POC – Hackers dream

Network #1,2

Hacking Group “OVERTIME”

crash <crashn5p@gmail.com>2008.11.10

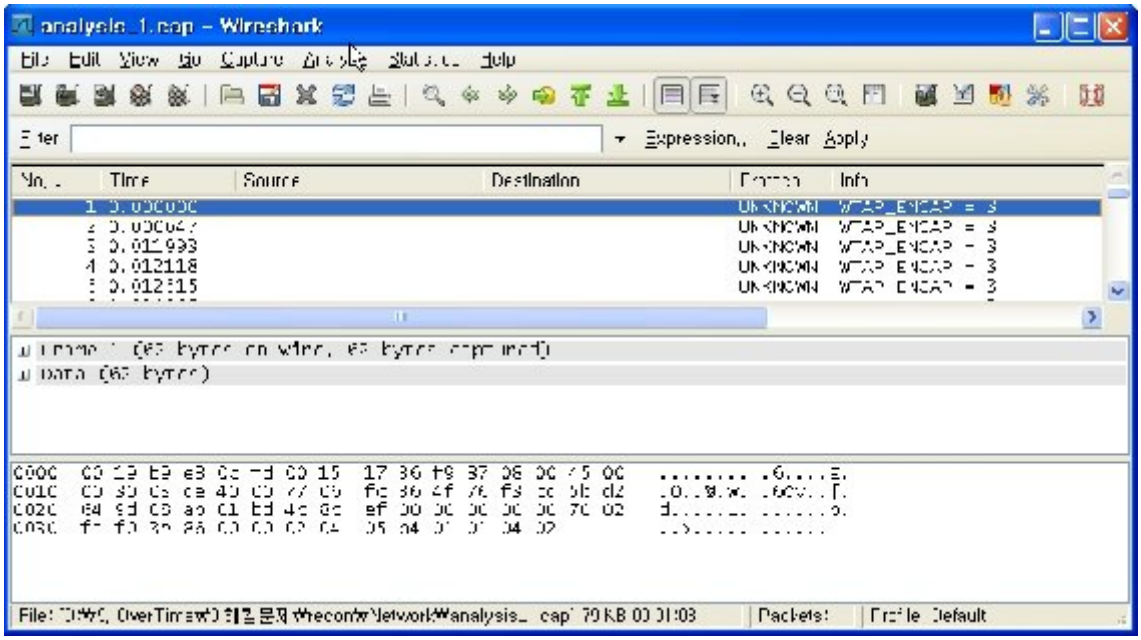
1. 문제 이해

1번 문제
- 잘 분석하면 답을 알 수 있다. (?)

2번 문제
- analysis_2의 압축 비밀번호는 1번문제의 정답 + 100번째 패킷의 TCP 헤더 체크섬 값(십진수)
- 1번 정답이 2번 문제를 풀 수 있는 패킷 위치를 알려줌

