_Validation_in_Agent-Based_Simulation_Reproduction_of_Human-Like_Behaviors_and_Thinking_in_a_Sequential_Bargaining_Game
57. sarperavci/human_mouse: Ultra-realistic human mouse ... - GitHub, 6월 27, 2025에 액세스, https://github.com/sarperavci/human_mouse
58. Bezier curve - JavaScript.info, 6월 27, 2025에 액세스, https://javascript.info/bezier-curve
59. oxylabs/OxyMouse: Mouse Movement Algorithms - GitHub, 6월 27, 2025에 액세스, https://github.com/oxylabs/OxyMouse
60. Making mouse movements humanlike (using an arc rather than a straight line to the destination) - Stack Overflow, 6월 27, 2025에 액세스, https://stackoverflow.com/questions/8534189/making-mouse-movements-humanlike-using-an-arc-rather-than-a-straight-line-to-th
61. Gene Kogan: Perlin Noise, 6월 27, 2025에 액세스, https://genekogan.com/code/p5js-perlin-noise/
62. Gaussian Noise - GeeksforGeeks, 6월 27, 2025에 액세스, https://www.geeksforgeeks.org/electronics-engineering/gaussian-noise/
63. Gaussian Noise and how to remove it in Machine Learning, and understanding about NLM (Non Local Means) Denoising | by Abhishek Jain | Medium, 6월 27, 2025에 액세스, https://medium.com/@abhishekjainindore24/gaussian-noise-in-machine-learning-aab693a10170
64. Perlin Noise: A Procedural Generation Algorithm - Raouf's blog, 6월 27, 2025에 액세스, https://rtouti.github.io/graphics/perlin-noise-algorithm
65. Xetera/ghost-cursor: 🖱️ Generate human-like mouse movements with puppeteer or on any 2D plane - GitHub, 6월 27, 2025에 액세스, https://github.com/Xetera/ghost-cursor
66. Keystroke dynamics - Wikipedia, 6월 27, 2025에 액세스, https://en.wikipedia.org/wiki/Keystroke_dynamics
67. HoldKey - Macro Scheduler, 6월 27, 2025에 액세스,

https://www.mjtnet.com/manuals/b/v15/topics/holdkey.htm
68. Keyboard actions - Macro Recorder, 6월 27, 2025에 액세스, https://www.macrorecorder.com/doc/keyboard/
69. Anti-Cheat AutoClick & Macro Detection : r/hacking - Reddit, 6월 27, 2025에 액세스, https://www.reddit.com/r/hacking/comments/afs202/anticheat_autoclick_macro_detection/
70. Keyboard and Mouse Idle Time - Macro Library, 6월 27, 2025에 액세스, https://forum.keyboardmaestro.com/t/keyboard-and-mouse-idle-time/26465
71. Pausing Macros, great! but how? - Vizrt Forums, 6월 27, 2025에 액세스, https://forum.vizrt.com/index.php?threads/pausing-macros-great-but-how.236431/
72. Delays - Macro Express Pro - The Windows Automation Tool, 6월 27, 2025에 액세스, https://macros.com/helppro/Topics/Delay_Preferences.htm
73. Macro Recorder - Randomize parameters - YouTube, 6월 27, 2025에 액세스, https://www.youtube.com/watch?v=3oblDsu8Klw
74. What is Human Typing Simulation? - Multilogin, 6월 27, 2025에 액세스, https://multilogin.com/glossary/human-typing-simulation/
75. HossamElwahsh/AvaloniaInputSimulator: Windows Input Simulator (C# SendInput Wrapper - Simulate Keyboard and Mouse) works with Avalonia - GitHub, 6월 27, 2025에 액세스, https://github.com/HossamElwahsh/AvaloniaInputSimulator
76. c# - How to simulate keyboard input in ALL applications? - Stack Overflow, 6월 27, 2025에 액세스, https://stackoverflow.com/questions/8760397/how-to-simulate-keyboard-input-in-all-applications
77. michaelnoonan/inputsimulator: Windows Input Simulator (C# SendInput Wrapper - Simulate Keyboard and Mouse) - GitHub, 6월 27, 2025에 액세스, https://github.com/michaelnoonan/inputsimulator
78. Not Fair!!1!: Bypassing Anti-Cheat With Direct Memory Access - CypherCon, 6월 27, 2025에 액세스, https://cyphercon.com/portfolio/not-fair1-bypassing-anti-cheat-with-direct-memory-access/
79. Obfuscation explained: A comprehensive guide to code protection ..., 6월 27, 2025에 액세스, https://promon.io/resources/knowledge-center/code-obfuscation-techniques
80. Obfuscation Principles - Moataz Osama - Medium, 6월 27, 2025에 액세스, https://mezo512.medium.com/obfuscation-principles-16b8affb5f74
81. Macros trigger the Anti-cheat? :: HELLDIVERS™ 2 Anticheat, 6월 27, 2025에 액세스, https://steamcommunity.com/app/553850/discussions/2/4206994473257803717/?l=dutch&ctp=2
82. Python vs C#: The Battle of Titans | Shakuro, 6월 27, 2025에 액세스, https://shakuro.com/blog/python-vs-c-sharp
83. HumanCursor - PyPI, 6월 27, 2025에 액세스, https://pypi.org/project/HumanCursor/
84. Simulating User Input Easily in Python | by Coldstart Coder - Medium, 6월 27,

2025에 액세스,
https://medium.com/@coldstart_coder/simulating-user-input-easily-in-python-895c3eb32487