정 보 보 호 학 회 지
제26권 제3호, 2016. 6

실시간 대응 프로세스에 포커스를 맞춘 서버사이드 봇 탐지 프레임워크 구현 사례

강 병 탁*, 유 동 규**, 최 해 길***

요 약

서버사이드 탐지 시스템은 온라인 게임보안 업계에서 필수와 표준이 되어가고 있다. 이에 따라서 탐지시스템의 실시간 대응능력을 더욱 높이고 탐지 룰을 시스템의 점검시간 없이 더욱더 다이나믹하게 조정할 수 있도록 효율성을 증가시키는 방법론과 시스템적 프로세스가 게임 개발사나 게임 서비스 제공자, 데이터 분석가들에게 요구되고 있다. 이 논문에서는 기존에 존재하던 서버사이드 탐지 시스템의 한계를 살펴보고, 이를 어떤 식으로 극복할 수 있을지 실제 게임에 반영한 사례를 통하여 새로운 구조와 디자인을 제안하였다.

Ⅰ. 서 론

온라인 게임 보안이 발달함에 따라 그 탐지방법에 대한 연구가 계속 활발히 연구되어 왔다.[1,2] 클라이언트 사이드에서 API 후킹 등을 기반으로 한 전통적인 방식에 이어 최근에는 서버에서 행위기반 로그를 근간으로 봇과 핵을 탐지하는 기술이 발전되어 오고 있다. 게임 이용자들의 행위기반[3,4]이나 이동경로 기반[5]에 대한 탐지 기술이 서버사이드 분석에 대한 시초를 시작했고 최근에는 봇들의 현금 거래 내역을 데이터 마이닝 기법으로 탐지하여 게임 내 지하경제의 암거래나 골드 파밍의 기법을 탐지하는 논문도 발표되고 있다.[6,7] 또한 사용자의 대화 기록을 바탕으로 한 채팅 기록 메시지나[8] 파티 플레이의 로그를 이용한 소셜 액션과 연관된 서버사이드 탐지 시스템에 대한 연구도 이루어지고 있다.[9]

게다가, FPS (First Person Shooting) 게임 분야에서는 클라이언트 사이드에서 월핵(WallHak)이나 에임봇(Aimbot) 등은 DirectX 나 언리얼 엔진(Unreal Engine)의 특성상 클라이언트에서 보호를 할 수밖에 없다는 과거의 이론과는 달리, 이제 FPS 분야도 서버사이드에서 탐지를 하는 것이 정설이 되어가고 것처럼 게임의 장르를 가리지 않고 서버사이드 탐지 시스템은 온라인 게임 보안의 표준이 되어가고 있다.[10,11]

하지만 이같은 서버 사이드 탐지 시스템을 현업에서 사용하기에는 여러 가지 현실적인 제약이 존재하는데, 그 첫번째로는 로그의 처리 방법에 있다. 게임 플레이어의 모든 행동을 로그 파일로 남기는 작업은 매우 방대한 용량을 필요로 한다. 동시접속자의 수에 따라 결과는 달라지겠지만 기본적으로 하루에 수십 기가 이상의 용량을 필요로 하는 것이 일반적이다. 또한 플레이어의 과거 데이터 히스토리 트래킹을 위해 로그 보관의 주기도 최소 1년까지는 남겨놓는 것이 대부분 게임회사의 정책으로 사용되고 있기 때문에 게임 서버에서 로그 보관을 위하여 상당한 양의 하드디스크를 필요로 하며 이는 곧 스토리지 관리의 문제로 이어지게 된다. 더군다나 쌓인 로그를 분석하기 위해서는 수십 기가 급의 로그를 게임 서버에서 분석 서버나 분석가의 PC 로 복사하는 작업도 게임 서버의 퍼포먼스 저하로 이어지게 되며, 때로는 게임 서비스의 장애로 이어지게 된다. 그 때문에 게임 서비스를 잠시 중단시킨 점검 시간에 이를 수행해야 하는 등, 분석을 원하는 시점에 바로 로그를 볼 수 없는 문제도 가지고 있다.

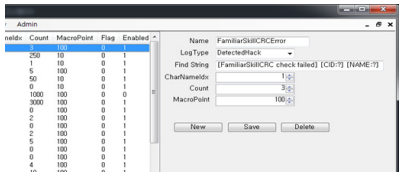
이런 문제를 극복하기 위해 보통 게임사에서는 로그

* 넥슨 아메리카 InfoSec Team, Sr. Security Manager
** 넥슨 아메리카 InfoSec Team Sr. Security Engineer
*** 넥슨 아메리카 InfoSec Team Lead Security Engineer

를 데이터베이스에 쌓고 일정 주기로 한 번씩 수행되는 잡(Job) 으로 분석 작업과 계정 제재를 수행한다. 하지만 앞서 언급한 퍼포먼스의 한계를 완전히 극복하지 못하였기 때문에, 보통 DB Job 은 하루에 한번 특히 서버의 퍼포먼스에 가장 영향이 적은 새벽 시간에 가동되며, 이것은 문제와 핵이나 봇이 제재가 일어나기 전인 최장 24시간 동안 게임 내에서 악의적 플레이를 계속할 수 있다는 문제점을 내재하고 있다. 이는 곧 게임사 입장에서 그동안은 아무런 대응을 할 수 없다는 한계로 이어진다.

