



Pwning multiplayer games

- case Starcraft-Broodwar

Code⚡Engn

www.CodeEngn.com

2013 CodeEngn Conference 08

Kwon Hyuk (pwn3r) | B10S | 2013/07/13

Outline

1. Introduce

1. Who am I?
2. Summary
3. Why multiplayer game?
4. Choosing target

2. Starcraft

1. What is starcraft?
2. Attack vectors

3. Fuzzing starcraft

1. Structure of map file
2. Attack scenario
3. Fuzzing scenario
4. Exploitable crashes

4. Exploitation

1. Heap spray
2. Process continuation

5. Conclusion

1. Timeline
2. I have talked about ..
3. Q & A

Chapter 1: Introduce

- 1) Who am I?
- 2) I will talk about ..
- 3) Why multiplayer game?
- 4) Choosing target

1. Introduce

\$ whoami

Name : Kwon Hyuk
Age : 19
FB : <http://fb.com/kwonpwn3r>
Blog : <http://pwn3r.tistory.com>
Job : 고삼완전체



1. Introduce

I will talk about ..

- BoB 에서 진행했던 '멀티플레이어 게임 취약점 점검' 프로젝트
- Multiplayer game 중 하나인 Starcraft-broodwar에서 발견한 RCE(remote code execution) 취약점과 공략하기까지의 과정
- 간단한 Exploitation 기술들



1. Introduce

Why multiplayer game?

1. 수 많은 게임 이용자

- 통계자료를 대지 않아도 다들 알고 있는 엄청난 이용자 수

2. 게임 보안의 불균형한 발전

- Multiplayer 게임의 시장규모와 이용자 수가 증가함에 따라 게임보안도 발전함.
- But, 게임의 룰에서 탈출하는 Abusing bug를 막는 데에 주력하여 취약성 측면에서의 보안을 놓치고 있음.

3. 다양한 공격벡터

- 게임에 따라 Map, Chatting, Image, Sound 등 다양한 공격 벡터가 존재함.

4. Just for fun

- 취약점 찾다가 잘 안되면 게임 한판 ㄱㄱ

1. Introduce

Choosing target



- 현재 가장 유명한 게임



- 그림으로 대화하는 기능



- 음성 채팅 기능



- 그림 영상 정보를 주고받는 기능

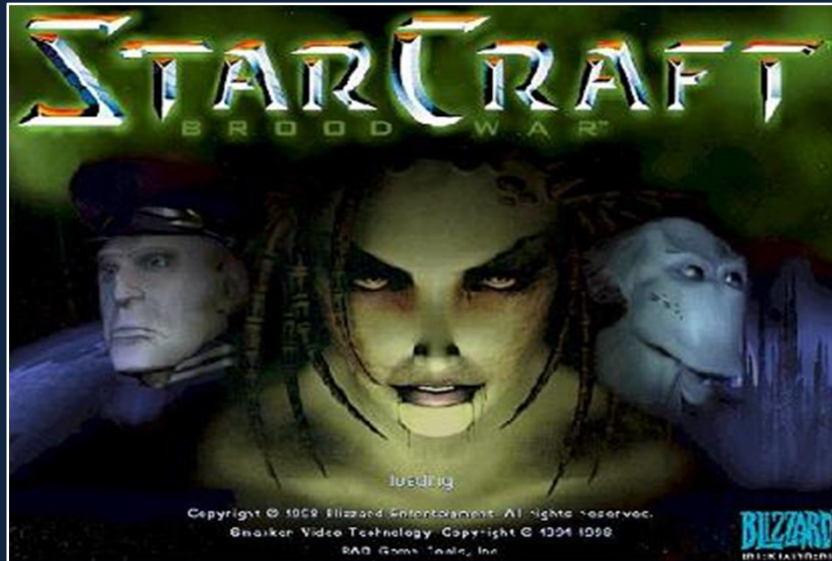
Chapter 2: Starcraft

1) What is starcraft?

2) Attack vectors

2. Starcraft

What is starcraft?



2. Starcraft

게임 진행 방식

- 기본적으로 하나의 맵으로 방을 개설하고, 타 플레이어들이 방에 입장하여 같이 게임을 진행하는 방식.
- 방에 입장한 플레이어들에게 방장이 맵을 보내줌.
- 즉, 플레이어들은 같은 맵 파일을 공유하여 게임을 진행하게 됨.



2. Starcraft

게임 진행 방식

- 기본적으로 하나의 맵으로 방을 개설하고, 타 플레이어들이 방에 입장하여 같이 게임을 진행하는 방식.
- 방에 입장한 플레이어들에게 방장이 맵을 보내줌.
- 즉, 플레이어들은 같은 맵 파일을 공유하여 게임을 진행하게 됨.