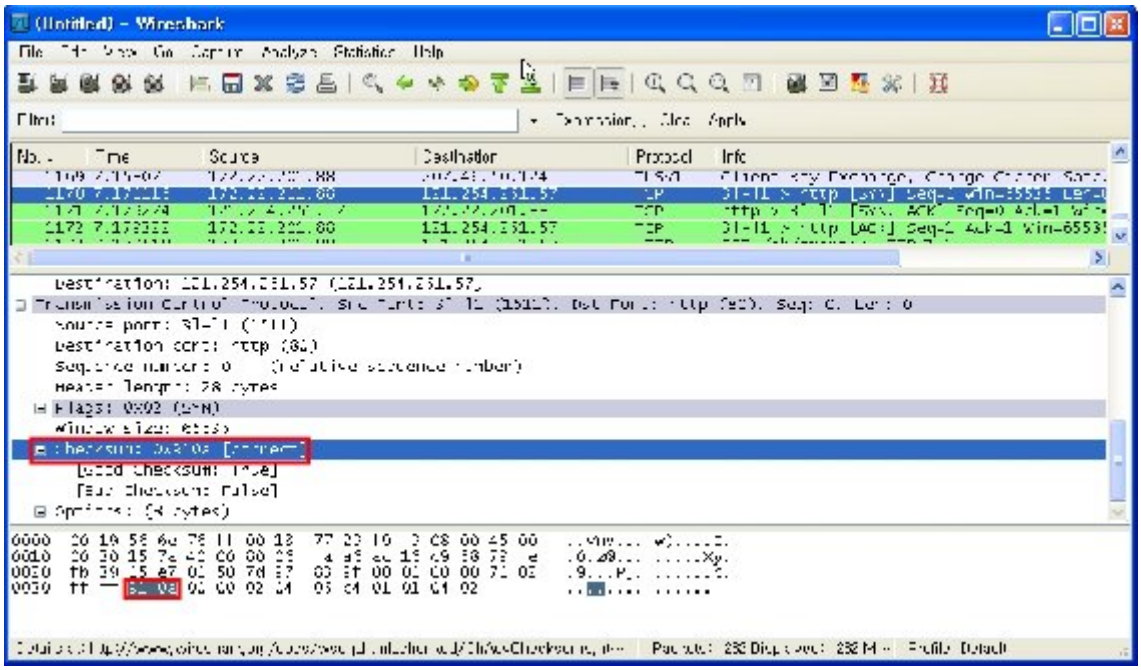
2. analysis_1 풀이

1번 문제를 풀어 보기로 하고 Wireshark로 파일을 열어보니 아래와 같은 화면이 나왔다.



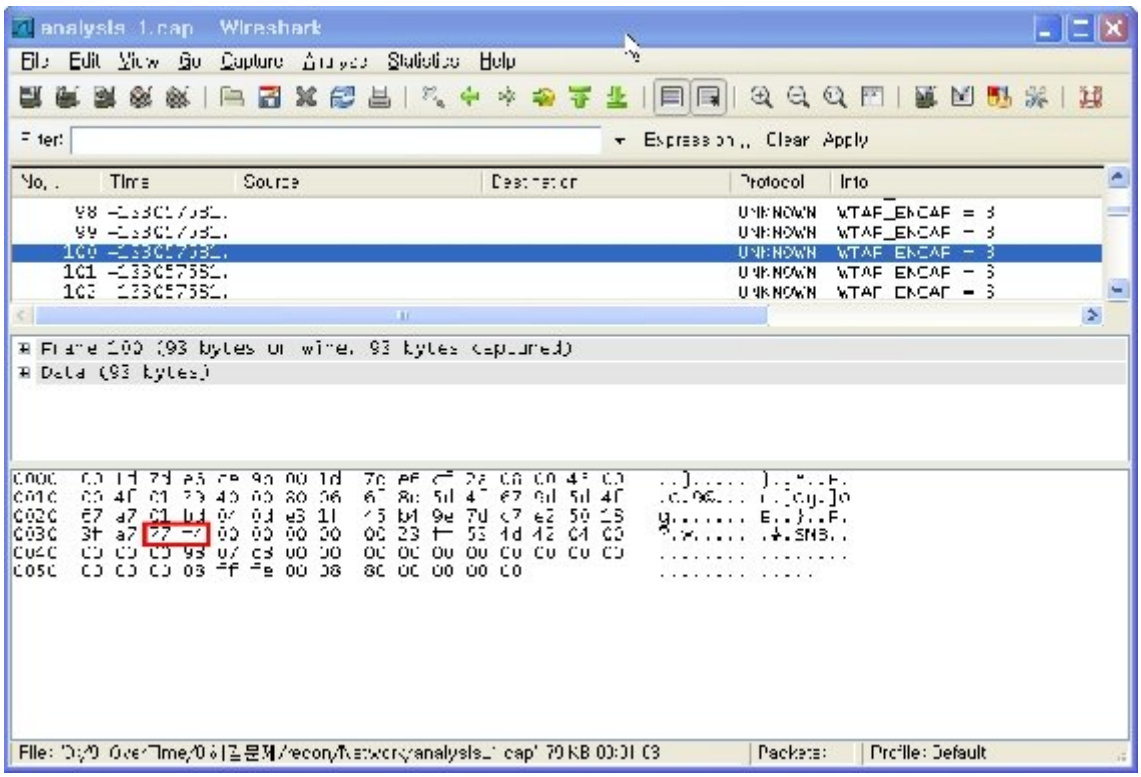
역시 뭔가 쉽게 끝나지 않을 거란 생각이 든다.
Protocal은 UNKNOWN 이고 Info는 WTAP_ENCAP = 3 라는 정보만 나올 뿐 헤더의 상세 정보는 분석 할 수 없게 되어 있었다.
뭔가에 의해 encapsulation되어 헤더 정보를 Wireshark에서 분석 할 수 없게 된 것을 보고 실제 패킷을 켑쳐 한 후 패킷 정보가 어떻게 들어 가는지

확인을 해본다.



