

스타크래프트 맵핵 제작하기

DLL Injection을 이용한!

린츠
q_c@naver.com

Content

1. 목적	1
2. 맵핵이란 무엇이며 준비해야할 것들	2
3. 프로그램으로 만들자!	11
4. 실험 및 결과	19
참고문헌	22

1. 목적

DLL Injection을 공부하면서 지뢰 찾기 맵핵을 제작해볼 기회가 생겼는데, 그와 비슷한 방법으로 스타크래프트 맵핵도 가능할 것 같아 도전해 보았고, 엄청난 삽질을 경험한 끝에 그렇게 단순한 방법으로는 블리자드라는 큰 벽을 넘을 수 없다는 것을 깨달았으나 포기하지 않고 계속해서 삽질에 매진한 끝에 한 지인의 도움으로 스타크래프트 맵핵을, 부족하지만 구현하는데 성공하였다. 그리고 3일도 안되서 다 잊어먹는 망할 기억력을 떠올리며 혹시나 잊어버리더라도 나중에 이 문서보고 참고할 수 있도록 대략적인 방법을 서술하게 되었음. 이 문서는 개인의 공부를 위한 목적으로 작성되었으며, 이를 악용할 시에 모든 책임은 당사자에게 있다. 아무튼 이 문서를 작성할 당시에는 스타크래프트 1.16버전이었고, 시간이 지난 현재 배틀넷에서는 전혀 사용이 불가능하기 때문에, 날림으로 작성된 부족한 문서를 공개해본다.

2. 맵핵이란 무엇이며 준비해야 할 것들

맵핵이라는 것을 모르는 사람이라면 아마 스타크 한번 안해보신 꼬꼬마이시거나 너무 정직하게 게임에 임하여 부정한 방법 따위 나와 전혀 관계 없다라는 생각을 가지신 분일 것이다. 따라서 맵핵에 대해서 간략히 설명한다.

맵핵이란 말 그대로 맵을 전부다 보여주는 프로그램을 말한다. 이를테면 김린츄씨와 바렘봉이 신나게 스타크래프트를 하고 있는데, 정직한 바렘봉이 김린츄씨 몰래 리버를 셔틀에 태워서 빙글빙글 돌아 기지 뒤에서 슬쩍 내려놓으려고 할 때 김린츄씨가 맵핵이라면 이미 밝혀진 맵에 바렘봉의 셔틀이 몰래몰래 접근하는 것을 파악하고 궁병이 리버씨가 내리자마자 다크로 슬렁 슬렁 쏘어주시는 시추에이션이 나오게 만드는 것이 가능하다. 펍이나 이상한 예를 들어서 미안하게 생각하나 본인은 이 문서를 공개할 생각이 없으므로 웃으면서 넘어가주길 바란다. (바렘 바 지못미 >ㅁ<)



그림 1 스타크래프트 맵핵

자, 이제 맵핵이 무엇이라는 것도 알았고, 본격적으로 맵핵을 만들기 위해 준비해야 할 것들을 알아보자. 우선 분석을 해야한다. 스타크래프트라는 게임이 어떻게 돌아가고 있는지 파악하기 위해 리버스 엔지니어링을 해야할 필요가 있다. 이때 필요한 것이 올리디버거이며, 아이이다. 물론 스타크래프트 전체를 분석한다는 것은 아니고 맵을 그리는 부분만을 중점적으로 보면 된다. 사실 리버싱을 하면서 코드를 수정해야한다. 가장 중요한 작업은 여기서 다 하는 편이고, 나중에 어떤 코드를 어떻게 수정해야 되는지 알아내게 되면 DLL Injection을 이용해 프로그램을 만들 것이다. 물론 다른 여러 방법이 있을 수 있겠지만 적어도 김린츄씨는 이런 방법을 사용했다. 잊어버릴뻔 했는데 가장 중요한 스타크래프트를 빼먹지 말자.

간단한 요약!

올리디버거, 아이다Pro(hex-ray), 스타크래프트 1.16

3. 맵핵을 위한 리버싱

이제 리버싱을 시작해보자. 필자는 하나하나 코드를 읽어가며 정확하게 분석해가며 이 루틴은 여기 사용하고 저 루틴은 저기 사용하는구나!라고 말할 수 있는 능력이 되지 않기 때문에, 약간 추상적으로 말하자면 하나의 실마리를 찾고 이것을 이용해 다음 실마리를 찾아 의심가는 코드를 수정해보고 안되면 이건 아닌값다.. 하는 비상한 방법을 사용했다. 물론 시간은 엄청 걸리고 무한 삽질로 인해 땅을 많이 파야하는 힘든 작업이다. 다음은 위에서 언급한 방법을 바탕으로 스타크래프트 맵핵을 제작하는 과정(리버싱)을 다룰 것이다. 나는 그때위 귀찮은 작업 안해도 일일이 분석하면서 다 잘할 수 있어!라면 그냥 이 문서 끄고 앳싸리 맵핵 열심히 만들면 된다. 스타크래프트 버전에 따라 주소들이 달라질 수 있으니 주의하기 바람.