오진 문제 역시 자동화된 서버사이드 탐지 시스템에서 매번 지적되는 심각한 문제로 등장하고 있다. 전통적인 클라이언트 사이드의 탐지 방식에서는 패턴 매칭 방식을 근간으로 하여, 탐지 결과도 C:\temp\hack.exe 와 같은 봇 프로그램으로 보이는 분명한 대상이 존재하였지만, 서버 사이드 탐지의 경우는 그렇게 구체적인 대상은 찾을 수 없기 때문에 대상 판별이 보호한 때가 많으며, 또한 고가의 아이템 사용이나 게임 유저의 플레이 패턴 변경으로 인하여 평소와 다른 흐름이 나올 경우 그것이 곧 봇의 움직임으로 오인사살한 오탐 이슈로 이어지기도 한다. 따라서 분석된 결과를 매번 수동으로 더블체크하는 작업이 필수적으로 동반되며, 이런 상황 때문에 휴먼 리소스 없이 자동으로 제재하는 시스템을 만들기 쉽지 않은 문제가 존재하게 된다.

탐지 기준값의 변경이 어려운 문제도 큰 난관 중 하나로 인지되고 있다. 평소에는 10 이상 15 미만의 스레시홀드를 설정해 둔 경우, 이를 다시 11 이상 16 이상으로 수정하려고 할 때, 게임 서버 소스코드에서 코드를 수정해 주어야 하거나 데이터베이스의 확장 프로시저(Stored Procedure)를 매번 수정해 주어야 하는 등, 탐지/제재를 수행할 때 기준값의 변경이 매우 번거로운 문제도 존재한다. 게임사에서는 이를 웹 운영툴로 전환하여 보다 간편하게 룰 설정을 처리하려는 시도도 많이 있어왔지만, 그러한 기준 값들을 실시간으로 변경하는 것은 일부 데이터에 한해 매우 제한적이기 때문에 좀 더 정밀한 데이터분석을 할 수 없는 문제가 있어 왔다. 또한 소스코드의 내부 로직이나 데이터베이스의 로직 자체도 웹 운영툴에서 수정하기는 구조적으로 어렵기 때문에, 결국 탐지 기준값을 실시간으로 조정해주기는 쉽지 않은 문제가 서버사이드 탐지 시스템의 고질적인 문제로 지적되고 있다.



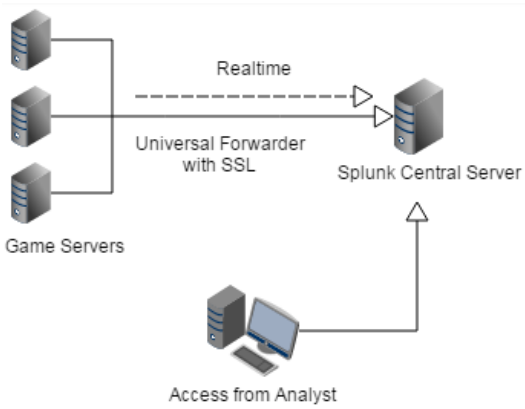
〔그림 1〕 제한적으로만 변경할 수 있는 실제 한 온라인게임의 운영툴

또한 탐지 시스템의 결과나 현황을 볼 수 있는 그래프 데이터인 가시성(visibility) 도 서버사이트 탐지 컨셉에서 중요한 부분 중 하나인데, 보통의 경우는 일일 단위로 취합한 그래프만 보여주는 형태가 일반적이라 실시간으로 현황이나 추이를 보여주는 시스템이 많지 않은 편이다. 데이터분석 작업의 습성상 분석시간에 소요되는 리소스를 실시간으로 반영하기 어려운 점이 있기 때문에, 일정 시간 주기로 그래프를 확인해야만 하지만, 그 주기가 시간 단위 혹은 일일 단위로 매우 크기 때문에 실제 운영조직에 있어서 이 문제는 가시성 문제나 현황파악 부분에 있어서 큰 불편을 가져다 준다.

마지막으로 작업 자체에 대한 R&R (Role & Responsibility) 문제도 부각이 되고 있다. 서버사이드 탐지 시스템은 여러 직군의 담당자들이 함께 모여 일을 하게 된다. 기본적으로 게임서버를 개발하는 게임개발자가 로그를 남기는 작업과 동시에 이 탐지시스템을 개발하기도 한다. 하지만 경우에 따라서는 데이터베이스로 로그 처리를 옮기는 탓에 DB팀에서 이 탐지 시스템을 개발하는 경우도 존재하며, 데이터마이닝 작업이 필요하다는 관점에서 통계 전문가들이나 데이터분석 직군에서 이 시스템을 개발하는데 참여하기도 한다. 또한 해킹대응이라는 관점에서 보안팀에서 이런 탐지시스템을 개발하는데 동원되기도 하는데, 이렇듯 워낙 많은 분야의 담당자들과 업무영역이 중복되기 때문에 이 시스템을 디자인하거나 이끌어갈 명확한 주체가 없다는 문제가 발생하며 애매모호한 R&R 문제로 이어지기도 한다.