Chapter 3: Fuzzing Starcraft

- 1) Structure of map file
- 2) Fuzzing scenario
- 3) Exploitable crashes

3. Fuzzing Starcraft

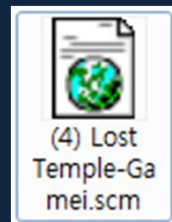
Map file structure

1) 확장자

- .scm / .scx

2) File format

- MPQ compressed file format
(MPQ = Mike O'Brien Pack)
- 맵 파일은 하나의 MPQ 압축파일.
- 맵을 구성하는 파일들이 하나의 MPQ 압축 파일에 압축되어 있는 구조



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	4D	50	51	1A	20	00	00	00	3D	7C	04	00	00	00	03	00	MPQ. ...=
00000010	ED	3B	04	00	ED	7B	04	00	00	04	00	10	05	00	00	00	i;...i{.....
00000020	EE	E4	B5	86	7A	D9	7B	AE	F5	4B	A1	7D	1B	AC	C6	BA	iäptzÛ{@öK;}.~E°
00000030	AD	DA	3C	14	54	AD	FB	8A	B5	E2	FB	B6	16	18	84	6B	.Ú<.T.ûŠpâûŕ...„k

3. Fuzzing Starcraft

Map file structure

3) MPQ decompressed

: MPQ 압축파일에 압축된 파일들

- (listfile)

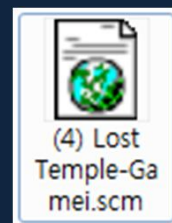
=> 압축된 파일명 list

- staredit\scenario.chk

=> 맵의 설정 정보를 저장하는 핵심 파일

- etc ..

=> 그 외 게임설정에 따라 음악파일, 그래픽 파일 등이 추가될 수 있다.



MPQ
Decompress

File Name	Type	Locale	Size
staredit	파일 폴더		<DIR>
(listfile)	파일	Neutral	23
File Name	Type	Locale	Size
scenario.chk	CHK 파일	Neutral	193 855

3. Fuzzing Starcraft

Map file structure

4) Scenario.chk

- 1) 맵의 실질적인 설정 정보가 저장된 파일
- 2) 섹션 단위로 구성됨.
- 3) TYPE, VER, OWNER, TRIG, ... 등 39개 종류의 Section이 존재.
- 4) [Section name] [Section size] [Section data] 가 반복된 단순한 구조
(4 byte) (4 byte) (n bytes)

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	54	59	50	45	04	00	00	00	52	41	57	53	56	45	52	20	TYPE ... RAWS VER
00000010	02	00	00	00	3F	00	49	56	45	32	02	00	00	00	0B	00 ? . IVE2
00000020	56	43	4F	44	10	04	00	00	34	19	CA	77	99	DC	68	71	VCOD....4.ÊwÜh
00000030	0A	60	BF	C3	A7	E7	75	A7	1F	29	7D	A6	D7	B0	3A	BB	.`¿Ã\$çu\$.)} ×°:»

Section name

Section size

Section data

3. Fuzzing Starcraft

Attack scenario

취약점 발생시점에 따른 공격 시나리오



1) Victim이 악성 맵으로 방을 Open하다가 취약점 유발

- Victim에게 악성 맵을 전송해주고, Victim이 해당 맵으로 방을 개설하는 과정에서 취약점이 유발됨.
- 악성 맵을 인터넷에 유포해도 해당 맵을 받아서 방을 개설할 사람들은 많지 않기 때문에, 공격대상이 적어 파급력이 떨어짐.



2) 공격자가 악성 맵으로 방을 Open하고, Victim이 접속시 취약점 유발

- 공격자가 만든 방에 접속하는 Victim은 모두 취약점이 유발됨.
- 감염된 Victim은 퇴장시켜버리면 또 다른 Victim들의 접속을 기다릴 수 있기 때문에 파급력이 높음.



3) 공격자가 악성 맵으로 방을 Open하고, 게임 시작 시 취약점 유발

- 공격자가 만든 방에 접속하고, 게임을 시작하면 Victim은 같은 방에 있는 Victim은 모두 취약점이 유발됨.
- 게임을 시작해야 감염이 된다는 점 때문에, 한번에 방에 입장할 수 있는 최대인원(8명)까지만 취약점을 유발시킬 수 있음.
(2번에 비해 파급력이 떨어지긴 하지만, 자동화 시킨다면 낮지 않은 파급력)

3. Fuzzing Starcraft

Attack scenario



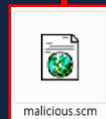
1) Victim이 악성 맵으로 방을 Open하다가 취약점 유발

- Victim에게 악성 맵을 전송해주고, Victim이 해당 맵으로 방을 개설하는 과정에서 취약점이 유발됨.
- 악성 맵을 인터넷에 유포해도 해당 맵을 받아서 방을 개설할 사람들은 많지 않기 때문에, 공격대상이 적어 파급력이 떨어짐.

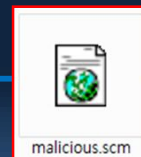


Attacker

Send
Map file



Victim



Choosing map



3. Fuzzing Starcraft

Attack scenario

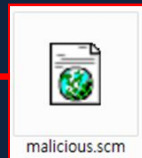


2) 공격자가 악성 맵으로 방을 Open하고, Victim이 접속시 취약점 유발

- 공격자가 만든 방에 접속하는 Victim은 모두 취약점이 유발됨.
- 감염된 Victim은 퇴장시켜버리면 또 다른 Victim들의 접속을 기다릴 수 있기 때문에 파급력이 높음.



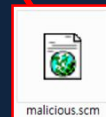
Attacker



Choosing map
& Create game

방제목:
2:2 로템 ㅋㅋ

Send
Map file



Join the game



Victim



3. Fuzzing Starcraft

Attack scenario

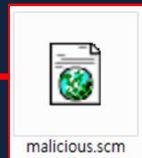


3) 공격자가 악성 맵으로 방을 Open하고, 게임 시작 시 취약점 유발

- 공격자가 만든 방에 접속하고, 게임을 시작하면 Victim은 같은 방에 있는 Victim은 모두 취약점이 유발됨.
- 게임을 시작해야 감염이 된다는 점 때문에, 한번에 방에 입장할 수 있는 최대인원(8명)까지만 취약점을 유발시킬 수 있음.



