크리덴셜 스터핑의 공격면 이해하기

Akamai

서론

크리덴셜 스터핑 공격의 위험을 받고 있다면 이 공격을 효과적으로 방어하기 위해 봇 관리 벤더 또는 솔루션을 선택하는 것 이상의 노력이 필요합니다. 보안 솔루션이 효과를 발휘하는데 웹사이트 아키텍처가 중요한 역할을 담당합니다.

그 이유를 이해하기 위해 이 공격의 작동 방식과 보안 솔루션의 공격 방어 방식을 살펴보겠습니다. 크리덴셜 스터핑 공격은 유출된 인증정보로 애플리케이션에 자동으로 로그인하는 봇넷을 이용합니다. 자동화된 봇과 정상적인 인간 사용자를 구분하기 위해 최신 봇 탐지 기술은 웹 페이지 보안을 위해 JavaScript를, 네이티브 모바일 앱에서 사용되는 API 보안을 위해 모바일 소프트웨어 개발 키트(SDK)를 사용합니다. 웹사이트 아키텍처 및 웹사이트와 상호작용하는 클라이언트 종류에 따라 공격면을 최소화하는 능력이 제한될 수 있습니다.

본 백서에서는 최신 봇 관리 솔루션의 효과적인 도입과 관련된 구조적 문제, 크리덴셜 스터핑 공격을 성공적으로 방어하기 위한 이상적인 웹사이트 아키텍처, 공격면을 줄이는 중간 옵션, 각 옵션의 리스크와 제약 사항에 대해 알아봅니다.

봇을 탐지하는 방법

봇을 탐지하려면 요청 전송자가 인간인지 아닌지 식별해야 합니다. 간단한 봇 탐지 기법은 헤더 확인과 같이 요청 콘텐츠를 검사하는 것입니다. 하지만 요청 헤더를 쉽게 스푸핑(spoofing)할 수 있기 때문에 정교한 봇을 탐지하기는 어려운 방법입니다. 보다 진화한 봇 탐지 기법은 JavaScript 삽입, 브라우저 핑거프린팅, 이상 동작 분석을 사용합니다.

보다 진화한 봇 탐지 기법은 JavaScript 삽입, 브라우저 핑거프린팅, 이상 동작 분석을 사용합니다.

브라우저 세션에 JavaScript 삽입

대부분의 봇 탐지 기법은 클라이언트의 브라우저로 전송되는 웹 페이지의 HTML에 JavaScript 코드 스니펫을 삽입합니다. JavaScript 코드 스니펫은 브라우저가 HTML을 로딩할 때 실행되며, 인간과 봇을 구별하기 위해 다양한 기법을 적용할 수 있습니다.

정교함이 가장 떨어지는 기법은 JavaScript를 실행할 수 있는 브라우저에서 요청이 수신되는지 확인하는 JavaScript 삽입입니다. JavaScript 삽입은 JavaScript를 실행할 수 없는 단순한 봇을 탐지할 때 유용합니다.

더 정교한 기법인 브라우저 핑거프린팅은 브라우저 유형 및 버전, 화면 해상도, 설치된 플러그인, 설치된 글꼴 등 브라우저에 관한 다양한 특성을 수집합니다. 봇 관리 솔루션은 브라우저 핑거프린트를 검사하여 클라이언트가 정상적인 브라우저로 위장하려는 봇임을 나타내는 이상 현상을 식별합니다.

현재 가장 정교한 기법은 이상 동작 분석으로, 클라이언트의 입력 디바이스에서 동작 데이터(예: 컴퓨터의 키보드 입력 및 마우스 움직임이나 모바일 디바이스의 터치 이벤트, 자이로스코프 판독값, 가속도계 판독값)를 수집하는 방식입니다. 이 데이터는 봇 관리 솔루션으로 전달되고 봇을 판단하는 기준에 부합하는지 동작 패턴을 분석하는 용도로 사용됩니다.

애플리케이션용 모바일 SDK

JavaScript 삽입 기반 기법은 데스크톱, 모바일 브라우저 등 브라우저 기반 클라이언트에서 작동합니다. 하지만 네이티브 모바일 앱, 기타 자동화된 서비스 같은 대부분의 API 기반 클라이언트는 JavaScript를 실행할 수 없습니다. 결과적으로 JavaScript 삽입 기법은 이러한 클라이언트로부터 응답을 받지 못하게 됩니다. 봇 관리 솔루션은 적절한 응답이 없으면 클라이언트가 봇이라고 가정하고 이에 대한 조치를 취합니다.

이런 상황을 극복하기 위해 일반적으로 봇 관리 벤더는 네이티브 모바일 앱에 봇 탐지 기능을 통합하는 모바일 SDK를 제공합니다. 모바일 SDK가 있으면 모바일 앱이 앱 내에서 필요한 데이터를 수집한 후 봇 관리 솔루션으로 데이터를 전송해 분석할 수 있습니다.