가장 먼저 볼 것은 sub_4800F0()라는 함수이다. 이 함수는 스타크래프트의 지형을 그려주는 역할을 하는데, 이 함수 내부의 LABEL_20이라는 루틴에서 보듯이 v3변수에 v2의 값을 가지고 다음 루틴으로 점프하게 되면 모든 지형이 그려지는 것을 확인할 수 있다(미니맵은 따로임). 다만 미네랄이나 적의 건물들은 아직 보여지지 않는 것을 볼 수 있다. 하지만 이 실마리를 바탕으로 하나하나 해결할 수 있다. 아래코드를 보고 그 실마리를 찾아보자.

```
v4 = v18;
v3 = v17;
v5 = (unsigned __int16) dword_57F1B0 - 1;
v22 = 24;
while ( 1 )
{
    if ( !dword_6D0EF4 )
    {
        if ( dword_57EE98 & *( _DWORD *)v4 )
        {
            *( _BYTE *)v3 = 0;
            goto LABEL_21;
        }
        if ( dword_57F090 & *( _DWORD *)v4 )
        {
            *( _BYTE *)v3 = v19;
            goto LABEL_21;
        }
    }
LABEL_20:
    *( _BYTE *)v3 = v2;
    goto LABEL_21;
}
if ( dword_6D0EFC )
    goto LABEL_20;
v11 = 0;
BYTE1(v11) = dword_6D0EF8;
v6 = ~*( _DWORD *)v4;
```

그림 2 sub_4800F0

LABEL_20 루틴으로 무조건 점프하게 만들었을 때 모든 지형을 볼 수 있다. 그렇다면

LABEL_20으로 가기 위한 조건이 맵을 밝히는 열쇠가 될 수 있지 않을까 라는 생각을 했다면 Great! 바로 그거다. 우리가 찾아야 할 것은 위 조건문에서 보듯이 dword_6D0EF4의 값과 dword_6D0EFC가 1임을 비교하는 루틴이다. 유닛을 그리는 루틴이나, 미네랄을 그리는 루틴에서도 위와 마찬가지로 두 변수의 값을 확인함으로써 숨겨진 지형지물을 그릴지 말지 결정 할테니 말이다. 우선 무조건 LABEL_20으로 점프하도록 코드를 수정하도록 하자.

```

00480178 85C9 TEST ECX,ECX
0048017A 8A15 F80E6D00 MOV DL,BYTE PTR DS:[6D0EF8]
00480180 74 30 JE SHORT StarCraf.004801B2
00480182 8B00 FC0E6D00 MOV ECX,DWORD PTR DS:[6D0EFC]
00480188 85C9 TEST ECX,ECX
0048018A 75 48 JNZ SHORT StarCraf.004801D4
0048018C 8B0F MOV ECX,DWORD PTR DS:[EDI]
0048018E 33DB XOR EBX,EBX
00480190 8AFA MOV BH,DL
00480192 F7D1 NOT ECX

```

그림 3 수정 전 코드

```

00480155 0FB735 B0F15701 MOVZX ESI,WORD PTR DS:[57F1B0]
0048015C 8B7D F8 MOV EDI,DWORD PTR SS:[EBP-8]
0048015F 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C]
00480162 4E DEC ESI
00480163 C745 F0 180000 MOV DWORD PTR SS:[EBP-10],18
0048016A EB 06 JMP SHORT StarCraf.00480172
0048016C 8A1D 7C7A6500 MOV BL,BYTE PTR DS:[657A7C]
00480172 8B00 F40E6D00 MOV ECX,DWORD PTR DS:[6D0EF4]
00480178 85C9 TEST ECX,ECX
0048017A 8A15 F80E6D00 MOV DL,BYTE PTR DS:[6D0EF8]
00480180 90 NOP
00480181 90 NOP
00480182 8B00 FC0E6D00 MOV ECX,DWORD PTR DS:[6D0EFC]
00480188 85C9 TEST ECX,ECX
0048018A EB 48 JMP SHORT StarCraf.004801D4
0048018C 8B0F MOV ECX,DWORD PTR DS:[EDI]
0048018E 33DB XOR EBX,EBX
00480190 8AFA MOV BH,DL
00480192 F7D1 NOT ECX
00480194 85D9 TEST ECX,EBX
00480196 75 05 JNZ SHORT StarCraf.0048019D
00480198 C600 00 MOV BYTE PTR DS:[EAX],0
0048019B EB 39 JMP SHORT StarCraf.004801D6
0048019D 84CA TEST DL,CL
0048019F 75 07 JNZ SHORT StarCraf.004801A8
004801A1 8A4D FF MOV CL,BYTE PTR SS:[EBP-1]
004801A4 8B08 MOV BYTE PTR DS:[EAX],CL
004801A6 8B08 MOV BYTE PTR DS:[EAX],CL

