한국컴퓨터정보학회 하계학술대회 논문집 제18권 제2호 (2010. 7)

# 일회용 실행 코드를 이용한 보안 시스템 및 보안방법

이세훈*, 전상표**, 신영진°, 박전진***
*인하공업전문대학 컴퓨터시스템과
**남서울대학교
°***웰비아닷컴
e-mail: seihoon@inhatc.ac.kr, spjun7129@dreamwiz.com,codewiz@wellbia.com, apuro@wellbia.com

# System and Method For Security Using ONE-TIME Execution Code

Se-Hoon Lee*, Sang-Pyo Jeon**, Young-Jin Shin°, Jeon-Jin Park***
*Dept. of Computer Systems & Engineering, Inha Technical College
**Nam-Seoul University
°***Wellbia.co, Co.,Ltd.

● 요 약 ●

본 논문에서는 기존 C/S 환경에서 이용되는 보안 방법에 대해 새로운 방식으로 접근하고 이를 통하여 전통적인 C/S 환경의 정보시스템에서 클라이언트를 위조하거나 변조하는 등 해킹시도 및 해킹 여부를 판별하여 안전한 클라이언트 시스템이 작동중임을 판별할 수 있도록 한다.

**키워드:** 일회용(ONE-TIME), 실행코드(Execution Code), 해킹 방어(Hacking protection)

## I. 서론

C/S 환경에서 이용되는 다양한 형태의 보안방식 중 클라이언트 시스템의 위조나 변조에 대처하는 방안은 클라이언트 시스템 또는 클라이언트 프로그램이 가동시 정상적인 형태의 값을 가지고 있는가, 혹은 인증된 클라이언트가 접속을 하였는가를 판별하는 것이다. 하지만 서버로 전송하는 값을 위조나 변조 하거나, 응답을 허위로 보내는 행위에 대해 대응할 수 있는 방안은 여러 보안형태를 취함에도 불구하고 부족할 수밖에 없는 것이 현실적인 보안시스템에서의 한계점이라 할 수 있다. 이에 클라이언트 시스템 또는 클라이언트 프로그램이 가지고 있는 특정 값 또는 실시간으로 확인할 수 있는 값에 대해 랜덤한 질의 등을 통해서 응답을 받고, 응답을 실행하는 실행코드를 일반적 저장장치인 HDD에서 실행하는 것이 아닌 Memory상의 특정 컨테이너에서 실행 뒤 응답을 하고, 클라이언트 시스템 혹은 클라이언트 프로그램의 종료와 함께 휘발시키는 형태를 취한다면 리버싱이나 해킹에 대응력을 가질 수 있는 구조가 될 것이다. 이를 온라인 게임에서의 클라이언트 프로그램 보안에 적용하는 방안에 대해 모색하도록 한다.

## II. 본론

현재 클라이언트 시스템 또는 클라이언트 프로그램을 확인하는 방법 중 보편적으로 이용되는 형태로는 CRC체크, Hardware 인증, 수신제한시스템(CAS), 공인인증서(PKI) 등이 있으며 이를 응용한 형태의 보안 방식인 카드사의 인증방식이나 하드웨어 형태에서 벗어난 소프트웨어 형태의 다운로드형 수신제한시스템(D-CAS) 등이 있고 이외에도 다양한 형태로 응용이 되고 있다.

### 2.1 응용 분야

온라인 게임에서는 게임 클라이언트의 안정성을 확보하고 다양한 사용자시스템에서 문제가 발생하지 않도록 최적화된 형태의 서비스에 중점을 두고 있다. 온라인 게임 클라이언트 프로그램은 사용자의 컴퓨터에 설치가 되므로 악의적인 목적의 사용자는 자신의 컴퓨터에 설치된 온라인 게임 클라이언트를 다양한 방식으로 해킹 또는 리버싱 할 수 있기에 온라인 게임개발사 또는 서비스사는 다양한 보안대책을 강구하고 있음에도 취약성이 존재할 수 있을 뿐만 아니라, 보안이 고려되지 않은 개발코드(API)와 주요 실행 파일(EXE, DLL)에 대한 Packing등을 대응하지 않는 경우가 발생하는 등 여러 형태의 취약한 구조를 가져가고 있는 상황이다.

이러한 온라인 게임 클라이언트의 취약한 형태는, 최근 온라인 게임을 통한 게임머니, 아이템거래 등의 시장이 수천억 단위에서 조단위로 확장되고, 공장형태처럼 가동되는 속칭 작업장이 개별 단위에서 수백대 단위로 은밀하게 운영되고 있고 개인들 또한 해킹툴 등을 구매하여 이용을 하는 등 게임이라는 특성, 즉 즐기기 위한 게임의 형태가 아닌 온라인 게임을 통하여 금전적 이득을 취하고자 하는 해커집단 및 판매업자들의 이해관계가 얽혀 점차 음성적인

형태로 나아가는 등 즐거움을 제공해야 할 게임이 이제는 하나의 불법적인 모습에서 자유롭지 못하게 되는 상황에 처해 있는 현실이다.