웹사이트 아키텍처 이해하기

최신 웹사이트는 복잡할 뿐 아니라 수백 혹은 수천 개의 웹 페이지로 구성되어 있고 다양한 종류의 클라이언트와 트래픽을 지원합니다. 또한 웹사이트 소유권이 주로 조직 전반에 걸쳐 분산되어 있습니다. 크리덴셜 스터핑 공격을 방어하고 현재 직면한 리스크 수준을 파악하려면 웹사이트 아키텍처와 서로 다른 페이지에서 로그인 엔드포인트로 이어지는 클라이언트 흐름을 이해해야 합니다.

엔드포인트란?

엔드포인트는 클라이언트로 접속할 수 있는 개별 URL입니다. 크리덴셜 스터핑 공격을 방어하려면 사용자 인증정보를 확인하는 트랜잭션 URL을 식별하고 보호해야 합니다. 예를 들어, 기존 계정에 로그인하거나 새 계정을 생성할 때 URL이 사용될 수 있습니다. 크리덴셜 스터핑 외에도 보안이 필요한 엔드포인트에는 상품권 잔액 확인, 항공편 검색, 장바구니에 상품 추가 등 다양한 용도의 URL이 포함됩니다.

모든 엔드포인트 식별

대부분의 조직은 크리덴셜 스터핑에 취약한 여러 개의 엔드포인트를 가진 웹사이트를 운영합니다. 예를 들면, 금융 서비스 기관은 각각의 로그인 엔드포인트가 서로 다른 이용자, 소규모 비즈니스, 직원 서비스를 보유하고 있습니다. 또한 사업부별로 별도의 계정 가입 엔드포인트를 갖고 있을 수 있습니다.

일부 IT 또는 사업부 직원에게만 알려져 있는 보조 엔드포인트가 있는 경우도 있습니다. 일부 직원만 알고 있는 엔드포인트를 가진 레거시 앱도 있지만 집요한 공격자는 이런 앱을 찾아낼 수 있습니다. 여기에는 API, 매장 키오스크 또는 고객 서비스 챗봇을 서비스하는 엔드포인트가 포함될 수 있습니다.

모든 봇 관리 구현 계획의 첫 단계는 보호가 필요한 대상의 목록을 만드는 것입니다.

보안이 필요한 대상 파악

모든 봇 관리 실행 계획의 첫 단계는 보안이 필요한 대상의 목록을 만드는 것입니다. 하지만 최신 웹사이트의 규모와 복잡성을 고려할 때 이는 매우 어려운 작업입니다. 구체적인 사례는 다음과 같습니다.

- 보안이 필요한 대상을 생각할 때 주로 페이지만 생각하고 엔드포인트는 고려하지 않습니다.
- 하나의 로그인 엔드포인트는 주로 여러 페이지에서 사용됩니다. 예를 들어, 이커머스 사이트의 경우 모든 페이지에서 사용자가 로그인 인증정보를 입력할 수 있습니다. 하지만 인증정보는 확인을 위해 동일한 로그인 엔드포인트로 전송됩니다.
- 일반적으로 엔드포인트 유지 관리를 담당하는 팀은 로그인 엔드포인트로 연결되는 페이지를 생성하는 팀이 아닙니다. 예를 들어, 네이티브 모바일 앱은 데스크톱 사용자용 웹 페이지를 개발하는 팀과 조율 과정을 거치지 않는 다른 팀에서 주로 개발됩니다.
- 주어진 모든 엔드포인트에 보안을 적용하려면 클라이언트가 접근할 수 있는 모든 경로를 파악하고 해당 페이지에 보안 기능을 배포해야 합니다. 모든 당사자 간에 밀접한 조율 과정이 없으면 결국에는 보안이 취약한 엔드포인트로 접근하는 경로가 무방비 상태가 되는 결과를 초래합니다.

다양한 종류의 클라이언트를 위한 보안 아키텍처

웹사이트는 조직의 필요에 따라 여러 종류의 클라이언트와 상호작용할 수 있습니다. 클라이언트의 종류는 다음과 같습니다.

- 데스크톱, 노트북, 모바일 브라우저를 사용하는 정상 사용자 클라이언트
- 조직의 네이티브 모바일 앱을 사용하는 정상 사용자 클라이언트
- 금융 애그리게이터, 리셀러 파트너, 예약 파트너 등 자동화된 써드파티 서비스

모든 종류의 클라이언트를 보호하는 보안 체계를 구축하려면 상당한 노력이 필요합니다.