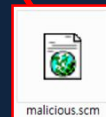
Attacker



Choosing map
& Create game

방제목:
2:2 로템 ㅋㅋ

Send
Map file



Join the game



Victim

Start game!



3. Fuzzing Starcraft

Fuzzing scenario

Scenario 1

- 1개의 Client 필요

Client 1 : 스크립트가 맵 파일을 조작한 후 Strarcraft를 실행하여 방을 개설함.

Scenario 2

- 2개의 Client 필요

Client 1 : Starcraft를 실행하여 UDP 모드로 방을 개설하고, 스크립트가 Starcraft 프로세스의 메모리에 있는 맵 파일을 조작함을 반복

Client 2 : 스크립트가 Starcraft를 실행하여 UDP 모드에서 Client (1)이 개설한 방에 참가함을 반복 -> Exception 발생여부 확인

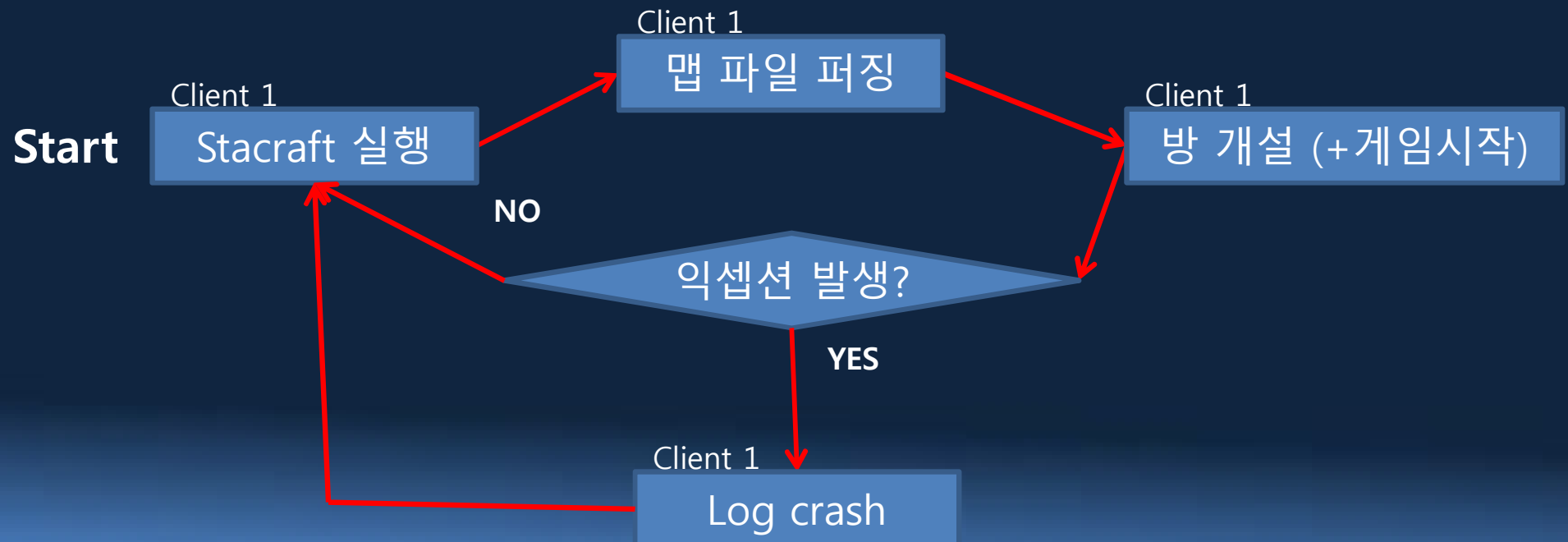
3. Fuzzing Starcraft

Fuzzing scenario

Scenario 1.

- 1개의 Client 필요

Client 1 : 스크립트가 맵 파일을 조작한 후 Strarcraft를 실행하여 방을 개설함.



3. Fuzzing Starcraft

Fuzzing scenario

Scenario 1.