## 2.2 시스템 구조

온라인 게임 클라이언트 프로그램의 보안에 일회용 실행코드(ONE-TIME Execution Code)라는 새로운 방식으로 접근하고자 한다.

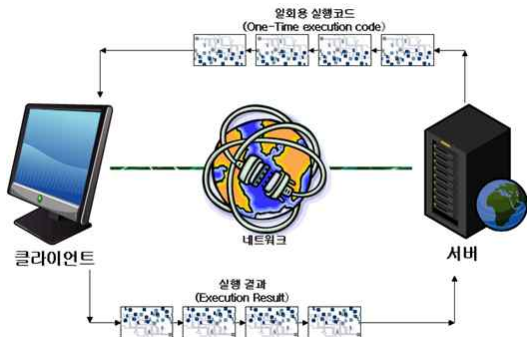


그림 1. 시스템 구조
Fig. 1. System Architecture

일회용 실행코드는 서버와 클라이언트가 일정주기로 통신을 함에 있어 동일한 질의가 아닌 일회성을 가지는 질의를 통하여 클라이언트 프로그램의 해킹 여부를 진단하게 된다.

일회용 실행코드를 실행하게 되는 위치는 클라이언트

프로그램이 할당한 메모리상의 컨테이너 안에서 수행되고, 온라인 게임 클라이언트 프로그램이 실행중인 경우에만 메모리를 할당받아 수행할 수 있기에 온라인 게임 클라이언트 프로그램이 종료되는 것과 동시에 사용자 컴퓨터에 특정한 흔적을 남기지 않고 종료를 할 수 있다.


그림 2. 일회용실행코드 구조
Fig. 2. One-Time Execution Code Architecture

그림2에 표시된 일회용 실행코드는 다음과 같다.

① 암호화 방법 : 일회용 실행코드 서버 API에서는 일회용 실행코드를 암호화하기 위해 21 종류의 암호화 방법을 제공하고 있는데, 이 중 1~20번까지는 일회용 실행코드 서버 API에서 미리 정의된 암호화 방법이며, 마지막 21번째는 사용자가 정의해서 사용하는 암호화 방법이다. 일회용 실행코드 서버가 적용된 서버 프로그램은 랜덤 함수를 이용하여 총 21종류의 암호화 방법 중 하나를 선택하여 사용하게 되는데, 이 때 선택된 암호화 방법을 여기에 기입하게 된다.

② 암호 키 : 선택된 암호화 방법에 따라 일회용 실행코드를 암호화할 때 사용되는 암호 키를 의미한다.

③ GUID : 선택된 일회용 실행코드를 클라이언트로 전송하기 위해 생성된 핸들의 GUID이다. 통상적으로 GUID는 클라이언트로 전송하기 위해 선택된 일회용 실행코드마다 다르게 생성되며, '코드 생성기'에 의해 데이터베이스에서 임의의 일회용 실행코드가 선택됨과 동시에 랜덤하게 결정된다.

④ 현재 블록 No. : 선택된 일회용 실행코드를 클라이언트로 전송하기 위해서는 해당 코드를 전송 가능한 패킷 사이즈 단위로 분할하여야 하는데, 분할된 코드의 순서를 의미한다. 예를 들어 현재 블록 No.가 1이면 현재 블록이 분할된 일회용 실행코드의 첫 번째 블록임을 말하며, 이 숫자는 다음에 설명할 '전체 블록 수'를 넘을 수 없다. 서버가 실행코드 블록을 클라이언트로 전송할 때는 분할된 실행코드 블록 순서로 전송하는 것이 아니라 각 블록을 인터리브하여 전송하게 되므로 클라이언트는 '현재 블록 No.'를 참조하여 수신된 실행코드 블록을 재조합하게 된다.

⑤ 전체 블록 수 : 선택된 일회용 실행코드를 전송 가능한 패킷 사이즈 단위로 분할한 후의 총 블록 개수다. 통상적으로 다음에 설명할 '전체 코드 사이즈'를 기 정의된 전송 가능한 패킷 사이즈로 나누면 '전체 블록 수'를 얻을 수 있으며, 선택된 일회용 실행코드를 클라이언트로 전달하기 위해서는 '전체 블록 수' 만큼의 패킷 전송이 이루어져야 한다.