```

그림 4 지형 모두 보이도록 패치

480180에 있는 점프 문은 dword_6D0EF4의 조건을 따르고 있고, 아래 48018A는 dword_6D0EFC의 조건을 따르고 있다. 수정이 완료되었으면 이 실마리를 가지고 다음을 찾아가자. 올리디버거의 경우 Search for의 Constant를 사용하도록 하자. CTRL+L을 눌러 적당히 의심가는 부분까지 가도록 하자.

sub_4981C0함수에 도착해보면 dword_6D0EF4의 값과 dword_6D0EFC의 값을 비교하는 루틴이 보인다. 이 부분이 바로 본 화면의 유닛 및 건물 등을 그려주는 루틴이다. 조건이 맞을 경우 sub_42D4C0()함수를 실행하도록 하고 있다. 코드를 적당히 수정함으로써 무조건 sub_42D4B0()함수가 실행될 수 있도록 하자. 이 부분을 영똥하게 수정하면 모든 유닛이 보이지 않도록 할 수도 있다. 린츄씨의 경우 드론으로 마린을 상대하다가 갑자기 유닛이 전부다 사라지더니 마린 총소리가 들리지만 불쌍한 드론만 저세상으로 갔다는 의미없는 이야기.

```

dword_6C4A0C = 0;
if ( dword_6D0EF4 )
{
    v5 = a1;
    if ( a1 <= a2 )
    {
        v6 = dword_6D0EF8;
        do
        {
            for ( i = dword_629668[v5]; i; i = *(_DWORD*)(i + 4) )
            {
                if ( v6 & *(_BYTE*)(i + 12) || dword_6D0EFC && *(_BYT
                    sub_42D4B0());
            }
            ++v5;
        }
        while ( v5 <= a2 );
    }
}

```

그림 5 sub_4981C0

004981D4	A1 F40E6D00	MOV EAX,DWORD PTR DS:[6D0EF4]
004981D9	85C0	TEST EAX,EAX
004981DB	C705 0C4A6C00	MOV DWORD PTR DS:[6C4A0C],0
004981E5	75 34	JNZ SHORT StarCraf.0049821B
004981E7	8B7D 08	MOV EDI,DWORD PTR SS:[EBP+8]
004981EA	3B7D 0C	CMP EDI,DWORD PTR SS:[EBP+C]
004981ED	77 79	JA SHORT StarCraf.00498268
004981EF	8A1D 90F05700	MOV BL,BYTE PTR DS:[57F090]
004981F5	8B34BD 68966200	MOV ESI,DWORD PTR DS:[EDI*4+629668]
004981FC	85F6	TEST ESI,ESI
004981FE	74 11	JE SHORT StarCraf.00498211
00498200	845E 0C	TEST BYTE PTR DS:[ESI+C],BL
00498203	74 05	JE SHORT StarCraf.0049820A
00498205	E8 A652F9FF	CALL StarCraf.0042D4B0
0049820A	8B76 04	MOV ESI,DWORD PTR DS:[ESI+4]

그림 6 수정 전 코드

4981E5와 498203에 있는 점프문들을 NOP처리 한다

004981BF	CC	INT3
004981C0	55	PUSH EBP
004981C1	8BEC	MOV EBP,ESP
004981C3	53	PUSH EBX
004981C4	56	PUSH ESI
004981C5	57	PUSH EDI
004981C6	33C0	XOR EAX,EAX
004981C8	B9 40000000	MOV ECX,40
004981CD	BF 709C6200	MOV EDI,StarCraf.00629C70
004981D2	F3:AB	REP STOS DWORD PTR ES:[EDI]
004981D4	A1 F40E6D00	MOV EAX,DWORD PTR DS:[6D0EF4]
004981D9	85C0	TEST EAX,EAX
004981DB	C705 0C4A6C00	MOV DWORD PTR DS:[6C4A0C],0
004981E5	90	NOP
004981E6	90	NOP
004981E7	8B7D 08	MOV EDI,DWORD PTR SS:[EBP+8]
004981EA	3B7D 0C	CMP EDI,DWORD PTR SS:[EBP+C]
004981ED	77 79	JA SHORT StarCraf.00498268
004981EF	8A1D 90F05700	MOV BL,BYTE PTR DS:[57F090]
004981F5	8B34BD 68966200	MOV ESI,DWORD PTR DS:[EDI*4+629668]
004981FC	85F6	TEST ESI,ESI
004981FE	74 11	JE SHORT StarCraf.00498211
00498200	845E 0C	TEST BYTE PTR DS:[ESI+C],BL
00498203	90	NOP
00498204	90	NOP
00498205	E8 A652F9FF	CALL StarCraf.0042D4B0
0049820A	8B76 04	MOV ESI,DWORD PTR DS:[ESI+4]
0049820D	85F6	TEST ESI,ESI
0049820F	75 EF	JNZ SHORT StarCraf.00498200
00498211	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
00498214	47	INC EDI

그림 7 유닛 및 건물 보이도록 패치

여기까지 했다면 비록 미니맵은 보이지 않더라도 적 기지의 유닛 및 건물들을 볼 수 있다. 재미있는 건 직접 탐색하기 이전까지는 저그의 바닥에 깔려 있는 클립이 보이지 않는다.



