게임 해킹툴의 변칙적 공격 기법 분석

CodeEngn

www.CodeEngn.com

2013 CodeEngn Conference 09

박근영

AhnLab

2013.11.30

Contents

기사 자료

AhnLab

종류	메모리 조작	오토플레이	그래픽핵	파일 조작	패킷 조작	스피드핵	기타	합계
개수	519	59	189	2	9	1	16	795

▲ 온라인 게임 해킹툴 관련 집계 자료. <출처=안랩>

온라인 게임 해킹 기법은 날로 지능화되고 있다. 과거에는 DLL 인젝션(실행 중인 프로세스에 특정 DLL 파일을 강제로 삽입하는 것)으로 메모리를 조작하는 단순 기법이 많았다. 그러나 최근 게임 보안 솔루션이 이 기법을 차단하자 게임 프로세스가 아닌 OS 등 다른 영역을 해킹하는 우회 공격 기법이 증가하고 있다. 대표적인 것이 윈도 디스플레이 드라이버 모델(WDDM) 조작, 게임 런처로 위장, 게임 보호 모듈 조작, 게임 내 스크립트 호출 등이다.

안랩 박근영 선임연구원은 "최근에는 게임 프로세스 자체가 아닌 PC 환경에 전반적으로 영향을 미치는 영역을 해킹하는 툴이 다수 제작된다"며 "단순 호기심으로 실행했다가는 포맷을 해야 할 수도 있기 때문에 각별한 주의가 필요하다"고 당부했다.

실험 대상

TargetFPS.exe

[공격 기법 1] 게임 코드 조작 By OpenProcess

AhnLab

TargetFPS.exe

```
004071DE  MOV     EDX, DWORD PTR SS:[EBP-4]
004071E1  MOV     EAX, DWORD PTR DS:[EDX+88C]
004071E7  NOP NOP NOP
004071EA  MOV     ECX, DWORD PTR SS:[EBP-4]
004071ED  MOV     DWORD PTR DS:[ECX+88C], EAX
004071F3  MOV     EDX, DWORD PTR SS:[EBP-4]
```

CodeModify.exe

코드 조작 by OpenProcess

- hProcess =OpenProcess(...,TargetFPS_PID)
- VirtualProtectEx(hProcess, 0x4071e7, 0x3, PAGE_EXECUTE_READWRITE , ..)
- szPatchedByte[0] = 0x90;
  szPatchedByte[1] = 0x90;
  szPatchedByte[2] = 0x90;

  WriteProcessMemory(hProcess, 0x4071e7, szPatchedByte, 3, ..)

[방어 기법 1] OpenProcess 접근 방어

AhnLab

TargetFPS.exe

Protector_dll.dll

OpenProcess 접근 방어

```
IF (OpenProcess 대상 PID==TargetFPS PID)
   IF (접근 시도 Process != OS Process)
      return FAIL
```

CodeModify.exe

코드 조작 by OpenProcess

```
hProcess =OpenProcess(...,TargetFPS_PID)

VirtualProtectEx(hProcess, 0x4071e7, 0x3,
PAGE_EXECUTE_READWRITE , ..)

szPatchedByte[0] = 0x90;
szPatchedByte[1] = 0x90;
szPatchedByte[2] = 0x90;

WriteProcessMemory(hProcess,
0x4071e7, szPatchedByte, 3, ..)
```

[공격 기법 2] 게임 코드 조작 By BrainSwitching

AhnLab

TargetFPS.exe

Protector_dll.dll

OpenProcess
접근 방어

IF (OpenProcess 대상 PID==TargetFPS PID)
IF (접근 시도 Process != OS Process)
return FAIL

CodeModify.exe

BrainSwitching.exe

코드 조작 by OpenProcess

[공격 기법 2] 게임 코드 조작 By BrainSwitching

AhnLab

ResumeThread(TaskMgr_PI->hThread)

CodeModify.exe

BrainSwitching.exe

TaskMgr.exe
(Process)

- header
- .text
- .rdata
- .data
- .rsrc

CodeModify.exe
(File-base)

- header
- .text
- .rdata
- .data
- .rsrc

1. CreateProcess(..)
2. NtDll ! UnmapViewOfSection(..)
3. VirtualAllocEx(..)
4. WriteProcessMemory(..)
5. WriteProcessMemory(..)
6. GetThreadContext(..)
7. Ctx.Eax = CodeModify.이미지
8. SetThreadContext(..)
9. ResumeThread(..)