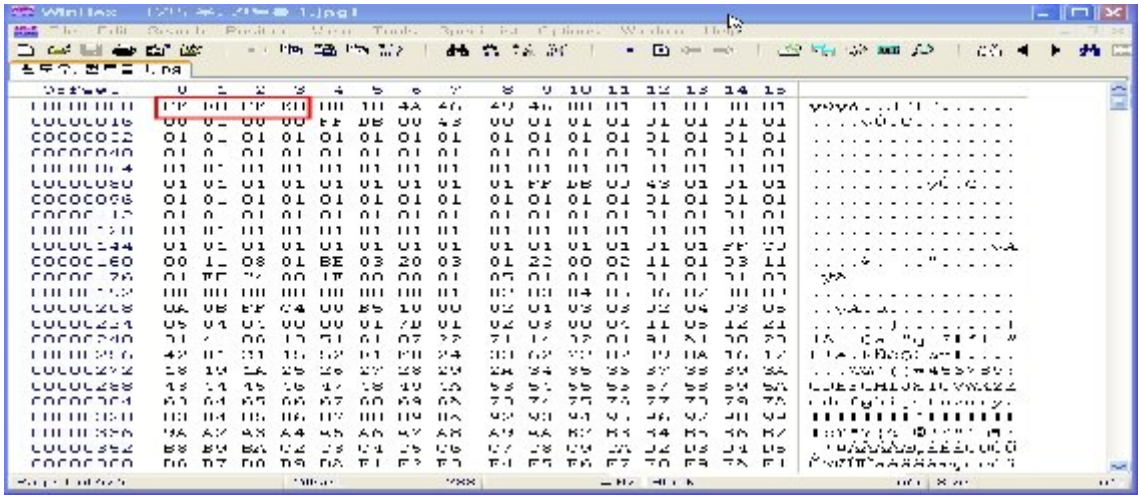
TCP헤더의 50번째 바이트 다음의 2바이트가 Checksum값 임을 알 수 있었다.

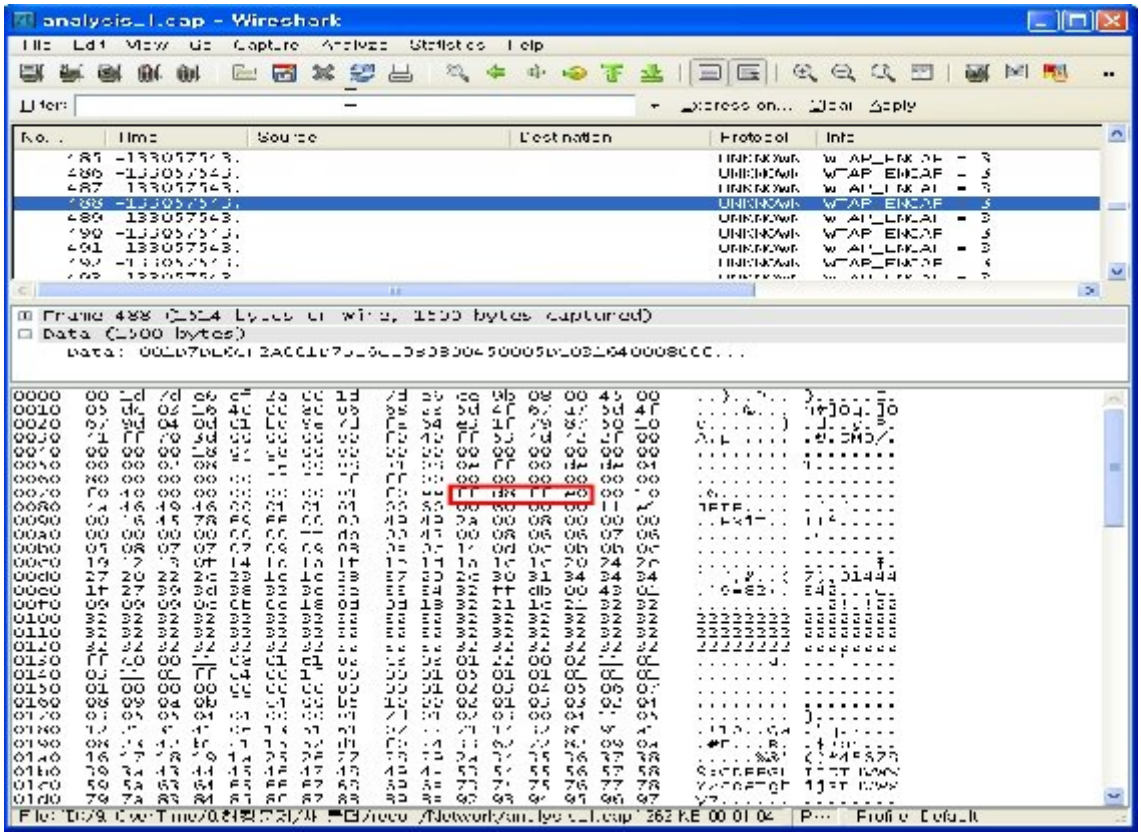
이렇게 해서 아래 와 같이 100번째 패킷을 열어보니 Checksum값이 16진수 77f4 라는 걸 알 수 있었다. 10진수로 변환하니 30708 이라는 값이 나타났다.