그림 8 적 저그의 기지

이번에 찾은 sub_46F840()함수는 원래는 보이지 않아야 할 적 유닛들을 선택할 수 있도록 만들 수 있다. 지금까지의 과정만으로는 적기지가 보여도 위 그림의 해처리처럼 선택이 되지 않는다.

```

if ( dword_6D0EF4 )
{
    if ( !((_BYTE)dword_6D0EF8 & v4) && !dword_6D0EFC )
        goto LABEL_30;
    v18 = *(_DWORD*)(v3 + 220);
    if ( BYTE1(v18) & 3 && !(dword_6D0EF8 & *(_DWORD*)(v3 + 228)) )
    {
        |
        v6 = dword_6D0EFC == 0;
        goto LABEL_13;
    }
}
else
{
    if ( !((_BYTE)dword_57F090 & v4) )
        goto LABEL_30;
    v19 = *(_DWORD*)(v3 + 220);
    if ( BYTE1(v19) & 3 )
    {
        v6 = (dword_57F090 & *(_DWORD*)(v3 + 228)) == 0;
    }
}

```

그림 9 sub_46F840()

하지만 이번 함수는 특이하게도 6D0EF4의 값이 무조건 참인 것처럼 점프문을 수정하면 오히려 시야가 밝혀져 있는 우리 기지의 유닛들을 클릭할 수 없는 사태가 발생한다. 그렇기 때문에 반대로 else에 있는 조건들에 대해서 삽질을 해야 한다. 땅이 어느정도 파졌다면 dword_57F090의 조건에 대한 점프구문을 수정해야 된다는 것을 깨달을 것이다.

```

0046F8B2 8586 E4000000 TEST DWORD PTR DS:[ESI+E4],EAX
0046F8B8 75 2D UNZ SHORT StarCraf.0046F8E7
0046F8BA A1 FC0E6D00 MOV EAX,DWORD PTR DS:[6D0EFC]
0046F8BF 85C0 TEST EAX,EAX
0046F8C1 EB 1E JMP SHORT StarCraf.0046F8E1
0046F8C3 A1 90F05700 MOV EAX,DWORD PTR DS:[57F090]
0046F8C8 84C8 TEST AL,CL
0046F8CA 74 0C JE SHORT StarCraf.0046F9D5
0046F8D0 8B8E DC000000 MOV ECX,DWORD PTR DS:[ESI+DC]
0046F8D6 F6C5 03 TEST CH,3
0046F8D9 74 0C JE SHORT StarCraf.0046F8E7
0046F8DB 8586 E4000000 TEST DWORD PTR DS:[ESI+E4],EAX
0046F8E1 74 0C JE StarCraf.0046F9D5
0046F8E7 66:8B46 64 MOV AX,WORD PTR DS:[ESI+64]
0046F8EB E8 30F9FFFF CALL StarCraf.0046F220
0046F8F0 85C0 TEST EAX,EAX
0046F8F2 75 2D UNZ StarCraf.0046F9D5

```

그림 10 수정 전 코드

46F8CA에 있는 점프 구문을 모두 NOP처리 하도록 하자.

```

0046F881 A1 F40E6D00 MOV EAX,DWORD PTR DS:[6D0EF4]
0046F886 85C0 TEST EAX,EAX
0046F888 8B7E 0C MOV EDI,DWORD PTR DS:[ESI+C]
0046F88B 8A4F 0C MOV CL,BYTE PTR DS:[EDI+C]
0046F88E 74 33 JE SHORT StarCraf.0046F8C3
0046F890 A1 F80E6D00 MOV EAX,DWORD PTR DS:[6D0EF8]
0046F895 84C8 TEST AL,CL
0046F897 75 0E UNZ SHORT StarCraf.0046F8A7
0046F899 8B8D FC0E6D00 MOV ECX,DWORD PTR DS:[6D0EFC]
0046F89F 85C9 TEST ECX,ECX
0046F8A1 74 33 JE StarCraf.0046F9D5
0046F8A7 8B8E DC000000 MOV ECX,DWORD PTR DS:[ESI+DC]
0046F8AD F6C5 03 TEST CH,3
0046F8B0 74 35 JE SHORT StarCraf.0046F8E7
0046F8B2 8586 E4000000 TEST DWORD PTR DS:[ESI+E4],EAX
0046F8B8 75 2D UNZ SHORT StarCraf.0046F8E7
0046F8BA A1 FC0E6D00 MOV EAX,DWORD PTR DS:[6D0EFC]
0046F8BF 85C0 TEST EAX,EAX
0046F8C1 EB 1E JMP SHORT StarCraf.0046F8E1
0046F8C3 A1 90F05700 MOV EAX,DWORD PTR DS:[57F090]
0046F8C8 84C8 TEST AL,CL
0046F8CA 90 NOP
0046F8CB 90 NOP
0046F8CC 90 NOP
0046F8CD 90 NOP
0046F8CE 90 NOP
0046F8CF 90 NOP
0046F8D0 8B8E DC000000 MOV ECX,DWORD PTR DS:[ESI+DC]
0046F8D6 F6C5 03 TEST CH,3