게임 코드 조작 By DLL-Injection

AhnLab

TargetFPS.exe

```
004071DE  MOV     EDX, DWORD PTR SS:[EBP-4]
004071E1  MOV     EAX, DWORD PTR DS:[EDX+88C]
004071E7  NOP NOP NOP
004071EA  MOV     ECX, DWORD PTR SS:[EBP-4]
004071ED  MOV     DWORD PTR DS:[ECX+88C], EAX
004071F3  MOV     EDX, DWORD PTR SS:[EBP-4]
```

PID)

IF (접근 시도 Process != OS Process)
return FAIL

CodeModify.exe

BrainSwitching.exe

코드 조작 by OpenProcess

DLL_Injector.exe

CodeModify_dll.dll

코드 조작 by DLL-인젝션

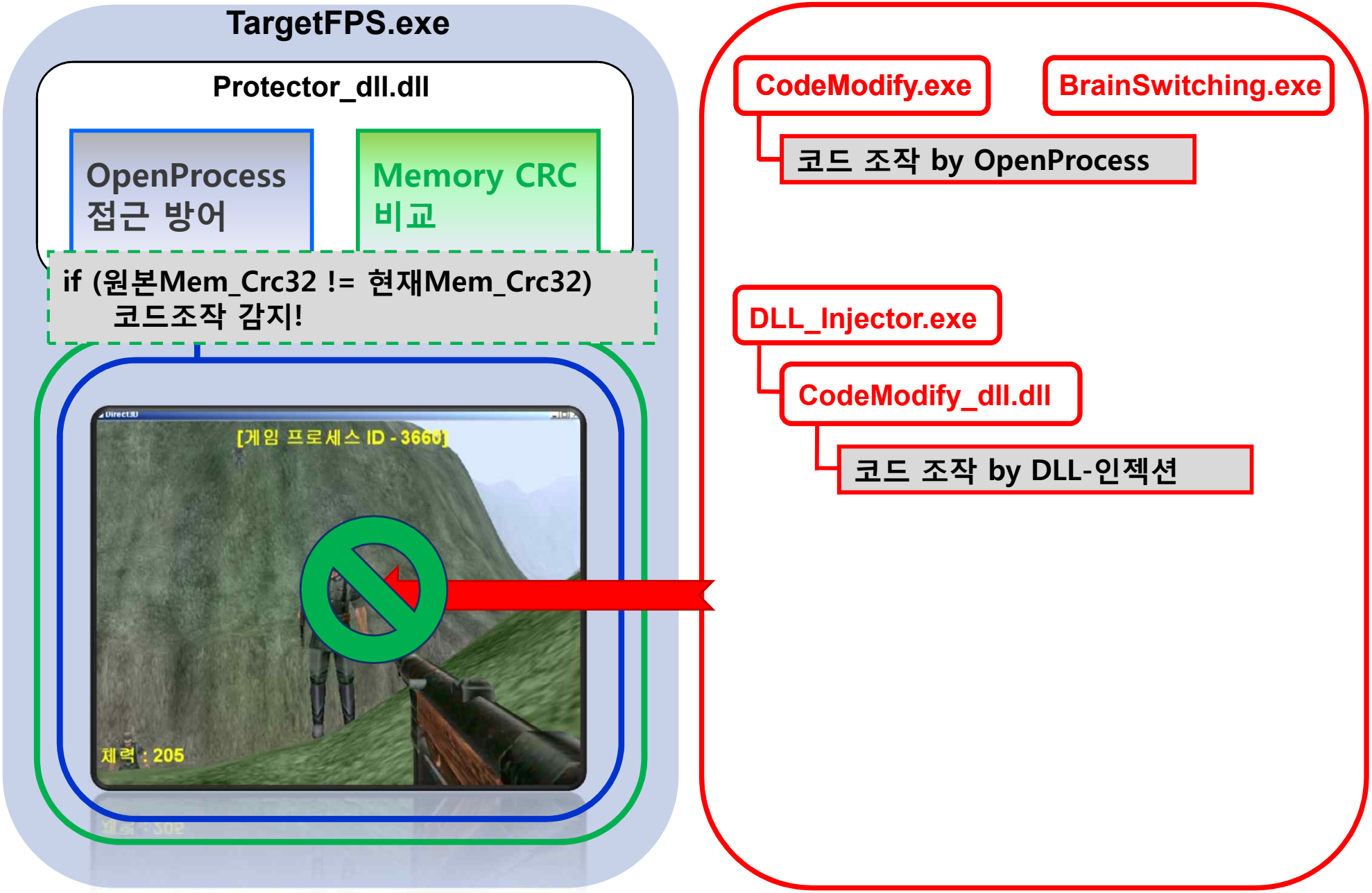
```
VirtualProtect(0x4071e7, 0x3,
PAGE_EXECUTE_READWRITE , ..)

szPatchedByte[0] = 0x90;
szPatchedByte[1] = 0x90;
szPatchedByte[2] = 0x90;

memcpy((LPVOID) 0x4071e7,
szPatchedCodes, 3);
```