장점

- 스크립트 작성이 간단함.
- 1대의 클라이언트만 필요

단점

- 맵 파일을 압축 해제하지 않고 조작할 경우, 방을 개설할 때에 크래시가 발생하기 때문에 압축 해제를 할 때에만 적용 가능한 시나리오

3. Fuzzing Starcraft

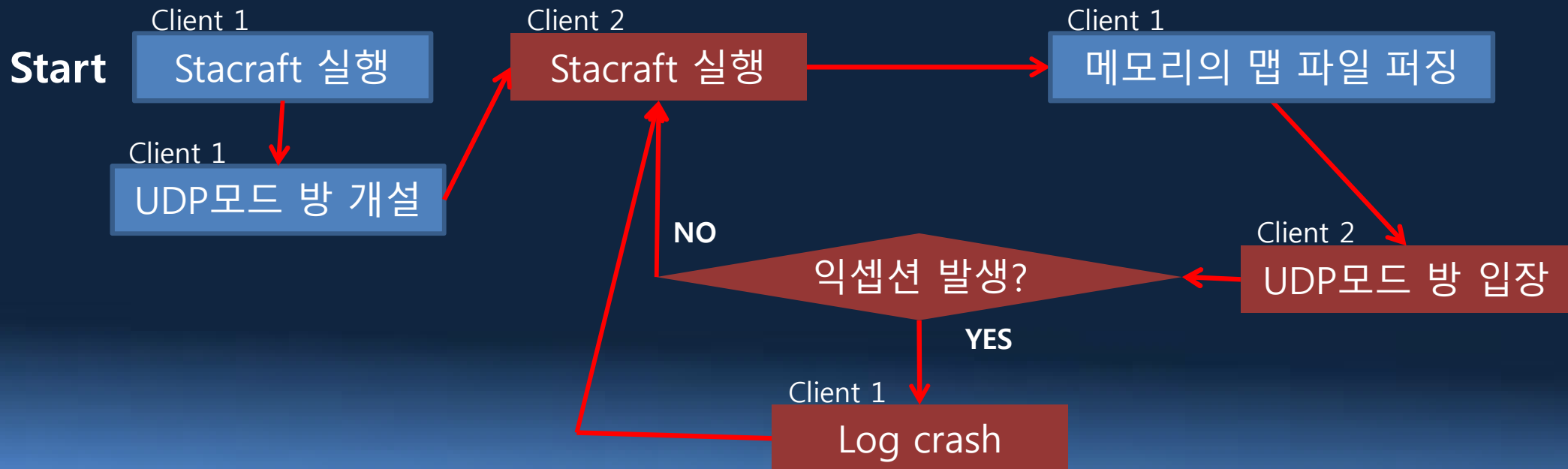
Fuzzing scenario

Scenario 2.

- 2개의 Client 필요

Client 1 : Starcraft를 실행하여 UDP 모드로 방을 개설하고, 스크립트가 Starcraft 프로세스의 메모리에 있는 맵 파일을 조작함을 반복

Client 2 : 스크립트가 Starcraft를 실행하여 UDP 모드에서 Client (1)이 개설한 방에 참가함을 반복 -> Exception 발생여부 확인



3. Fuzzing Starcraft

Fuzzing scenario

Scenario 2.

장점

- 타 플레이어가 방에 접속하는 상황에서 퍼징할 수 있음.
- 방을 만드는 Client는 메모리에 있는 맵 파일을 조작하기 때문에, 방을 한번만 열어두면 됨.

단점

- 구현이 어려움.
- 구현이 힘들.
- 구현이 짜증남.

3. Fuzzing Starcraft

Fuzzing scenario

취약점 발생 시점 압축해제 유무	방 접속 시 취약점 유발	방 접속 후 게임 시작 시 취약점 유발
압축 해제 전	①	②
압축 해제 후	③	④

3. Fuzzing Starcraft

①, ③ 압축 해제 전, 후 + 게임 시작 전



3. Fuzzing Starcraft

①, ③ 압축 해제 전, 후 + 게임 시작 전

Scenario 2 적용

동작

- 1) Client 1 : UDP 모드로 방 개설
- 2) Client 1 : 메모리에 올라와 있는 맵 파일 조작 (압축 해제 전/후)
- 3) Client 2 : UDP 모드에서 Client 1이 개설한 방 입장
- 4) Client 1 : Client 2에게 맵 파일 전송
- 5) Client 2 : 익셉션 발생여부 확인

3. Fuzzing Starcraft

①, ③ 압축 해제 전, 후 + 게임 시작 전

Scenario 2 적용

구현

Client 1 : Immunity debugger의 pycommand 이용 (Memory read/write)

```
def mutate(imm, loop, ptr, size):  
    mutate_db = {}  
  
    for i in range((size//100)*5):  
        rand_idx = random.randrange(0x20, size)  
        rand_val = random.randrange(0x0, 0x100)  
        mutate_db[rand_idx] = chr(rand_val)  
        imm.writeMemory(ptr+rand_idx, chr(rand_val))  
    return mutate_db
```

Client 2 : 키보드 단축키를 이용해 매크로 형식으로 구현

```
sendmsg(c_void_p(self.hWnd), 0x0102, ord("m"), 0)  
time.sleep(1)  
sendmsg(c_void_p(self.hWnd), 0x0102, ord("e"), 0)  
time.sleep(3)  
for i in range(0,5):  
    sendmsg(c_void_p(self.hWnd), 0x0100, 0x28, 0)  
    time.sleep(0.5)  
sendmsg(c_void_p(self.hWnd), 0x0102, ord("o"), 0)  
time.sleep(1.5)  
sendmsg(c_void_p(self.hWnd), 0x0102, ord("o"), 0)  
time.sleep(1.5)  
sendmsg(c_void_p(self.hWnd), 0x0102, ord("g"), 0)
```

3. Fuzzing Starcraft

Sub problems

Problem 1) MPQ 압축 포맷을 자동으로 decompress 시켜야 함.

=> 외국에서 MPQ 포맷 파싱을 위해 만들어둔 stormlib라는 오픈소스 라이브러리 이용.

```
9      TCHAR buf[256] = {0,};
10     HANDLE hMpq=0, hFile=0;
11     LPSTR lpFileData;
12     DWORD dwFileSize, dwByteRead, dwTmp;
13     const TCHAR* name = _T("C:\\mylogs\\rwqqwrqwr.scm");
14
15     SFileOpenArchive(name, 0, 0, &hMpq);
16     MultiByteToWideChar(CP_ACP, MB_PRECOMPOSED, argv[1], 256, buf, 256);
17     SFileAddFileEx(hMpq, (const TCHAR*)buf, argv[1], MPQ_FILE_IMPLODE, NULL, NULL);
18
19     SFileCompactArchive(hMpq, NULL, NULL);
20     SFileCloseArchive(hMpq);
21     return 0;
```


3. Fuzzing Starcraft

Sub problems

Problem 2) UDP 모드로 게임 목록 확인 후 10초가 지나면 더 이상 같은 게임이 검색되지 않음.