```

그림 11 시야 외 선택 가능(수정 후)

이제 미니맵 부분을 수정하도록 하자. 위와 비슷한 방법으로 계속 진행하면 된다. 문서 쓰다가 퇴근시간이 다되어서 이제부터는 그냥 날림으로 쓸테니 이해하기 바란다. 아래 sub_4A3A20()함수를 보면 dword_6D0EF4와 dword_6D0EFC의 조건이 걸려있는 것을 볼 수 있다. 물론 무조건 실행되도록 점프문을 수정하자. 참고로 이 부분은 미니맵의 지형을 그리는 함수이다.

```

if ( result )
{
    v8 = (unsigned __int16) dword_59CC54 * (unsigned __int16) dword_59
do
{
    v3 = (dword_57F090 & *(_DWORD *)v2) == 0;
    v4 = (dword_57EE98 & *(_DWORD *)v2) == 0;
    if ( dword_6D0EF4 )
    {
        v4 = 0;
        if ( dword_6D0EFC )
            goto LABEL_10;
        BYTE1(v4) = dword_6D0EF8;
        v5 = ~*(_DWORD *)v2;
        if ( v5 & v4 )
            ;
    }
}
}

```

그림 12 sub_4A3A20

4A4A88의 점프문을 NOP처리 한 후에 4A3A93의 점프문을 무조건 점프하도록 수정한다.

004A3A86	85DB	TEST EBX,EBX
004A3A88	74 32	JE SHORT StarCraf.004A3ABC
004A3A8A	A1 FC0E6D00	MOV EAX,DWORD PTR DS:[6D0EFC]
004A3A8F	33C9	XOR ECX,ECX
004A3A91	85C0	TEST EAX,EAX
004A3A93	75 22	JNZ SHORT StarCraf.004A3AB7
004A3A95	A0 F80E6D00	MOV AL,BYTE PTR DS:[6D0EF8]
004A3A9A	8AE8	MOV CH,AL
004A3A9C	F7D2	NOT EDX

그림 13 수정 전 코드

004A3A60	8B16	MOV EDX,DWORD PTR DS:[ESI]
004A3A62	8B0D 90F05700	MOV ECX,DWORD PTR DS:[57F090]
004A3A68	8B1D 98EE5700	MOV EBX,DWORD PTR DS:[57EE98]
004A3A6E	8BC2	MOV EAX,EDX
004A3A70	23C1	AND EAX,ECX
004A3A72	F7D8	NEG EAX
004A3A74	1BC0	SBB EAX,EAX
004A3A76	8BCA	MOV ECX,EDX
004A3A78	23C8	AND ECX,EBX
004A3A7A	8B1D F40E6D00	MOV EBX,DWORD PTR DS:[6D0EF4]
004A3A80	40	INC EAX
004A3A81	F7D9	NEG ECX
004A3A83	1BC9	SBB ECX,ECX
004A3A85	41	INC ECX
004A3A86	85DB	TEST EBX,EBX
004A3A88	90	NOP
004A3A89	90	NOP
004A3A8A	A1 FC0E6D00	MOV EAX,DWORD PTR DS:[6D0EFC]
004A3A8F	33C9	XOR ECX,ECX
004A3A91	85C0	TEST EAX,EAX
004A3A93	EB 22	JMP SHORT StarCraf.004A3AB7
004A3A95	A0 F80E6D00	MOV AL,BYTE PTR DS:[6D0EF8]
004A3A9A	8AE8	MOV CH,AL
004A3A9C	F7D2	NOT EDX
004A3A9E	85C0	TEST EAX,ECX

그림 14 지형 미니맵 수정 후

미니맵을 확인해보면 지형은 전부 보이지만 유닛이나 미네랄 등은 하나도 보이지 않음을 확인 할 수 있다. 유닛과 미네랄 함수는 서로 분리되어 있지만 구조가 매우 비슷하다. dword_6D0EFC를 찾아다니다 보면 sub_4A4540()함수에 도착할 것이다. 여기가 바로 미니맵에 미네랄을 표시해주는 루틴이다. (PS. 비슷한 루틴이 여럿 있는데 두 번째이다.)

```

{
  if ( (_BYTE)v1 & (_BYTE)dword_6D0EF8 )
    break;
  LOBYTE(v1) = dword_6D0EFC;
  v6 = dword_6D0EFC == 0;
}
if ( !v6 )
  break;
LABEL_26:
v3 = *(_DWORD *)(v3 + 108);
if ( !v3 )
{
  dword_59C188 = v4;
  return v1;
}
}
if ( v5 < 0xB0u || v5 > 0xB2u )
  v5 != 188;
LOBYTE(v1) = sub_4A3EC0(
  *(_WORD *)(*(_DWORD *)(v3 + 12) + 22),
  LOBYTE(dword_662840[*(_WORD *)(v3 + 100)]),
  BYTE2(dword_662840[*(_WORD *)(v3 + 100)]),
  LOBYTE(dword_664060[*(_WORD *)(v3 + 100)] & 1);