Ⅱ. 새로운 구조와 디자인 제안

지금까지 논의된 내용에서 알 수 있다시피, 서버사이드 탐지 시스템을 설계할 때 가장 첫 번째로 발생하는 문제는, 로그를 수집하는 시간, 로그를 분석하는 시간, 그리고 DB Job 을 통하여 제재를 수행하는 시간 간격 등 처리시간과 연관된 한계점들이 가장 많이 거론되고



〔그림 2〕 스플렁크와 유니버설 포워더를 사용하여 데이터 수집을 할 때의 모식도

있다. 본 논문에서는 이런 한계를 극복하기 위하여 제안하는 것은 빅데이터 솔루션 중 하나인 스플렁크(Splunk)를 이용한 로그 처리의 수행이다.[12] 스플렁크는 다양하며 복잡한 로그를 쉽게 분석해줄 수 있는 시스템으로 서버사이드 탐지 시스템 설계 시 특히 시간적인 문제가 되는 부분을 해결할 수 있는 여러 가지 기능을 제공한다.

2.1. 로그 수집 시간과 분석 시간 단축

먼저 로그 수집 시간을 줄이기 위하여, 실시간 로그 수집을 처리할 수 있는 유니버설 포워더(Universal Forwarder)를 사용하였다.[13] 유니버설 포워더는 리눅스의 syslog 와 유사한 개념으로, 쌓이는 로그 파일을 실시간으로 중앙 서버로 전달해 주며, 로그 타입에 관계없이 한 줄 로그의 끝을 알려주는 개행처리만 확실히 인지되면 별도로 타입을 정의할 필요 없이 로그를 개별적으로 전송할 수 있다. 따라서 게임서버의 유저 행위 로그를 유니버설 포워더로 전송하는 방법을 선택한다면, 로그를 수집하는 시간을 절약할 수 있고, 게임 서버에 쌓이는 로그양에 따른 스토리지 문제도 해결할 수 있다. SSL을 기본적으로 지원하므로 데이터 전송구간에서 발생하는 보안 문제도 처리된다.

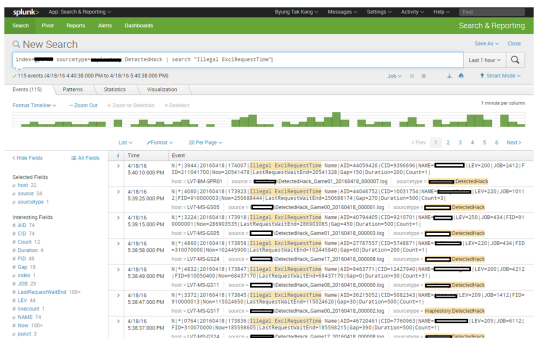
스플렁크는 쿼리 방식이라, 서버로그의 로우데이터를 어떤 형태로 조회하느냐에 따라서 데이터 분석의 결과가 달라진다. 서버로그를 바탕으로 원하는 쿼리를 입력한 뒤 레벨과 기준값을 조절하면 제재를 가하고 싶은 사용자의 계정을 추려낼 수가 있다. 그림 3을 보면 알 수 있듯이 간단한 문자열 서칭 방식의 쿼리를 통해서 원하는 결과를 뽑아낼 수 있고, 제재를 할 게임 이용자의 목록을 추출하는 작업도 어렵지 않게 가능하다.

2.2. API 연동 설계

여기까지는 사람이 직접 쿼리를 작성하는 인적 리소스의 자원이 필요한 상태이며, 이 쿼리의 결과를 바탕으로 24시간에 한번씩 제재를 수행한다면 단지 로그 수집만 실시간으로 진행되고 있을 뿐, 실제 제재 처리에 대해서는 전통적인 방식의 서버 사이드 탐지방식과는 크게 다르지 않다. 따라서 본 논문에서는 여기에 API 방식을 추가하여 자동화된 기능으로 제재를 할 수 있도록 디자인하였다. 백엔드 서버를 별도로 제작하고 메인 루프 로직을 작성한 뒤, 10분에 한번씩 쿼리를 던져서 해킹 사용자들을 추려내고, 해킹 유저가 발견되면 데이터베이스 서버에 제재를 수행하는 API를 호출하여, 자동적으로 차단 작업을 수행할 수 있도록 디자인하였다.

먼저 시간 제재 수행 작업을 위해 계정 제재를 처리할 수 있는 BAN API를 설계하였다. 본래 데이터베이스 사용자 테이블 디자인을 할 때 유저의 현재 상태를 알려주는 필드를 넣기 마련이며, 이 필드에 들어가는 항목으로는 정상 유저, 가입은 했지만 이메일 인증을 하지 않은 유저, 제재 대상인 유저 등 여러 가지 현재 상태를 구분하게 된다. 따라서 악의적인 행위를 한 사용자들은 데이터베이스의 이 필드에 사전에 정의한 값을 기록하게 되며, 이후 로그인을 할 때나 데이터베이스의 그 필드 값을 주기적으로 읽어와 악성 유저로 판단하여 게임에서 종료되도록 처리하는 것이 일반적이다.