하나는 해결 했고 다음 문제를 또 풀기 위해 패킷을 하나씩 확인해 나갔다. 패킷을 하나씩 쭉 읽어 나가 던 중 488번째 패킷 부터 어떤 파일이 전송되는 것을 확인 할 수 있었다.

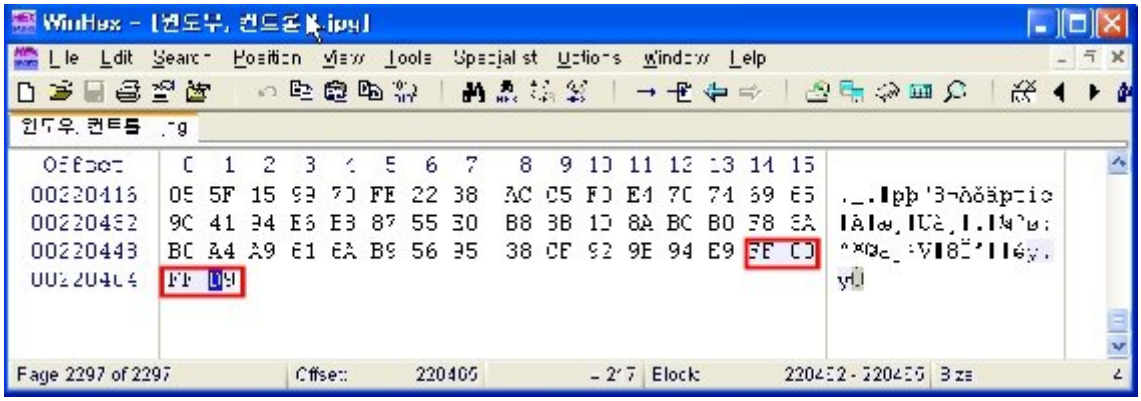
488번 패킷을 상세히 보니 어디 선가 많이 본듯한 것을 알 수 있었다. 그래서 몇개의 파일들을 WinHex로 열어 본다. JPG파일을 열어 보는 순간 488번째 패킷과 헤더가 FFD8FFE0로 일치 하는 것을 알 수 있었다.





아하! JPG 파일 형식으로 전송되는 파일을 보면 그림에 답일 일을 것 같은

생각이 들었다. 그럼 이 여러 개로 쪼개져서 가는 패킷의 끝이 어딘지 알기 위해 JPG파일을 마지막을 확인한다.



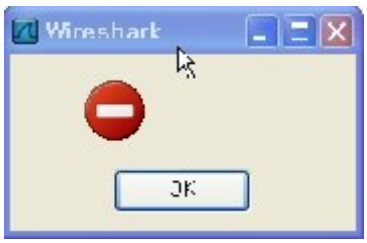
확인결과 끝은 FF00FFD9로 끝이 났다. 결과적으로 JPG파일은 FFD8 ~ FFD9 와 같은 형식으로 되어 있었다.
WinHax라 덤프뜬 내용을 Data부분만 잘라서 필요없는 부분을 버리고 JPG 파일을 만들어 보았다. 아~ 인내심의 한계!!
잘라 붙이는 곳이 문제 인지 JPG 파일이 깨끗하게 나오지 않았다.