```

위 코드를 보면 약간 복잡하긴 하지만 dword_6D0EFC가 1이라면 break하고 있다. 저 루틴은 반복문으로 이루어져 있는데, break하지 않으면 sub_4A3EC0()함수를 실행시키지 않고 return 하며 break하면 sub_4A3EC0함수를 실행시키게 된다. sub_4A3EC0함수는 미네랄을 미니맵에 그린다. 따라서 무조건 break해서 빠져나가도록 점프문을 수정한다.

004A4619	EB 06	JMP SHORT StarCraf.004A4621
004A461B	8505 90F05700	TEST DWORD PTR DS:[57F090],EAX
004A4621	74 58	JE SHORT StarCraf.004A467B
004A4623	66:81F9 B000	CMP CX,0B0
004A4628	72 07	JB SHORT StarCraf.004A4631
004A462A	66:81F9 B200	CMP CX,0B2

그림 16 수정 전 코드

4A4621의 점프문을 NOP처리하자.

004A45FC	85D2	TEST EDX,EDX
004A45FE	75 23	JNZ SHORT StarCraf.004A4623
004A4600	8B15 F40E6D00	MOV EDX,DWORD PTR DS:[6D0EF4]
004A4606	85D2	TEST EDX,EDX
004A4608	74 11	JE SHORT StarCraf.004A461B
004A460A	8405 F80E6D00	TEST BYTE PTR DS:[6D0EF8],AL
004A4610	75 11	JNZ SHORT StarCraf.004A4623
004A4612	A1 FC0E6D00	MOV EAX,DWORD PTR DS:[6D0EFC]
004A4617	85C0	TEST EAX,EAX
004A4619	EB 06	JMP SHORT StarCraf.004A4621
004A461B	8505 90F05700	TEST DWORD PTR DS:[57F090],EAX
004A4621	90	NOP
004A4622	90	NOP
004A4623	66:81F9 B000	CMP CX,0B0
004A4628	72 07	JB SHORT StarCraf.004A4631
004A462A	66:81F9 B200	CMP CX,0B2
004A462F	76 07	JBE SHORT StarCraf.004A4638
004A4631	66:81F9 BC00	CMP CX,0BC
004A4636	75 08	JNZ SHORT StarCraf.004A4640

그림 17 미니맵 미네랄 수정 후

sub_4A46A0()함수가 미니맵에 유닛을 그려주는 함수이다. 미네랄때와 마찬가지로 수정해주면 된다. 코드가 유사하기 때문에 자세한 설명은 생략한다.

```

004A475E | 85C9 | TEST ECX,ECX
004A4760 | 74 11 | JE SHORT StarCraf.004A4773
004A4762 | 8405 F80E6D00 | TEST BYTE PTR DS:[6D0EF8],AL
004A4768 | 75 11 | JNZ SHORT StarCraf.004A477B
004A476A | A1 FC0E6D00 | MOV EAX,DWORD PTR DS:[6D0EFC]
004A476F | 85C0 | TEST EAX,EAX
004A4771 | EB 06 | JMP SHORT StarCraf.004A4779
004A4773 | 8505 90F05700 | TEST DWORD PTR DS:[57F090],EAX
004A4779 | 90 | NOP
004A477A | 90 | NOP
004A477B | 0FB746 64 | MOVZX EAX,WORD PTR DS:[ESI+64]
004A477F | 8B4E 0C | MOV ECX,DWORD PTR DS:[ESI+C]
004A4782 | C1E0 02 | SHL EAX,2
004A4785 | 33D2 | XOR EDX,EDX
004A4787 | 8A90 60406600 | MOV DL,BYTE PTR DS:[EAX+664060]
004A478D | 81E2 01FFFFFF | AND EDX,FFFFFF01
004A4793 | 52 | PUSH EDX
004A4794 | 33D2 | XOR EDX,EDX
004A4796 | 8A90 42286600 | MOV DL,BYTE PTR DS:[EAX+662842]
004A479C | 0FB680 40286600 | MOVZX EAX,BYTE PTR DS:[EAX+662840]
004A47A3 | 52 | PUSH EDX
004A47A4 | 0FBF51 16 | MOVSX EDX,WORD PTR DS:[ECX+16]
004A47A8 | 50 | PUSH EAX
004A47A9 | 0FBF41 14 | MOVSX EAX,WORD PTR DS:[ECX+14]
004A47AD | 8A4D FF | MOV CL,BYTE PTR SS:[EBP-1]
004A47B0 | 52 | PUSH EDX
004A47B1 | E8 0AF7FFFF | CALL StarCraf.004A8EC0

```

그림 18 미니맵 유닛 및 건물 수정 후

맵핵을 위한 코드는 모두 수정했다. 스타크래프트 상에서 보면 제대로 동작하고 있음을 확인할 수 있다.