사용자 유형에 따른 엔드포인트 설계

크리덴셜 스터핑의 공격면을 최소화하려면 해당 클라이언트에 적절한 접속 권한을 제공해야 합니다. 그 이상의 권한은 필요하지 않습니다. 다양한 유형의 웹사이트 사용자는 서로 다른 요구사항을 갖고 있습니다. 예를 들어, 뱅킹 사용자는 잔액 및 내역서를 확인하고 금융 거래를 진행하며 자신의 프로필을 업데이트해야 합니다. 하지만 해당 사용자의 금융 애그리게이터에는 계좌 잔액 조회를 위한 접속 권한만 필요합니다. 사용자의 개별 요구사항을 확인하고 이에 따른 적절한 수준의 데이터 접속 및 기능을 제공하는 엔드포인트를 생성하면 모든 봇 관리 솔루션을 능가하는 보안상의 장점을 제공할 수 있습니다.

모바일 SDK의 한계

모바일 SDK는 네이티브 모바일 앱을 서비스하는 API 보안에 도움이 되지만 접근 방식 측면에서 제약사항이 존재합니다. 먼저, 애플리케이션 라이프사이클을 살펴보겠습니다. 웹 페이지 보안 기능을 구현하는 방식과 비교해서 모바일 앱 업그레이드는 시간이 많이 소요될 뿐 아니라 소프트웨어 개발 및 테스트 리소스도 투입되어야 합니다. 또한, 봇 방어를 구현하기 전에 모든 최종 사용자의 모바일 앱을 최신 버전으로 업그레이드해야 합니다. 가장 중요한 점은 모바일 SDK가 자동화된 써드파티 서비스를 통해 접속되도록 설계된 API 보안을 지원할 수 없다는 것입니다. 이러한 서비스는 봇 및 기타 자동화된 툴을 활용합니다. 브라우저 핑거프린팅, 이상 동작 분석과 같은 최신 봇 탐지 기법은 인간과 봇을 구별하는 데 도움이 되지만 정상 봇과 비정상 봇을 구별할 수 없습니다.

브라우저 핑거프린팅, 이상 동작 분석과 같은 진화된 봇 탐지 기법은 인간과 봇을 구별하는 데 도움이 될 뿐, 정상 봇과 비정상 봇을 구별할 수 없습니다.

하이브리드 URL

하이브리드 URL은 여러 유형의 클라이언트를 서비스하도록 설계된 엔드포인트이기 때문에 모든 봇 관리 솔루션의 보안 기능 구현을 어렵게 만드는 요인입니다. 클라이언트 종류에 따라 브라우저 기반 클라이언트를 위한 JavaScript 삽입과 네이티브 모바일 앱을 위한 모바일 SDK를 조합해 구현된 봇 탐지를 사용하여 엔드포인트를 보호할 수 있습니다. 예를 들어, 브라우저와 조직의 네이티브 모바일 앱을 통해 전달되는 정상 사용자 트래픽만 서비스하도록 설계된 엔드포인트는 봇 관리 솔루션으로 완벽하게 보호할 수 있습니다.

하지만 다른 경우에는 웹사이트 재설계하는 과정 없이 가용한 공격면을 전체적으로 최소화하는 것이 불가능할 수 있습니다. 예를 들어, 모바일 SDK는 네이티브 모바일 앱과 자동화된 써드파티 서비스에서 사용하는 API 보안을 지원할 수 없습니다.

크리덴셜 스터핑 방어: 중간 옵션 및 리스크

크리덴셜 스터핑은 복잡한 문제입니다. 봇은 정교하고 웹사이트 아키텍처는 보안을 어렵게 만듭니다. 조직과 웹사이트의 요구사항에 부합하는 100% 효과적인 솔루션은 이론적으로는 가능할지 모르지만 구현하기 어렵습니다. 크리덴셜 스터핑 방어 전략에서는 조직, 웹사이트, 모바일 앱에 대한 영향, 비용, 변경 적용 기간과 관련된 보안 리스크의 균형을 맞춰야 합니다.

옵션 1: 다양한 클라이언트 종류에 적합한 웹사이트 아키텍처 설계

공격면을 최소화하는 것이 목표라면 이상적인 해결책은 다양한 종류의 클라이언트에 맞게 웹사이트 아키텍처를 구축하는 것입니다. 정도의 차이는 있지만 이미 이 작업을 완료한 조직들도 있습니다.
다른 조직들은 이 목표 달성을 위해 웹사이트를 재설계해야 합니다. 기존 엔드포인트를 개별 URL로 나누면 트랜잭션 URL 트래픽을 가장 정밀하게 제어할 수 있으므로 공격면을 줄이는 데 도움이 됩니다.
예를 들어, 다음과 같이 클라이언트를 URL별로 분할할 수 있습니다.