=> 직접 udp로 broadcast 패킷을 전송하여 해결

```
1  from socket import *
2  import time
3
4  f = open("data", "rb")
5  data = f.read()
6  f.close()
7
8  s = socket(AF_INET, SOCK_DGRAM)
9  s.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
10 s.bind(("0.0.0.0", 6333))
11 while 1:
12     s.sendto(data, ("192.168.123.104", 6111))
13     time.sleep(2)
14 s.close()
```

3. Fuzzing Starcraft

Sub problems

Problem 3) 방장이 클라이언트에게 맵을 전송할 때, 체크섬과 같이 전송함.
=> Starcraft를 분석하여 checksum생성루틴 찾아내 구현.

```
xor_table = [0, 1996959894, 3993919788L,  
def get_checksum(data):  
    global xor_table  
    a = 0xffffffff  
    for i in range(0, len(data)):  
        tmp = ord(data[i])  
        tmp = tmp ^ a  
        a = a >> 8  
        tmp = tmp & 0xff  
        tmp = xor_table[tmp]  
        tmp = tmp ^ a  
        tmp = tmp % 0x100000000  
        a = tmp  
    return a
```

Quiz !

돈금없이 책 뿌립니다

- 1) 발표자가 MPQ 압축을 풀기위해 사용했던 라이브러리 이름은?

Quiz !

돈금없이 책 뿌립니다

- 2) 제가 소개했던 공격벡터를 제외한 다른 멀티플레이어 게임에서의 공격벡터를 50자 이내로 서술하시오 (5점)

3. Fuzzing Starcraft

② 압축 해제 전 + 게임 시작 후

압축을 해제 하기 전에 맵을 조작 시

- 1) 방을 개설하다가 익셉션이 발생
 - 2) 방을 개설/입장하자마자 익셉션이 발생
- ⇒ 게임 시작자체가 불가능

따라서 Scenario 1 제외.



3. Fuzzing Starcraft

④ 압축 해제 후 + 게임 시작 후



3. Fuzzing Starcraft

④ 압축 해제 후 + 게임 시작 후

Scenario 2 적용

동작

- 1) 파일 조작
- 2) Starcraft 실행 -> 방 개설
- 3) 게임 시작
- 4) 익셉션 발생여부 확인

구현

- 키보드 단축키를 이용해 매크로 형식으로 구현 (1, 3번 케이스와 같은 방식)

```
sendmsg(c_void_p(self.hWnd), 0x0102, ord("m"),0)
time.sleep(1)
sendmsg(c_void_p(self.hWnd), 0x0102, ord("e"),0)
time.sleep(3)
for i in range(0,5):
    sendmsg(c_void_p(self.hWnd), 0x0100, 0x28, 0)
time.sleep(0.5)
sendmsg(c_void_p(self.hWnd), 0x0102, ord("o"),0)
time.sleep(1.5)
sendmsg(c_void_p(self.hWnd), 0x0102, ord("o"),0)
time.sleep(1.5)
sendmsg(c_void_p(self.hWnd), 0x0102, ord("g"),0)
```


3. Fuzzing Starcraft

Result

- Fuzzing scenario (4)에서 27종류의 unique crash획득
(다른 시나리오에서도 crash는 발생했지만 종류가 다양하지 않으며 exploitable하지 않음)
- 이 중 2종류는 Exploitable하다고 판단 및 Exploit 작성
(나머지는 exploitable하지 않거나 분석 중)

받은편지함	Fuzzer notify ..	ected ! The file saved as C:\mylogs\ci
받은편지함	Fuzzer notify ..	n detected ! The file saved as C:\mylc
받은편지함	Fuzzer notify ..	xion detected ! The file saved as C:\rr
받은편지함	Fuzzer notify ..	- Exception detected ! The file saved a
받은편지함	Fuzzer notify ..	tion detected ! The file saved as C:\my
받은편지함	Fuzzer notify ..	ption detected ! The file saved as C:\v
받은편지함	Fuzzer notify ..	he file saved as C:\mylogs\cr
받은편지함	Fuzzer notify ..	he file saved as C:\mylogs\cra
받은편지함	Fuzzer notify ..	- Exception detected ! The file saved a
받은편지함	Fuzzer notify ..	:ted ! The file saved as C:\mylogs\cra
받은편지함	Fuzzer notify ..	Exception detected ! The file saved a:
받은편지함	Fuzzer notify ..	on detected ! The file saved as C:\my



3. Fuzzing Starcraft

Exploitable crashes

Case (1) **ADD [edi*4 + DATA_TABLE], ebx**

Range of EDI : 0x00000000 ~ 0x0000ffff (Controllable)

Range of EBX : 1 or -1 (Fixed)

- 임의의 주소에 ebx 값(1 or -1)을 더할 수 있는 취약점.
- > DATA_TABLE 메모리 뒤에 있는 함수포인터 값을 반복적으로 ADD하여 원하는 코드의 주소로 바꿔준 후, 함수포인터가 호출되도록 하여 공략.