따라서 서버 사이드 탐지 시스템을 개발할 때에는 해당 시스템이 가동되고 있다가 해킹을 시도한 유저가



〔그림 3〕 스플렁크를 통해 데이터를 추출해낸 모습

ApiServer/auth/ <id>/block	PUT	{"block_end_date":<block_end_date:"2014-01-01 00:00:00">,"block_msg":<block_msg>,"user_id":<usesr_id:default(null)>,"cs_memo":<cs_memo:default(null)>}

〔그림 4〕 REST API 의 PUT 메소드 내용. API에 들어가야 될 인자

발견되었으면 이 필드에 악성유저라는 값을 써넣을 기능이 필요하다. 서버 사이드 탐지 시스템 같은 외부 프로그램에서 데이터베이스의 SP 에 값을 넣는 작업으로 사용할 API 는 RESTful 방식으로 설계하였다.[14] 계정 제재를 수행할 REST API 에는 계정제재를 수행하는 시간, 그리고 계정ID, 그리고 사유 등이 포함되도록 하였다.

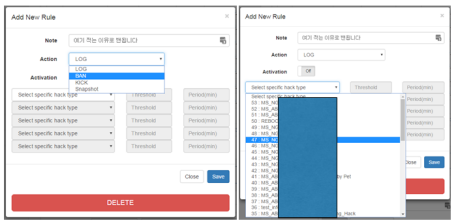
2.3. 부트스트랩(Bootstrap)을 이용한 웹 개발

그리고 스플렁크 쿼리의 관리를 위하여 웹 운영툴을 제작하였다. 서버사이드 탐지 시스템에서 가장 큰 문제로 지적되고 있는 속도의 효율성을 높이기 위하여 부트스트랩(Bootstrap)을 사용하였다.[15] 부트스트랩은 트위터(Twitter) 의 개발자가 오픈소스로 공개한 웹 UI 프레임워크로, 트위터 같은 심플한 웹 UI를 별도의 웹 디자이너 없이 쉽게 빠르게 구현해낼 수 있다. 또한 다양한 컴포넌트를 제공하는데 본 논문에서는 서버사이드 탐지시스템의 웹UI 에 목적에 가장 적합한 것으로 판단되는 SB Admin 2를 선택하였고.[16] 웹 운영툴의 이름은 SOC 라고 명명하였다.

하나의 스플렁크 쿼리를 만들어 두고 그것의 튜닝 과정을 거쳐 봇 탐지에 적합한 쿼리가 나온다면 그것을 저장할 수 있는 페이지를 만든다. 이것이 곧 하나의 룰이 되며, 또 다른 로그의 분석내용을 통하여 또 다른 쿼리가 나온다면 역시 두 번째 룰로써 내용을 입력할 수 있도록 웹 GUI를 설계한다. 그리고 하나씩 룰 넘버를 부여한다.

실시간 처리를 위하여 만들어진 룰은 세 가지 액션 중 하나를 선택할 수 있게 디자인한다. 첫 번째는 로깅 옵션으로, 만들어진 룰에 대해 실제 제재 작업을 수행하지 않고 단지 로깅만 할 수 있도록 작업한다. 룰을 반영한 뒤 사용자들이 즉시 피해를 입을 수 있기 때문에 로깅 모드로 해 놓고 어느 정도나 탐지가 되는지 확인하는 목적이다.

두 번째는 킥(Kick) 기능으로 룰에 걸리는 사용자가 나왔을 경우, 실제 계정 차단 작업은 수행하지 않지만



〔그림 5〕 SB Admin 2 로 개발한 서버사이드 탐지 시스템의 웹 운영툴

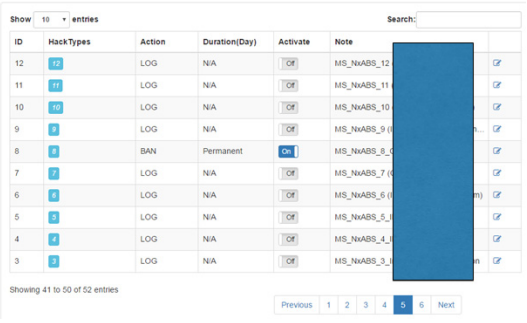
단지 게임에서 접속을 해제시키는 용도로만 사용한다. 해킹 시도의 위험수위에 따라서 유연하게 처리하는 방식 중 하나로, 굳이 게임에 다시 접속할 수 없도록 완전히 계정차단을 하는 것 보다, 현재의 해킹 행위는 중지시키되 다시 해킹시도를 하지 않을 것 같은 사용자에게 경고성으로 수행하는 방식이다. 해킹을 처음 시도한 초범 같은 사용자에게 적합한 옵션이다. 그리고 마지막 세 번째는 실제 제재를 수행하는 옵션이다. 또한 이 모든 룰은 실시간으로 on/off 할 수 있도록 UI를 만들어 둔다.

이렇게 해서 개발자나 DBA의 별도 작업 없이 탐지 기준의 스레시홀드를 변경할 수 있게 되었으며, 별도의 빌드 작업이나 서버 점검 없이 각 룰을 on/off 할 수 있도록 컨트롤 능력을 높이게 되었다.