URL 1: 데스크톱, 노트북, 모바일 브라우저를 사용하는 정상 사용자

URL 2: 네이티브 모바일 앱

URL 3: 업계 애그리게이터, 파트너 등 자동화된 써드파티 서비스

이 접근 방식에서는 적절한 봇 탐지를 URL 1 및 URL 2에 적용하고, 다른 유형의 사용자를 URL 3으로 안내할 수 있습니다. 이상 동작 봇 탐지를 사용하여 URL 3을 보호할 수는 없지만 해당 사용자에게 제공되는 데이터와 애플리케이션 기능은 제어할 수 있습니다.

장점	단점
최소 리스크 및 최소 공격면	웹사이트 또는 조직의 많은 부분을 수정해야 하기 때문에 프로젝트 규모가 과도하게 커질 수 있음 웹사이트 가용성이 최우선 목표인 경우 운영상의 차질이 발생할 수 있음

옵션 2: 네이티브 모바일 앱을 화이트리스트에 등록

브라우저 기반 클라이언트와 네이티브 모바일 앱을 모두 서비스하는 엔드포인트를 JavaScript 삽입 및 모바일 SDK를 제공하는 봇 관리 솔루션을 통해 보호할 수 있습니다. 하지만 적어도 당장은 모바일 앱을 벤더의 모바일 SDK로 업그레이드할 의향이 없는 조직도 있을 수 있습니다.

벤더의 SDK와 모바일 앱 통합 및 전체 사용자 기반 업그레이드가 상당한 노력을 수반한다는 점을 감안하여 임시방편으로 브라우저 기반 클라이언트를 위한 엔드포인트에 JavaScript 기반 삽입을 배포하고 요청 헤더의 일부(예: 사용자 에이전트 같은 헤더 필드 값)를 통해 모바일 앱을 화이트리스트에 등록할 수 있습니다.

장점	단점
작업이 쉬움 운영 중단 없음	모바일 앱을 화이트리스트에 등록했다는 사실을 공격자가 알아낼 때까지만 효과가 있음 공격자가 상대적으로 간단하게 앱으로 위장하여 보안 기능을 우회할 수 있음

옵션 3: 알려져 있는 자동화된 써드파티를 화이트리스트에 등록

클라이언트 트래픽(브라우저 또는 API 기반)과 제한적이고 알려진 자동화된 써드파티 서비스가 모두 동일한 엔드포인트에 접속하는 경우 자동화된 써드파티 서비스를 화이트리스트에 등록하는 옵션을 선택할 수 있습니다. 예를 들어, 은행은 Intuit Mint 같이 널리 사용되는 금융 애그리게이터의 IP 주소를 화이트리스트에 등록할 수 있습니다. 일반적으로 이러한 서비스는 잘 알려진 IP 주소 공간에서 운영되므로 공격자가 위장하기 어렵습니다. 명시적으로 화이트리스트에 없는 모든 자동화된 써드파티는 봇으로 처리됩니다.

장점	단점
작업이 쉬움 HTTP/HTTPS 프로토콜을 사용하여 IP 주소를 위장하기가 상대적으로 어려움	각 써드파티를 개별적으로 화이트리스트에 등록해야 함 화이트리스트에 등록된 써드파티가 사용자와 마찬가지로 동일한 데이터와 기능에 전체 접속 권한을 계속 보유하게 됨 모든 써드파티 서비스가 알려져 있고 그 수가 제한된 경우에만 구현 가능

다음 단계: 엔드포인트 아키텍처, 리스크, 옵션 이해하기

클라이언트(웹 브라우저, 네이티브 모바일 앱, 자동화된 써드파티 서비스)에 대해 독점 URL을 사용하지 않는 경우 웹 아키텍처를 반드시 검토해야 합니다.

첫 단계는 현재 공격면, 직면하고 있는 리스크 수준, 선택 가능한 옵션을 이해하는 것입니다. Akamai의 업계 최고의 크리덴셜 스터핑 보안 전문가는 고객의 크리덴셜 스터핑 방어 전략 수립을 위해 포괄적인 분석을 수행하고 플레이북을 개발합니다. 결과적으로 고객은 다음과 같은 지원 작업을 통해 웹사이트 아키텍처를 이해하고 현재 아키텍처가 모든 봇 탐지 솔루션의 구현에 어떤 영향을 미칠 것인지 파악할 수 있습니다.

1. 크리덴셜 스터핑으로부터 보호해야 하는 트랜잭션 URL과 요청을 URL로 전송하는 모든 HTML 양식의 엔트리포인트 식별
2. 해당 URL을 사용자 유형별로 분류
3. 하이브리드 URL 식별, URL 클라이언트 재구성 추천
4. 알려진 기술적 난제 식별 및 평가
5. 대상 및 정책 구조 매칭을 통해 URL 클라이언트 종류별로 적절한 보안 전략 결정
6. 크리덴셜 스터핑 공격 플레이북 개발, 봇 탐지 구현 계획의 다음 단계 수립