그림 19 스타크래프트 맵핵(리버싱中)

3. 프로그램으로 만들자!

리버싱을 완료했으니 DLL Injection을 이용해서 프로그램으로 만들어보자. 우선 수정해야 할 코드와 그 역할을 표로 정리해보면 다음과 같다.

지형 그리기	
480180	90 90
48018A	EB
유닛 및 건물 그리기	
4981E5	90 90
498203	90 90
적 유닛 클릭 가능	
46F8CA	90 90 90 90 90 90
미니맵 지형	
4A3A88	90 90
4A3A93	EB
미니맵 미네랄	
4A4621	90 90
미니맵 유닛 및 건물	
4A4779	90 90

표 1 수정 목록 테이블

DLL Injection에는 여러 가지 방법이 있지만 여기서 사용할 방법은 CreateRemoteThread를 이용해 LoadLibrary함수로 DLL을 동적으로 맵핑시키는 것을 사용할 것이다. 또한 DLL 내부에서 서브클레싱을 사용하여 특정 키가 눌러졌을 때 맵핵을 작동시키게 만들 것이다. DLL Injection의 자세한 방법은 생략하기로 하고 소스코드만 첨부한다.

삽입될 DLL에서 코드들을 직접 수정하도록 하고, Home키가 눌러질때 맵핵이 활성화 되었다가 한번더 누르게 되면 비활성화 되도록 제작 하였다.

```
// mapDLL.cpp : DLL 응용 프로그램에 대한 진입점을 정의합니다.
//

#include "stdafx.h"
// windows.h
```

```

#ifdef _MANAGED
#pragma managed(push, off)
#endif

#define PATCHCOUNT 9

struct replace
{
    long addr;
    int size;
    char data[10];
    char backup[10];
}    patch_data[PATCHCOUNT];

WNDPROC    oldWndProc;
bool map_enable;

void dataset()
{
    int index=0;

    //지형
    patch_data[index].addr=0x480180;
    patch_data[index].size=2;
    strcpy_s(patch_data[index].data,"Wx90Wx90");
    strncpy_s(patch_data[index].backup,    (char    *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    patch_data[index].addr=0x48018A;
    patch_data[index].size=1;
    strcpy_s(patch_data[index].data,"WxEB");
    strncpy_s(patch_data[index].backup,    (char    *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    //유닛 및 건물

```

```

    patch_data[index].addr=0x4981E5;
    patch_data[index].size=2;
    strcpy_s(patch_data[index].data,"Wx90Wx90");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    patch_data[index].addr=0x498203;
    patch_data[index].size=2;
    strcpy_s(patch_data[index].data,"Wx90Wx90");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    //적 유닛 클릭
    patch_data[index].addr=0x46F8CA;
    patch_data[index].size=6;
    strcpy_s(patch_data[index].data,"Wx90Wx90Wx90Wx90Wx90Wx90");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    //미니맵 지형
    patch_data[index].addr=0x4A3A88;
    patch_data[index].size=2;
    strcpy_s(patch_data[index].data,"Wx90Wx90");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    patch_data[index].addr=0x4A3A93;
    patch_data[index].size=1;
    strcpy_s(patch_data[index].data,"WxEB");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    //미니맵 미네랄

```

```

    patch_data[index].addr=0x4A4621;
    patch_data[index].size=2;
    strcpy_s(patch_data[index].data,"Wx90Wx90");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;

    //미니맵 유닛 및 지형
    patch_data[index].addr=0x4A4779;
    patch_data[index].size=2;
    strcpy_s(patch_data[index].data,"Wx90Wx90");
    strncpy_s(patch_data[index].backup, (char *)patch_data[index].addr,
patch_data[index].size);
    index+ + ;
}

void patch()
{
    DWORD temp;

    for(int i=0; i<PATCHCOUNT; i+ + )
    {
        VirtualProtect((LPVOID)patch_data[i].addr, patch_data[i].size,
PAGE_EXECUTE_READWRITE, &temp);
        for(int j=0; j<patch_data[i].size; j+ + )
            *(char *)(patch_data[i].addr+j) = patch_data[i].data[j];

        VirtualProtect((LPVOID)patch_data[i].addr, patch_data[i].size,
PAGE_EXECUTE_READ, &temp);
    }
    map_enable=true;
}

void patch_restore()
{
    DWORD temp;

    for(int i=0; i<PATCHCOUNT; i+ + )