2.4. 탐지/분석 수행 백엔드 서버 설계

이번에는 만들어진 웹 GUI를 가진 SOC 그리고 계정 제재 API 와 스플렁크 API를 이용하여 서버사이드 탐지 시스템의 메인 기능이 될 백엔드 서버를 제작하였다. 백엔드 서버의 로직 요약은 다음과 같다.

탐지 시스템의 메인 목적은 최대한 빠른 시간 내에



〔그림 6〕 룰마다 on/off를 처리할 수 있는 화면

해킹을 탐지하여 실시간으로 제재를 시키자는 데에 있다. 따라서 루프는 10분에 한번씩 가동되는 것을 기본으로 하며 10분 이내에 담당자가 사전에 SOC 사이트에 입력해 놓은 룰을 수집하여 스플렁크에 쿼리를 던져보고, 나온 결과값에 의거하여 계정 제재를 수행하는 것까지 마무리하는 것을 목표로 한다.

먼저 백엔드 서버의 메인 루프의 초기화 로직에서는 부트스트랩으로 개발한 웹 운영툴에 미리 기입해놓은 스플렁크 쿼리를 룰별로 모아오는 작업을 수행한다. 모아온 룰은 메모리상에 적재하여 둔다.

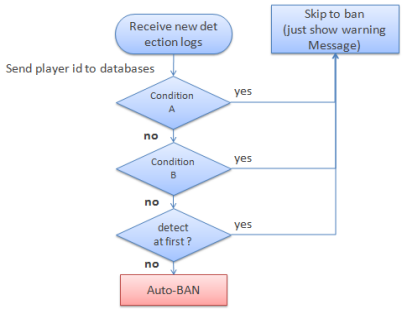
다음으로는 모아온 스플렁크 쿼리에 대해 API 호출을 수행하고 최근 10분간의 로그를 기준으로 그 결과를 받아온다. 결과가 0 보다 넘는 값이 나온다면 그것은 최근 10분 내에 해킹시도를 한 유저 또는 봇 유저가 있었다는 결과로 볼 수 있다.

결과가 나온 유저들에 대해서는 다시 SOC 사이트에서 결과 처리방식을 어떻게 할 것인지를 읽어온다 (Log, Kick, Ban) 담당자가 사전에 각 룰별로 설정해 놓은 세 가지 옵션 중 하나를 실시간으로 읽어들여 정의되어 있는 대로 처리를 수행한다. 그리고 1건 이상 탐지된 룰이 Ban 으로 선택되어 있다면 REST 방식으로 API 를 호출하여 사용자 데이터베이스에 계정 제재를 수행한다.

10분이 지나면 다시 루프의 처음부터 돌아가서, 10분 전에 체크한 로그의 이후 부분을 다시 골라내서 다시 로그분석을 수행한다. 그리고 최근 10분간의 로그에서 탐지 건이 있으면 다시 담당자가 SOC 에 설정한 내용대로 이후 액션을 수행한다.

2.5. 필터링 로직을 통한 오진 문제 해결

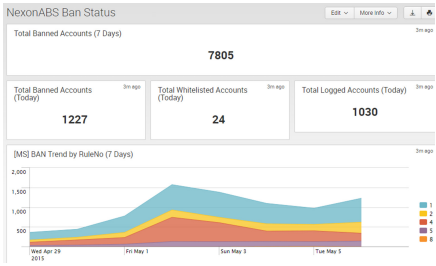
여기까지는 자동화된 기능으로 계정 제재를 수행하는 것 까진 완료되었지만 서론에서 제기한 문제대로 오진 문제를 피해갈 수는 없다. 그것을 해결하기 위해서 계정제재 API 호출 직전에 필터링 API를 하나 더 제작하는 방식의 디자인을 추가하였다. 조건문 A, 조건문 B, 등의 필터링 조건을 추가하고, 해당 조건에 부합하면 제재를 수행하지 않고 스킵하는 방식이다. 해당 조건은 가입년도나, 평소 소셜 액션에 투자하는 시간, 평소 플레이 패턴, 아이템 구매 현황 등 다양한 조건을 추가할 수 있다.



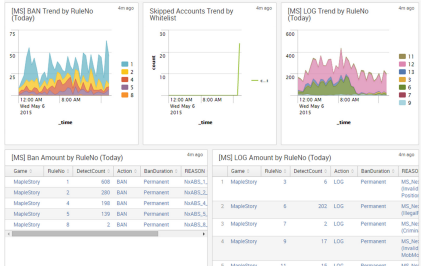
〔그림 7〕 오진 문제를 해결하기 위한 화이트리스트 로직 순서도

2.6. 대시보드를 통한 가시성 확보

이렇게 화이트리스트 로직까지 통과하여 만들어진 결과들은 매우 간단하게 그래프로 변환할 수 있다. 각 룰별로 탐지된 결과 내용을 스플렁크에서 자동적으로 그래프로 그려준다. 또한 유니버설 포워더를 통하여 실시간으로 들어오는 데이터를 바탕으로, 그래프 또한 실시간으로 업데이트 되는 것을 확인할 수 있다. 이런 식으로 서론에서 언급하였던 가시성 문제도 스플렁크로 간단히 해결하였다.