⑥ 코드 사이즈 : 분할된 일회용 실행코드 블록 중 현재 블록에 포함된 코드 블록의 암호화되기 전 사이즈를 의미한다.

⑦ 암호화된 코드 사이즈 : 분할된 일회용 실행코드 블록 중 현재 블록에 포함된 코드 블록의 암호화된 후 사이즈를 의미하며, 전송 가능한 패킷 사이즈에서 전송에 필요한 해더 정보들의 사이즈를 뺀 값을 넘을 수 없다.

⑧ 전체 코드 사이즈 : 선택된 일회용 실행코드의 전체 사이즈를 의미한다.

⑨ 코드 구분 : 선택된 일회용 실행코드의 용도를 구분하는 값이며, 실행코드 데이터베이스에 정의된 값을 사용한다. 클라이언트는 이 값을 참조하여 해당 실행코드를 메모리 상주형으로 사용할 지, 일회용으로 사용할 지를 결정한다.

⑩ 암호화된 코드 : 분할된 일회용 실행코드 블록 중 현재 블록을 암호화한 코드 블록이며, 이 코드 블록의 사이즈는 '암호화된 코드 사이즈'에 표시된다.

위에서 언급한 일회용 실행코드 서버 API의 구성은 다음과 같이 된다.

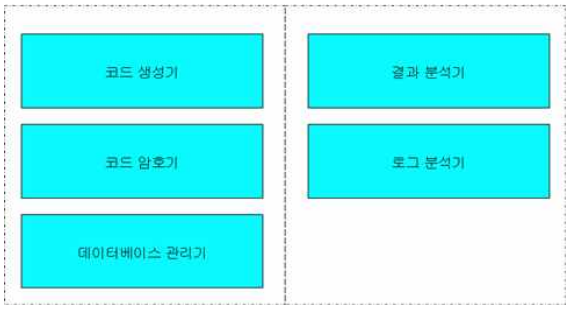


그림 3. 일회용실행코드 서버 API
Fig. 3. One-Time Execution Code Server API

일회용 실행코드 서버 API는 클라이언트로 전송할 일회용 실행코드를 생성하는 '코드 생성기', 클라이언트로 전송된 일회용 실행코드 실행 결과 값을 확인하는 '결과 분석기', 클라이언트로 전송할 일회용 실행코드를 암호화하는 '코드 암호기', 일회용 실행코드 실행 결과 값의 세부 로그 정보를 확인하는 '로그 분석기'로 이루어진다.

'코드 생성기'는 일회용 실행코드 서버가 적용된 서버 프로그램에서 정해진 크기를 갖는 일회용 실행코드의 구조화 데이터를 생성한다.
'코드 암호기'는 일회용 실행코드의 구조화 데이터를 암호화 한다.
'데이터베이스 관리기'는 일회용 실행코드 관련 데이터데이스를 관리한다.
'결과 분석기'는 클라이언트로부터 전달받은 일회용 실행코드의 실행 결과 값을 확인하고 분석한다.
'로그 분석기'는 클라이언트로부터 전달받은 일회용 실행코드의 실행 결과 값의 상세한 로그를 확인한다.

상기와 같이 일회용 실행코드 서버는 클라이언트 프로그램과의 통신상에서 일회성 질의를 송신하고 클라이언트 프로그램은 송신된 질의에 대해 도출된 결과값을 일회용 실행코드 서버에 전송하며 클라이언트 프로그램으로부터 수신된 결과 값이 정당한 가를 판별하는 역할을 한다. 이를 통해 클라이언트 프로그램의 위조와 변조 등의 여부를 판별하여 서버 프로그램에 전달을 하는 역할을 한다.

## 2.3 구현 설명

서버와 클라이언트가 통신을 위한 준비되면 서버에서는 코드 생성기에서 데이터베이스 관리기를 이용하여 일회용 실행코드를 준비한다. 클라이언트와 미리 정의된 패킷 사이즈에 맞추어 블록으로 나눈 후 차례대로 클라이언트로 전송한다. 그리고 서버는 이후 클라이언트가 일회용 실행코드를 수행한 후 응답을 하기 까지 대기상태로 빠진다. 클라이언트는 일회용 실행코드를 차례대로 받는다. 전체 블록의 개수만큼 모두 수신 될 때까지 대기한다. 만약 전체 블록이 모두 수신되면 재조합 과정을 시작한다. 이는 서버에서 패킷단위로 일회용 실행코드를 블록단위로 나누면서 순서를 인터리브하게 수정하기 때문이다. 재조합 과정을 마치고 난 뒤, 메모리 컨테이너에서 일회용 실행코드를 실행한다. 여기에서 일회용 실행코드를 실행한다는 것은 일회용 실행코드 내부에 있는 함수를 실행한다는 의미이다. 일회용 실행코드의 결과는 일반적으로 간단한 문자열이다. 클라이언트는 이 간단한 문자열을 서버로 전송한다. 그리고 다시 서버에서 일회용실행코드가 전송될 때까지 대기상태에 빠진다. 서버는 클라이언트에서 전송된 일회용 실행코드 결과 패킷을 결과 분석기를 통해 클라이언트에 대한 패널티를 적용한다. 만약 결과 분석기에서 정상적인 클라이언트라는 판정이 나오면 클라이언트와 계속해서 연결을 유지하고 클라이언트가 변조되었거나 인티그리티 적합성을 통과하지 못한 결과가 나올 경우 강제로 연결을 끊는 것을 의미한다.