```



```

    {
        VirtualProtect((LPVOID)patch_data[i].addr,          patch_data[i].size,
PAGE_EXECUTE_READWRITE, &temp);
        for(int j=0; j<patch_data[i].size; j++)
            *(char *)(patch_data[i].addr+j) = patch_data[i].backup[j];

        VirtualProtect((LPVOID)patch_data[i].addr,          patch_data[i].size,
PAGE_EXECUTE_READ, &temp);
    }
    map_enable=false;
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT nMsg, WPARAM wParam,
LPARAM lParam)
{
    if (nMsg==WM_KEYDOWN)
    {
        if (wParam==VK_HOME && map_enable==false)
        {
            patch();
        }
        else if (wParam==VK_HOME && map_enable==true)
        {
            patch_restore();
        }
    }

    return CallWindowProc(oldWndProc, hWnd, nMsg, wParam, lParam);
}

BOOL APIENTRY DllMain( HMODULE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved
                      )

```

```

{
    HWND hwnd;

    if (ul_reason_for_call==DLL_PROCESS_ATTACH)
    {
        map_enable=false;
        dataset();

        hwnd=FindWindow(0, "Brood War");
        if(hwnd==0)
        {
            MessageBox(0, "can't find Brood War!", "Error!", 0);
        }
        else
        {
            oldWndProc=(WNDPROC)SetWindowLong(hwnd,
GWL_WNDPROC, (LONG)WndProc);
        }

    }

    return TRUE;
}

#ifdef _MANAGED
#pragma managed(pop)
#endif

```

표 2 DLL 소스 코드

아래는 DLL을 직접 주입시키기 위한 코드이며, 프로그램이 실행된 후 4초가 경과했을 때 활성화 되어있는 프로세스에 대해 DLL 주입을 시도한다. 따라서 본 프로그램을 실행시킨 후 4초가 경과하기 이전에 스타크래프트를 활성화 시키면 DLL이 주입되고 스타크래프트 내에서 home키를 이용해 맵핵을 활성화 시킬 수 있다. 물론 이 프로그램은 꼭 스타크래프트만이 아닌 모든 프로세스에서 사용이 가능하며, 주입시키기 위한 DLL의 경로만 재대로 지정해주었다면 활성화된 프로세스에 DLL Injection시키는 유용한 프로그램이다. 물론 린츄씨는 간단히 테스트를 위해 아래 프로그램을 사용했으며 실제로 맵핵이라는 틀을 갖추기 위해서는 직접 스타크래프트에 대한 Injection틀을 제작하길 권장한다. 언제나 귀차니즘은 무섭다아~

```

#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    HWND h;
    HANDLE handle, threadhandle;
    HMODULE khandle;

    PTHREAD_START_ROUTINE getproc;

    void *mem;
    char *dllpath;

    //DLL 경로
    dllpath="C:WWtest.dll";

    DWORD pid, len;

    //4초후 활성화된 윈도우에 DLL 주입
    Sleep(4000);

    h=GetForegroundWindow();
    if (h == 0)
    {
        printf("Error!");
        exit(0);
    }

    GetWindowThreadProcessId(h, &pid);
    handle=OpenProcess(PROCESS_ALL_ACCESS, 0, pid);

    //LoadLibrary함수의 주소
    khandle=GetModuleHandle("kernel32.dll");
    getproc=(PTHREAD_START_ROUTINE)GetProcAddress(khandle,

```

```

"LoadLibraryA");

    //타겟 프로세스 공간에 DLL 경로 복사
    mem=VirtualAllocEx(handle, 0, strlen(dllpath)+ 1,
MEM_RESERVE|MEM_COMMIT, PAGE_READWRITE);
    WriteProcessMemory(handle, mem, dllpath, strlen(dllpath)+ 1, &len);

    //Thread 생성
    threadhandle=CreateRemoteThread(handle, 0, 0, getproc, mem, 0, 0);

    CloseHandle(threadhandle);
    CloseHandle(handle);

    printf("complete!");
}

```

표 3 DLL Injector 소스코드

4. 실험 및 결과

우선 스타크래프트를 실행시킨 후 DLL Injector를 실행시키고 바로 스타크래프트를 활성화시킨다. 그 후 컴퓨터 몇 대를 집어넣고 게임을 플레이 해보자. HOME키를 눌러보면 잘 작동한다는 것을 확인할 수 있다.



그림 20 강건너 불구경

크립이 보이지 않는 저그 기지에서 히드라와 질럿이 싸움을 벌이고 있다. 물론 린츨씨는 강건너 불구경이다.



그림 21 적기지 염탐

추가. 버전별 수정 코드

패치로 인한 먼산 모드. 일단 1.161버전에 대한 추가!

지형 그리기	
47FCE0	90 90
47FCEA	EB
유닛 및 건물 그리기	
4982F5	90 90
498313	90 90
적 유닛 클릭 가능	
46F42A	90 90 90 90 90 90
미니맵 지형	
4A3B98	90 90
4A3BA3	EB
미니맵 미네랄	
4A4731	90 90
미니맵 유닛 및 건물	
4A4889	90 90

표 4 1.161수정 목록 테이블

참고문헌

- [1] Robert Kuster, Three Ways to Inject Your Code into Another Process
- [2] Darawk, DLL Injection Tutorial