Case (2) **CALL [edx* 4 + FUNC_TABLE]**

Range of EDX : 0x00 ~ 0xff (Controllable)

- FUNC_TABLE은 0x3c * 4의 크기를 가지며, 그 뒤로는 쓰레기 값이 들어가 있음.
- EDX를 0x3d~ 0xff 사이의 값으로 넣어주면, 쓰레기 값을 코드 주소로서 호출할 수 있음.
- > 쓰레기 값이 나타내는 주소에 쓸만한 가젯이 있거나, 맵 파일의 데이터가 있다면 best case.
- > 아니라면 Heap spray로 해당 메모리를 할당시켜 공략.

Chapter 4: Exploitation

- 1) Heap spray
- 2) Process continuation

4. Exploitation

How to exploit?

Crash case (2)

`CALL [edx* 4 + FUNC_TABLE]`

- 기존 FUNC_TABLE의 크기는 $0x3c * 4$ 이지만, `edx`의 크기를 제한하지 않음.
- `edx` 값을 $0x3c < idx \leq 0xff$ 의 범위로 만들어주면 FUNC_TABLE에 있는 정상 함수포인터가 아닌, 쓰레기 값을 호출하다가 crash발생.


EDX: 0x0000003d

FUNC_TABLE			
Handler1	Handler2	Handler3	Handler4
.....
Handler39	Handler3a	Handler3b	Handler3c
Trash value	Trash value	Trash value	Trash value

Crash



4. Exploitation

It can not be exploited.. 12/12/24

**권혁**✕

eip 00000000으로 바뀌는 크래시 찾았는데 크리스마스 선물이길 빌며 ㅜㅜ

2012.12.24 ☆ 댓글쓰기 좋아요 싫어요

**권혁**✕

크흑 이건 익스플로잇 불가능한거였네요.

스타맵파일에서 트리거라는 기능이 저장되어있는 부분에서 인덱스를 가져와서, 함수포인터 테이블을 참조하여 call 하다가 죽는거였는데 함수포인터 테이블에 NULL있어서 거기서 점프한거였습니다.


이 함수포인터 테이블이 조작 가능하면 모르겠는데, 이게 스타.exe를 실행할때 처음부터 고정되어있는거네요;; gg..

결국 개발자들이 함수포인터 테이블을 잘못만들어 놓았던거였습니다;


2012.12.25 좋아요 싫어요


4. Exploitation

It can be exploited!! 13/02/02



권혁
eip 00000000으로 바뀌는 크래시 찾았는데 크리스마스 선물이길 빌며 ㅜㅜ
2012.12.24 ☆ 댓글쓰기 좋아요 싫어요






권혁
크흑 이걸 익스플로잇 불가능한거였네요.

스타맵파일에서 트리거라는 기능이 저장되어있는 부분에서 인덱스를 가져와서, 함수포인터 테이블을 참조하여 call 하다가 죽는거였는데 함수포인터 테이블에 NULL있어서 거기서 점프한거였습니다.

이 함수포인터 테이블이 조작 가능하면 모르겠는데, 이게 스타.exe를 실행할때 처음부터 고정되어있는거네요;; gg..

결국 개발자들이 함수포인터 테이블을 잘못만들어 놓았던거였습니다;

2012.12.25 좋아요 싫어요



권혁
스타 계산기 몇네요 ㅋㅋㅋㅋ 아 이제서야 뜨다니 -_- 암울하네요 ㅜㅜ ㅋㅋㅋ
원래 exploit불가능하다고 생각해서 버려뒀던 취약점 있었는데 준보형수업듣고 힙스프레이 하는 방법 더 찾아봤는데 결국 힙스프레이로 공략 성공했습니다 ㅋㅋ 감사
합니다 있다가 영상을릴게여 ㅋㅋ