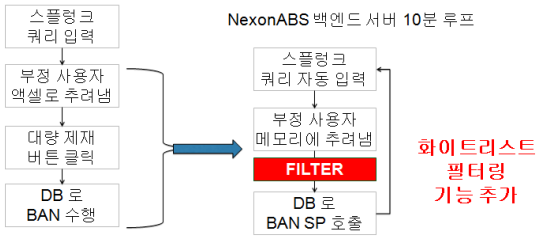
〔그림 8〕 탐지 결과에 따른 대시보드1



〔그림 9〕 탐지 결과에 따른 대시보드2

2.7. 구조도 요약

그림 10에서는 전통적인 방식의 서버사이드 탐지 프로세스를 지금까지 제안한 내용을 바탕으로 한 백엔드 서버로 업그레이드한 과정을 간단한 순서도로 표현하였다. 수작업으로 진행되던 부분을 백엔드 서버와 API 호출로 활용하여 10분에 한번씩 실시간으로 계정 제재가 일어나도록 구조가 바뀌었다는 것을 알 수 있다.



〔그림 10〕 백엔드 서버에서 진행되는 작업

Ⅲ. 실제 온라인게임 도입 사례

앞 장에서 제안한 디자인을 토대로 새로운 서버사이드 탐지 시스템을 개발하였고, N사의 M게임에 적용한 실험 결과를 소개한다. 새로 제안된 서버사이드 탐지 시스템의 개발기간은 3개월이 소요되었고 2015년 4월부터 적용을 시작하였다. 먼저 봇 현황 관련하여 현재 M게임이 처해 있는 문제는 다음과 같았다.

M게임은 장수 게임으로 이미 봇들이 활발하게 활동중이었고 안랩의 핵쉴드라는 클라이언트 사이드의 게임 보안 솔루션이 가동되고 있었다. 하지만 보안솔루션으로 얻는 차단효과는 거의 미미하였기 때문에 거의 의존하지 않는 상황이었다. 또한 봇 탐지에 큰 도움이 될 수 있는 다양한 타입의 많은 게임서버로그가 존재하고 있었지만, 로그 수집의 불편함으로 인하여 활용도가 높지 않은 상태였다. 물론 일부의 로그를 기반으로 서버사이드 탐지 시스템이 가동되고 있었지만 자동 제재 기능에 대해 오탐이 많은 문제가 있었기 때문에, 자동제제 기능은 비활성화 상태로, 매번 담당자들의 수작업으로 제재가 이루어지고 있던 상황이었다. 마지막으로 해킹대응이 적극적으로 이루어지지 않고 있던 탓에 게임 내 골드의 인플레이션 현상이 심하여 게임 경제가 좋지 않은 문제를 가지고 있었다.

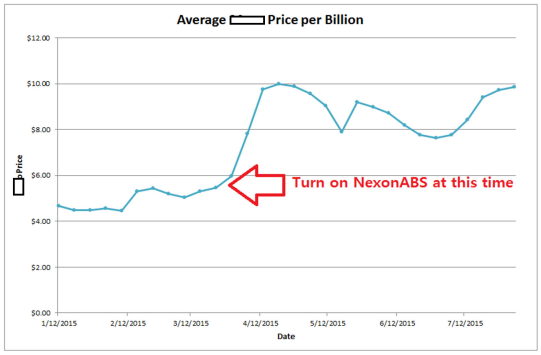
새로운 서버사이드 탐지시스템 도입 후, 서론에서 제시하였던 문제점이나 한계점을 대부분 해결할 수 있었다. 먼저 게임서버의 스토리지 부족 문제로 이어지던 로그양에 대한 문제점이 해결되었다. 게임서버에서는 로그 생성 후 유지 기간을 일주일 이하로 설정한 뒤 그 이후가 된 로그를 모두 삭제해도 스플렁크로 중앙관리되는 로그가 있기 때문에 해킹대응에 아무런 지장을 가져오지 않았다.

또한 실시간으로 전달되는 로그의 구조 덕에 24시간 동안 모아두었다가 제재를 가하는 기존의 프로세스에서 벗어나서 실시간 데이터분석이 가능해졌으며, API와 연동된 백엔드 서버의 가동으로 10분에 한번씩 제재를 수행함으로써 봇들이 실시간으로 게임 내에서 사라지기 시작했으며, 게임 이용자들로부터 해킹이 눈에 보이는데도 조치를 취하지 않고 있다는 항의들이 줄어들기 시작하였다.