[방어 기법 2] 메모리 CRC 비교

AhnLab



[공격 기법 4] 게임 코드 조작 By DR 조작

AhnLab

CodeModify.exe

BrainSwitching.exe

코드 조작 by OpenProcess

DLL_Injector.exe

CodeModify_dll.dll

코드 조작 by DLL-인젝션

DR_Modify_dll.dll

코드 조작 by DR 조작

[공격 기법 4] 게임 코드 조작 By DR 조작

AhnLab

```
LONG __stdcall DebugHookHandler(PEXCEPTION_POINTERS ExceptionInfo)
{   …
    if(ExceptionInfo->예외발생주소 == 0x4071e7)
    {
        ExceptionInfo->ContextRecord->Eip = 0x004071e7 + 0x3; Return..;
    } …
}
```

TargetFPS.exe

DR_Modify_dll.dll

코드 조작 by DR 조작

- AddVectoredExceptionHandler(..)
- hThread = OpenThread(..)
- SuspendThread(hThread)
- ctx.ContextFlags= CONTEXT_DEBUG_REGISTERS;
  ctx.Dr0 = 0x4071e7;
  ctx.Dr7 |= 0x00000001;
- SetThreadContext(hThread,&ctx);
- ResumeThread(hThread);

[공격 기법 4] 게임 코드 조작 By DR 조작

AhnLab

```
Kernel32 ! AddVectoredExceptionHandler(0, DebugHookHandler)
```

```
hThread = OpenThread( THREAD_SUSPEND_RESUME | THREAD_GET_CONTEXT |
THREAD_SET_CONTEXT |THREAD_QUERY_INFORMATION, FALSE, TargetThreadID  )
```

```
SuspendThread( hThread )
```

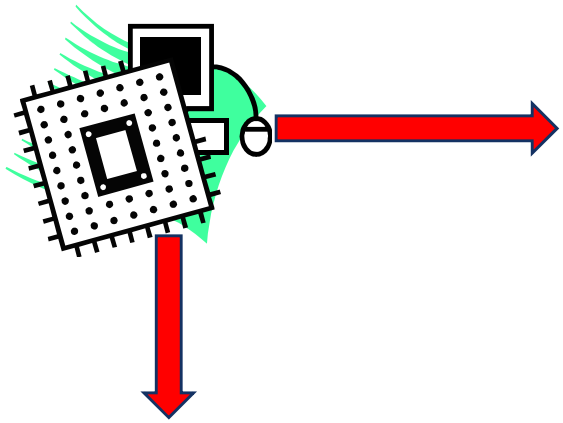
```
ctx.ContextFlags = CONTEXT_DEBUG_REGISTERS;
ctx.Dr0 = (DWORD) 0x004071e7
ctx.Dr7 |= 0x00000001
```

```
SetThreadContext( hThread, &ctx )
ResumeThread( hThread )
```

```
LONG __stdcall DebugHookHandler(PEXCEPTION_POINTERS ExceptionInfo)
{   …
    if(ExceptionInfo->예외발생주소 == 0x4071e7)
    {
        ExceptionInfo->ContextRecord->Eip = 0x004071e7 + 0x3; Return..;
    } …
}
```

Debug Register

DR0 = BP 주소



```
int main(int argc, char* argv[])
{
    printf("Hello World!\n");
    return 0;
}
```

인터럽트 1

[예외 핸들러]

인터럽트 발생 당시의 CPU 레지스터 정보를 갖고 있음

Debug Register

- Debug Register: 디버깅 목적으로 프로세서에서 사용하는 레지스터
  - DR0 ~ DR3
    - BreakPoint를 설정할 주소를 저장
  - DR6
    - 디버그 상태 레지스터
  - DR7
    - 디버그 제어 레지스터 : breakpoint 상태를 선택적으로 활성화/비활성화

조작 기법별 테크맵

AhnLab

CodeModify.exe

코드 조작 by OpenProcess

BrainSwitching.exe

DLL_Injector.exe

CodeModify_dll.dll

코드 조작 by DLL-인젝션

DR_Modify_dll.dll

코드 조작 by DR 조작

Thank you

Q&A