결국 checksum이 30708 라는 단서를 가지고 brute force attack으로 압축파일 패스워드를 찾아 보기로 했다. 어차피 두 값을 더하는 거니까 30708 부터 brute force attack을 시도 해보았다. 툴은 그냥 알집에 있는 패스워드 찾기로 돌렸다.
하하 5분도 되지 않아 값이 나왔다. 압축파일 패스워드가 30796 이었다.
그럼 그림 파일의 답은 88이겠거니 하고 그냥 넘어 갔다. JPG파일이 궁금했다.

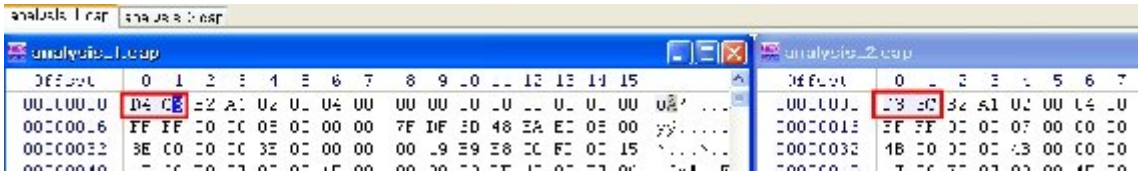
정리해보면 Checksum 값은 30708 + 88 = 30796 이라는 결론이 나왔다.

3. analysis_2 풀이

압축 파일을 열어 보니 analysis_2 파일이 있었다.
Wiresharck로 파일을 열어 보니 다음과 같은 error가 났다.

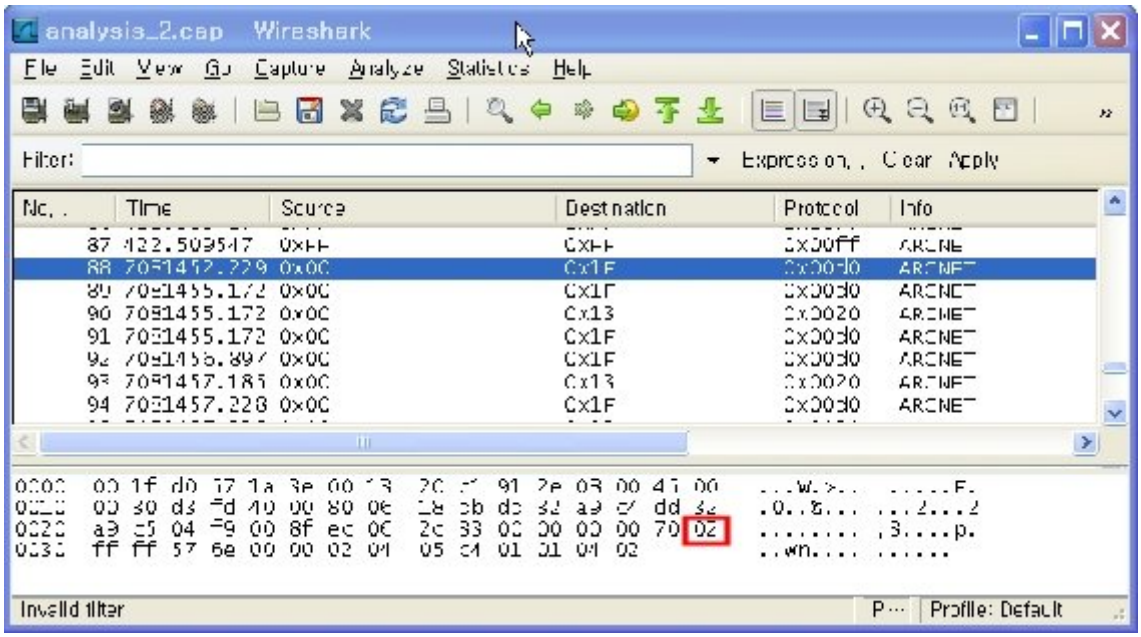


그래서 뭐가 다른지 보려고 analysis_1과 같이 WinHax로 열어서 확인해 보았다.