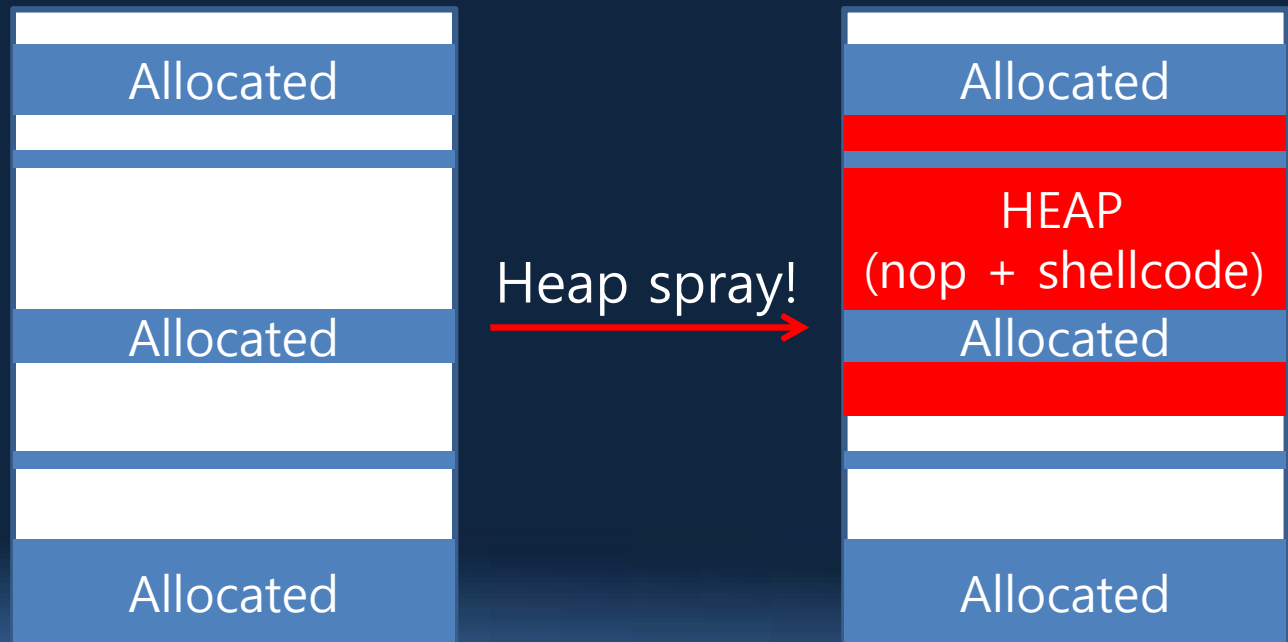
2013.02.02 ☆ 댓글쓰기 좋아요 싫어요

4. Exploitation

Heap spray

1) What is heap spray?

- One of exploitation technique
- Heap에 nop와 셸코드를 spray하듯이 엄청나게 뿌려서 원하는 메모리에 셸코드를 할당시키는 기술.



4. Exploitation

Heap spray

2) How to heap spray at Starcraft Broodwar?

- Browser나 Flash처럼 script를 사용할 수 없음.
- 맵 파일 데이터를 무작정 넣는다고 해서 그대로 메모리에 올라가지 않음.

→ 'STR' section in scenario.chk

- STR (String) section : 맵에서 필요로 하는 문자열을 저장해두는 section
- Starcraft는 scenario.chk파일의 STR section을 파싱할 때 STR section의 size를 제한하거나 검사하지 않음.
- STR section에 있는 값은 heap메모리에 그대로 복사됨.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000313F0 FF FF FF FF FF FF 53 54 52 20 E6 19 00 00 00 04 yyyyyySTRæ....
00031400 03 08 15 08 2A 08 52 08 92 08 B2 08 BA 08 1F 09 ...*.R.'.*.º...
00031410 86 09 8B 09 9A 09 ED 09 02 0A 1D 0A 2F 0A 54 0A t.<.š.ı.../.T.
00031420 85 0A 99 0A C9 0B ED 0B 2E 0C 56 0C 6B 0C 81 0C ...™.É.ı...V.k...
00031430 8D 0C 97 0C EE 0D 1C 0F 5C 0F 8C 10 BD 10 DE 10 ...ı...\.G.%.B.
00031440 0E 12 2D 12 6D 12 AE 12 FD 12 3E 13 62 13 73 13 ...m.®.ý.>.b.s.
00031450 9D 13 CB 14 EB 14 F6 14 37 15 BC 15 19 16 87 16 ...É.è.ö.7.%....#.
00031460 93 16 D5 16 04 18 12 18 51 18 59 18 5D 18 61 18 ".Ö.....Q.Y.j.a
00031470 65 18 69 18 6D 18 71 18 75 18 79 18 7D 18 81 18 e.i.m.q.u.y.)...
00031480 85 18 89 18 8D 18 91 18 95 18 99 18 9D 18 A1 18 ...%....'.*..™...j.
00031490 A5 18 A9 18 AD 18 B1 18 B5 18 B9 18 BD 18 C1 18 ¥.®...±.µ.º.%.Ä.
000314A0 C5 18 C9 18 CD 18 D1 18 D5 18 D9 18 DD 18 E1 18 Ä.É.İ.Ñ.Ö.Ü.Ý.Á.
000314B0 E5 18 E9 18 ED 18 F1 18 F5 18 F9 18 FD 18 01 19 ä.é.ı.ñ.ö.ù.ý...
000314C0 05 19 09 19 0D 19 11 19 15 19 19 19 1D 19 21 19 .....!..
000314D0 25 19 29 19 2D 19 31 19 35 19 39 19 3D 19 41 19 %.).-.ı.5.9.=.A.
000314E0 45 19 49 19 4D 19 51 19 55 19 5E 19 62 19 66 19 E.I.M.Q.U.^b.f.
000314F0 6A 19 6E 19 72 19 76 19 7A 19 7E 19 82 19 86 19 j.n.r.v.z.~.,.t.
00031500 8A 19 8E 19 92 19 96 19 9A 19 9E 19 A2 19 A6 19 Š.Ž.'-.š.ž.c.!.
00031510 AA 19 AE 19 B2 19 B6 19 BA 19 BE 19 C2 19 C6 19 *.®.*.q.º.%.Ä.Æ.
00031520 CA 19 CE 19 D2 19 D6 19 DA 19 DE 19 E2 19 02 08 È.İ.Ö.Ö.Ü.ß.Á...
00031530 00 46 72 65 64 6F 6D 20 28 53 2E 45 2E 45 2E .Freedom (S.E.E.
00031540 44 29 00 07 20 C7 C1 B7 CE BA EA 20 32 B0 B3 20 D)... ÇÁ.İ°è 2°³
00031550 C7 CA BF E4 C7 D4 2E 00 20 04 44 72 6F 70 73 ÇÊİ&ÇÖ...Drops
00031560 68 69 70 20 3A 20 22 43 61 6E 20 49 20 74 61 6B hip : "Can I tak
00031570 65 20 79 6F 75 72 20 6F 72 64 65 72 3F 22 20 00 e your order?" .
```

4. Exploitation

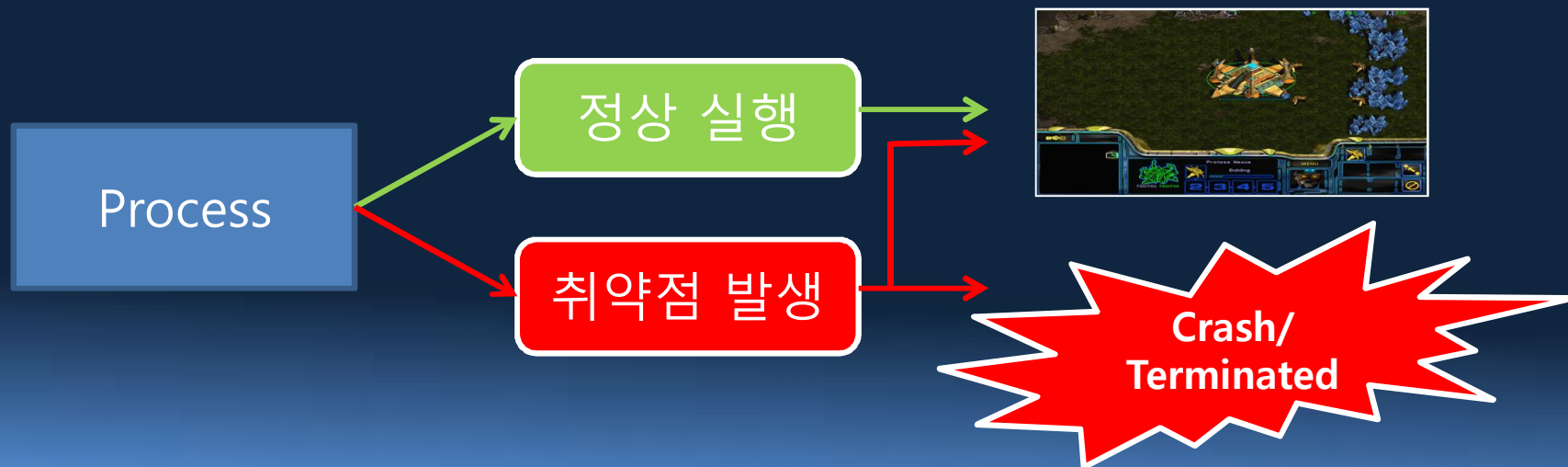
DEMO



4. Exploitation

Advanced Exploitation – Process continuation

- 일반적으로 임의의 코드를 실행하는 Exploit을 작성했을 때, 코드의 실행이 끝나면 해당 process는 종료되거나, 크래시로 이어짐.
- 하지만 취약점이 발생하지 않았을 때의 환경(Stack, Register, ...)을 복구해 주어 Process가 종료되지 않고 정상적으로 Continue 되도록 해주는 것을 Process continuation이라 부름.
- 취약점의 유형이나 상황마다 Process continuation을 구현하는 방법은 다양함.



4. Exploitation

Advanced Exploitation – Process continuation