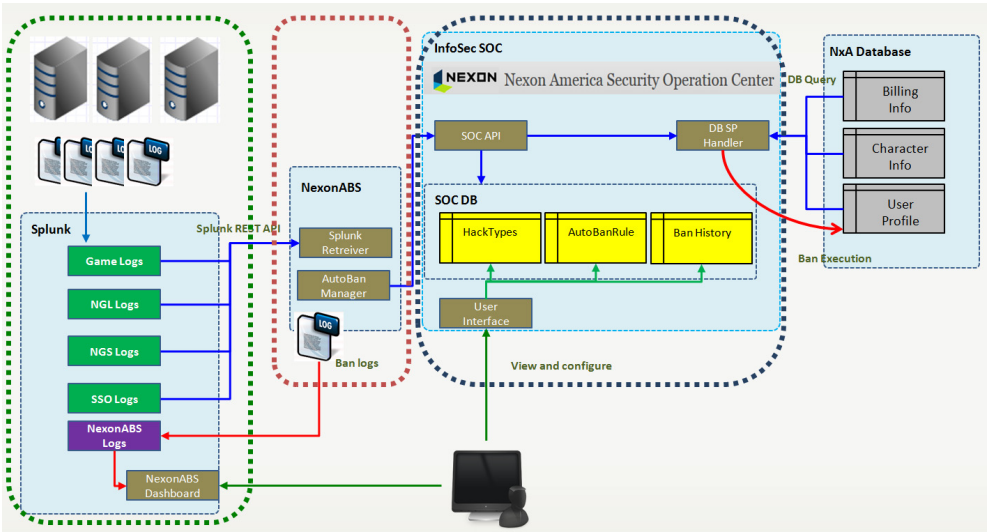
탐지 기준값에 대한 변경이 자유자재로 가능하게 되자, 운영조직에서도 개발팀이나 DB팀의 도움 없이 직접 다양한 시도를 할 수 있게 되었고, 게임 내 이벤트 등이 있을 때 이용자 수가 몰리는 특정 시간 동안 제재 룰을 잠시 느슨하게 해 놓는 등 다이나믹한 처리가 가능해졌다.

그래프를 자유자재로 그릴 수 있게 되자 가시성이 높아져 현재 탐지 현황에 대해 좀 더 구체적으로 알 수 있게 되었으며, 별도의 개발 리소스 없이 운영조직이나 GM (Game Master) 들이 새로운 그래프를 만들어서 가시성도 더욱 효율적으로 확보가 되었다.

오진이 발생하지 않게 되었다. 화이트리스트 룰을 SOC에서 자유자재로 다양한 조건으로 넣을 수 있게 되자 오진 확률이 더욱더 줄어들어갔으며 오진이 발생하지 않게 되자 자동 제재 시스템을 사용하는 빈도가 더



〔그림 12〕 시스템 반영 후 결과



〔그림 11〕 서버사이드 탐지 시스템의 전체적인 구조

욱 높아졌고 탐지 수는 더욱 높아져 갔다.

인플레이션 현상이 심하던 게임 경제가 서버사이드 탐지 시스템 반영 이후로 좋아지기 시작하였으며 그림 12 에서 알 수 있듯이 화폐 가치가 10억 게임머니당 $5 인 것이 $10 가까이 올라가기 시작하였다. 시스템 적용 후 1-2 개월만에 두 배 가까이 경제가 회복되었다.

Ⅳ. 발전방향과 과제

전통적인 서버사이드 시스템의 확장성 면에서의 문제는 그 게임의 로그를 파싱하는 작업이 들어가다보니 항상 특정 게임위주의 시스템에서 그쳤다는 부분이다. 좋은 시스템을 개발하여도 다른 게임에 탑재하려면 휴먼 리소스가 많이 들어간다는 문제를 들며 결국 하나의 게임 서비스만을 위한 시스템에 국한되어 버리는 문제를 가지고 있다.

게임 서비스 제공자들은 하나의 포털 안에 여러 가지 게임을 서비스 하기 때문에 가급적 통합된 시스템을 개발하는 것을 원하고, 각 게임별로 일일이 시스템을 구축하는 것 보다는 하나의 백엔드 서버 안에서 그리고 하나의 웹 운영툴 안에서 모든 게임서비스를 컨트롤 하기를 원하는 것이 일반적이다. 또한 이런 방향으로 가는 것이 중복 업무를 줄이는 휴먼 리소스 절약 부분에서 큰 이점을 가져다 준다.

따라서 이러한 탐지시스템을 개발할 때에는 언제나 새로운 다른 서비스를 감안하여 설계를 해야 한다. API 를 설계할 때에도 어떤 게임은 사용자ID를 기준으로 제재를 수행하는데, 어떤 게임은 캐릭터ID를 기준으로 제재를 수행하는 등, 비일관성 API 가 되지 않도록 충분히 검토해야 한다. 제재를 수행하는 데이터베이스의 SP 도 마찬가지로, 매번 게임마다 SP를 만들 필요 없이, 하나의 SP 안에서 게임 이름이나 제재 사유 등을 API 의 인자로 받아서, 여러 게임을 커버할 수 있는 형태로 구조를 만들어야 한다. 하나의 시스템으로 통합시켜야 되는 이유는 네트워크 보안 측면에서도 마찬가지다. API 호출을 위해서는 백엔드 서버와 데이터베이스 서버와의 네트워크 연결이 이루어져야 하는데, 만약 탐지 시스템이 각 게임마다 각기 나뉘어져 있다면 서버를 각각 만드는 것은 물론, 새로 만든 서버마다 데이터베이스 서버로 매번 ACL을 오픈해야 한다. 이는 네트워크 담당자들의 룰 관리 이슈로도 이어지며, 네트워크 보안 측면에서나 보안정책 측면에서도 매우 좋지 않은 구조로 이어진다.