## 2.4 위변조 탐지

윈도우즈 플랫폼에서 구현한 C/S 프로그램에 일회용 실행코드를 적용 및 탑재하고 윈도우즈 플랫폼에서는 일회용 실행코드를 DLL(동적라이브러리) 형태로 제작하여 클라이언트에서는 메모리상에 DLL을 바로 실행할 수 있는 컨테이너를 제작하여 클라이언트 프로그램에 탑재하고 위조 및 변조된 클라이언트 프로그램을 실행하여 각각의 결과값을 얻는다.


그림 4. 클라이언트 위변조 실행
Fig. 4. Client modified

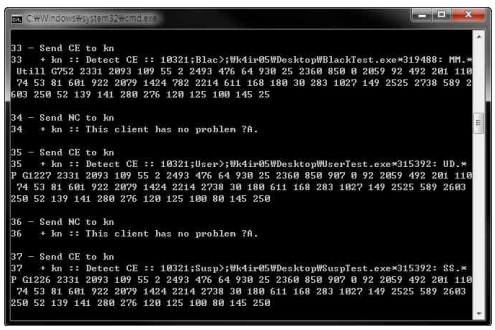


그림 5. 클라이언트 위변조 탐지
Fig. 5.Detect Client modified

## Ⅲ. 결론

서버에서 생성된 일회용 실행코드를 클라이언트 프로그램의 메모리상에서 수행을 한다는 것은 클라이언트 시스템에서의 최소 노출을 확보하고 클라이언트 프로그램 및 실행시점마다 수행되어 도출되는 결과값이 다르다는 것으로, 클라이언트 프로그램이 위조나 변조가 되었는지의 여부를 판단하고 이를 통해 정상적인 클라이언트 시스템 및 클라이언트 프로그램의 해킹여부를 탐지할 수 있다.

최근 게임클라이언트 프로그램의 해킹사례에 비추어 보면 정상적인 인증과 로그인의 절차는 밟으나 게임 클라이언트 프로그램의 UI적 요소들은 모두 제거하고 단순 실행파일 만으로 게임을 자동화 시킬 수 있는 논-클라이언트-봇(Non Client Bot)을 하나의 사용자 컴퓨터에서 수 십여 계정 이상을 가동할 수 있는 상당한 기술들을 확보하여 게임상의 정상 유저의 플레이를 방해하고 게임상에 존재하는 상거래 시스템 등을 손상시켜 게임 개발의 비용을 증대시키고 정상 유저들로 하여금 게임을 이탈하게 만드는 결과를 초래하거나, 게임내의 구조적인 화면 배치를 없애는 방법을 이용하여 벽 뒤에 숨어 있으면 볼 수 없는 상대 유저를 손쉽게 볼 수 있게 만드는 등 게임 클라이언트 프로그램을 해킹하는 상업적 프로그램이 시중에 만연하고 있는 상황을 해결할 수 있는 방안이 될 것이다.

이를 응용하여 게임 클라이언트 프로그램뿐만이 아닌 전자상거래 혹은 전용 클라이언트 프로그램을 이용하는 다수의 C/S 환경에서 일회용 실행코드를 이용하여 클라이언트 시스템 및 클라이언트 프로그램의 위조와 변조를 탐지하고 클라이언트를 인증하는 수단으로 이용한다면 C/S 환경의 서비스를 제공하는 주체에게는 서비스의 신뢰성을 제공 할 수 있을 것이다.

## 참고문헌

[1] 조성언, “온라인 게임에서의 보안 이슈들” 보안공학연구논문지 Vol.4 No.3, 7-14쪽, 2007년 8월

[2] 김동균, “온라인 게임 보안 강화, 게임 산업 분야별 현실과 방향 세미나, 2003.3

[3] 이면재, “게임성 있는 MMORPG 온라인 게임의 고찰”, 한국게임학회지 P280-281, 2004년 1월

[4] http://www.gamesutra.com