```
CALL     FUNC_TABLE[idx]
```

- 이 경우에선 취약점의 특성상 process continuation을 구현하기가 굉장히 간단함.
 - 취약점은 함수 포인터 테이블에서 잘못된 인덱스에 있는 함수를 호출함으로써 발생함.
- ⇒ 즉, 원래 정상적인 인덱스를 참조했을 때에 반환되는 레지스터/메모리 상태를 만들어주고 return하면 정상 프로세스 흐름을 이어갈 수 있음.

(하지만 운이 좋게도 레지스터 상태를 함수 호출 전과 똑같이 유지한 채 return해도 정상 프로세스 흐름을 이어감.)

4. Exploitation

Advanced Exploitation – Process continuation

- 1) PUSHAD와 같은 명령으로 레지스터 상태를 보존
- 2) 목표하는 셸코드 실행
 - 단, 셸코드가 esp아래에 있는 값을 변경하거나 스택 외에 메모리에 있는 값을 변경하면 안됨.
- 3) POPAD로 레지스터 상태를 함수 호출 직후로 복구

60	PUSHAD	
31C9	XOR ECX,ECX	
Shellcode		
(ESP 밑에 있는 값을 변경하지 않아야함)		
2E: 61	POPAD	Superfluous prefix
C3	RETN	

4. Exploitation

DEMO



Chapter 5: Conclusion

1) Timeline

2) I have talked about

3) Q & A

5. Conclusion

Timeline

- **2012/12/24**
 - Exploitable crash 발견 & 취약점 원인 분석 완료
(but, Heap spray할 방법 모름)
- **2013/02/02**
 - Heap spray할 방법 찾음 -> Exploit 작성 완료
- **2013/02/15**
 - Process continuation 성공
- **2013/02/27**
 - 취약점 분석 보고서 작성 & Blizzard 취약점 신고 담당 메일로 패치 권고.
 - ~ 2013/05/04 응답 없음
- **2013/05/05**
 - 취약점 패치 재권고
 - ~2013/07/13 응답 없음
- ~ **현재**
 - 응답 없음

5. Conclusion

I have talked about ..

- Multiplayer 게임들의 공격 벡터
- 맵 파일을 공격 벡터로 하여 취약점 점검하기
- Strarcraft 공략 story
- Heap spray
- 매우 간단한 process continuation
- 심준보 멘토님 짱짱

5. Conclusion

Q & A



Thank you!

Code⚡Engn

www.CodeEngn.com

2013 CodeEngn Conference 08