그런 면에서 본 논문에서 제시하였던 서버사이드 탐지 시스템은 스플렁크와 부트스트랩을 이용하였으므로 확장성이 아주 용이한 부분이 있다. 다양한 타입의 로그 분석이 가능한 스플렁크의 장점대로 새로운 게임의 새로운 로그가 들어와도 스플렁크로 역시 쉽게 분석하여 룰을 만들 수 있으며, 웹 개발 리소스를 줄이고 반복되는 작업을 없애주자는 취지를 잘 보여주는 부트스트랩

을 통하여 새로운 웹 운영페이지도 쉽게 추가할 수 있다. 새 프레임워크나 라이브러리 그리고 오픈소스를 사용할 때에도 언제나 확장성을 고려하여 선택을 해야 한다.

Ⅴ. 결 론

서버사이드 탐지 시스템에 있어 중요한 것은 얼마나 정확한 데이터를 분석해 내느냐는 것이 테크놀로지의 핵심이다. 하지만 아무리 정확히 탐지를 해 낸다 하더라도 그것을 효과적으로 사용할 수 있는 시스템 환경이 준비되어 있지 않다면, 보안 시스템으로서의 가치는 다소 떨어질 수 있다. 본 논문을 통하여 서버사이드 탐지 시스템을 개발할 때는, 데이터 분석의 질도 중요하지만 시스템의 아키텍쳐나 업무 흐름도를 통하여 시스템을 보다 완성도 있고 효율적으로 사용할 수 있는 환경을 꾸미는 작업도 반드시 필요함을 제시한다.

참 고 문 헌

[1] Jiyoung Woo, Huy Kang Kim, “Survey and Research Direction on Online Game Security,” Workshop at ACM SIGGRAPH ASIA 2012 , pp. 19-25, November 2012.

[2] 유동영, 서동남, 김휘강 , 최진영, “ 온라인게임 서비스 분야에 정보보호 사전진단 적용시 효과성에 관 한 연구”, 한국 IT서비스학회지, 10(2), pp. 293-308,

[3] Thawonmas, Ruck, et al., “Detection of MMORPG bots based on behavior analysis.”, Proceedings of the 2008 International conference on Advances in Computer Entertainment Technology. ACM, 91-94, 2008.

[4] Chen, Kuan-Ta, and Hong, Li-Wen, “User identification based on game-play activity patterns.”, Proceedings of the 6th ACM SIGCOMM M workshop on Network and system support for games. ACM, 7-12, 2007.

[5] van Kesteren, Marlieke, et al., “A step in the right direction: Botdetection in MMORPGs using movement analysis.”, Proceedings of the 21st Belgian-Dutch Conference on Artificial Intelligence. 2009.

[6] Fujita Atsushi, Hiroshi Itsuki, and Hitoshi Matsubara, “Detecting Real Money Traders in MMORPG by Using Trading Network,” IIDE, Oct. 2011

[7] Ahmad, M. A., Keegan, B., Sullivan, S., Williams, D., Srivastava, J., and Contractor, N., “Illicit bits: Detecting and analyzing contraband networks in Massively Multiplayer Online Games,”Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing, pp. 127-134, Oct. 2011

[8] Ah Reum Kang, Huy Kang Kim, Jiyoung Woo, ”Chatting pattern based game BOT detection: Do they talk like us?,“ KSII Transactions on Internet and Information Systems, 6(11), pp. 2866-2879, November 2012.

[9] Ah Reum Kang, Jiyoung Woo, Juyong Park, Huy Kang Kim, "Online game bot detection based on party-play log analysis", Computers & Mathematics with Applications, Volume 65, Issue 9, May 2013, Pages 1384–1395

[10] Jung Kyu Park, Mee Lan Han, Huy Kang Kim, "A Study of Cheater Detection in FPS Game by using User Log Analysis", Journal of Korea Game Society Vol.15 No.3 pp.177-188

[11] Kim, Seon Min; Kim, Huy Kang, "A research on improving client based detection feature by using server log analysis in FPS games", Journal of the Korea Institute of Information Security and Cryptology, Volume 25, Issue 6, 2015, pp.1465-1475

[12] Splunk, http://www.splunk.com/

[13] Universal Forwarder, https://www.splunk.com/en_us/download/universal-forwarder.html

[14] RESTful - Wikipedia, https://en.wikipedia.org/wiki/Representational_state_transfer

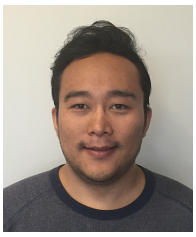
[15] Bootstrap, http://bootstrapk.com/

[16] SB Admin 2, http://startbootstrap.com/template-overviews/sb-admin-2/

〈저 자 소 개〉

강 병 탁 (Byungtak Kang)
현재 : 넥슨 아메리카 InfoSec Team,
Sr. Security Manager

최 해 길 (Hai Gil Choi)
현재 : 넥슨 아메리카 InfoSec Team
Lead Security Engineer



유 동 규 (Dongkyu Yoo)
현재 : 넥슨 아메리카 InfoSec Team
Sr. Security Engineer