Analysis_1은 D4C3으로 시작 하고 Analysis_2는 D33C로 시작하는 것을 보고 Analysis_2을 D33C을 D4C3로 수정 후 열어 보니 잘 열리는 것을 알 수 있었다.

Analysis_1에서 나온 답 88이 Analysis_2의 88번째 패킷에 정답이 있다는 문제를 따라 88번 패킷을 보니 뭐 별거 없었다.
그래서 다음 다음 으로 넘어가면서 패킷을 분석한 결과 92번째 패킷에 nop Sled가 있는 것으로 보아 어떤 코드가 들어 있다는 것을 알 수 있었다.
이렇게 88번부터 92번까지의 패킷을 보니 통신과정을 나타내고 있었다.



47바이트 다음의 숫자가 TCP Flag라는 것을 Analysis_1에서와 같이 다른 패킷을 보고 확인 하였다. 이렇게 확인 결과 다음과 같은 결과가 나왔다.

88 (02)SYN
89 (02)SYN
90 (14)SYN/ACK
91 (10)ACK
92 (18)PSH/ACK

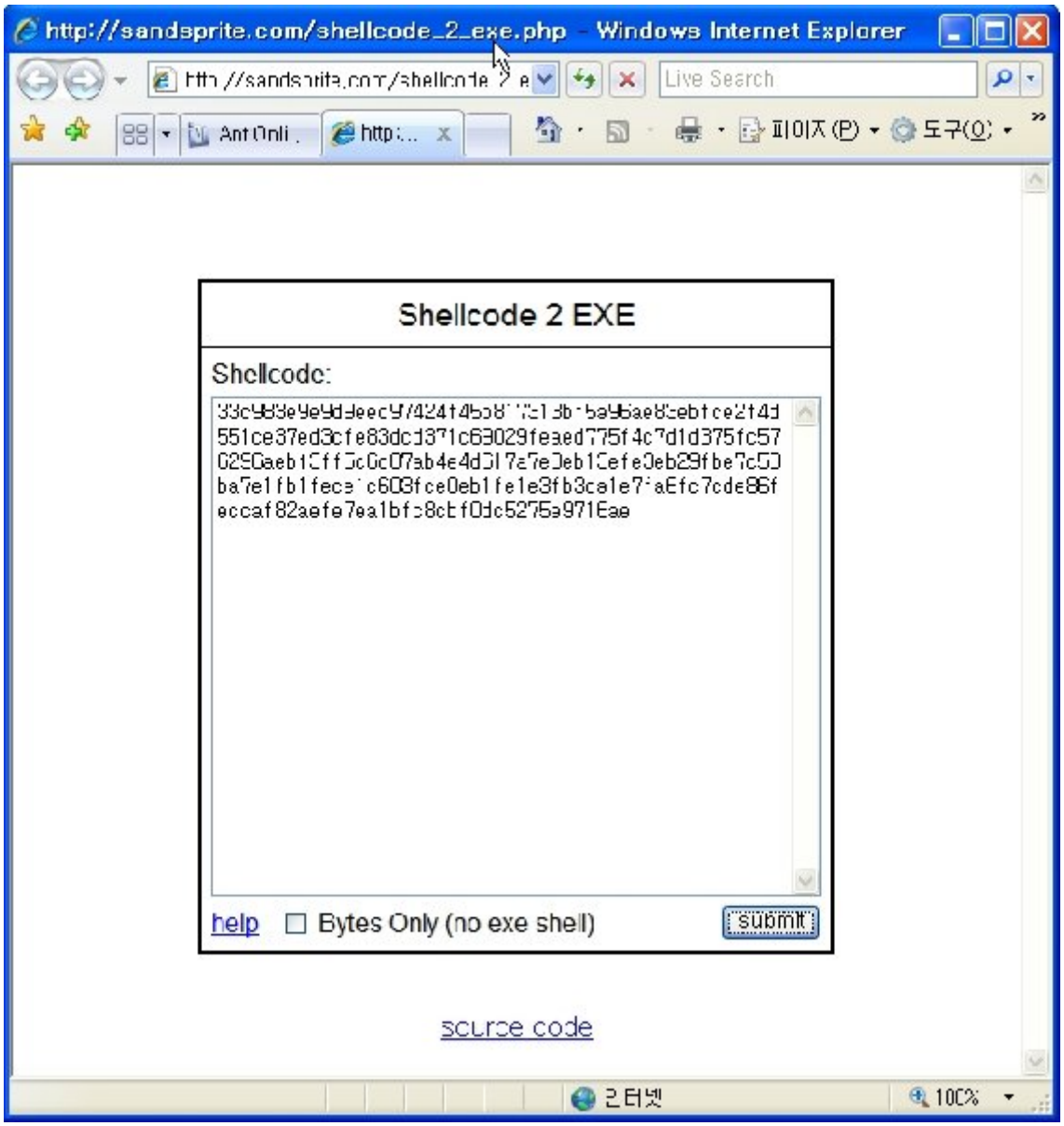
이렇게 해서 또다시 92번째 패킷에 무슨 Data를 전송 했다는 걸 알 수 있었다. 패킷을 복사 한 결과 다음과 같은 값을 얻을 수 있었다.

```
001fd0571a3e001320c1912e08004500043ed4054000800614c5dd32a9c4d
d32a9c504f9008fec062c34227747e75018ffff7fdb00002a206c6f67696e209
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
0909090909090909090909090909090909090909090909090909090909090
9090909090909090909090909090909090909090909090909090909090909
09090909090909090909090909090909090909090909090909033c983e9e9
d9eed97424f45b817313bf5a96ae83ebfce2f4d551ce37ed3cfe83dcd371c69
029feaed775f4c7d1d375fc576296aebf3ff5c6d07ab4e4d617a7e3eb13efe3e
b29fbe7c50ba7e1fb1fece1c603fce0eb1fe1e3fb3da1e7fa6fc7cde86feccaf8
```

2aefe7ea1bfc8cbf0dc5275e9716ae90909090909090909090909090909090909090909090909090909090909090909083f3ffbf88f8ffbf206261680a00

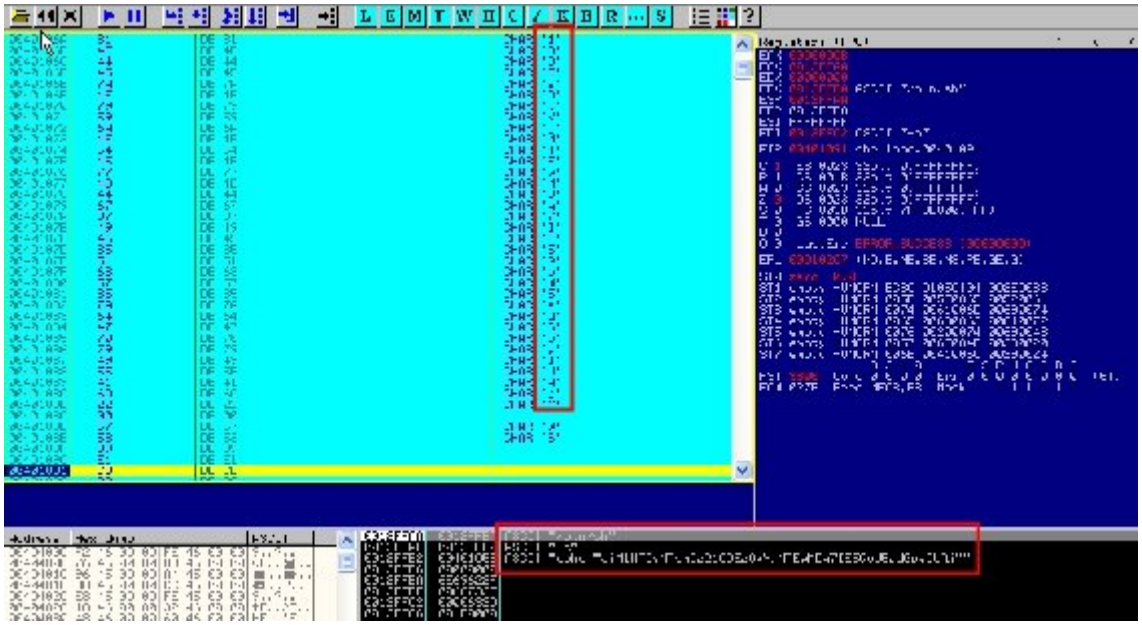
Nop Sled를 뺀 중간의 값이 어떤 코드라는 걸 알고 그 부분을 복사하여 아래의 싸이트에 가서 exe파일로 만들어 보았다

http://sandsprite.com/shellcode_2_exe.php



Submit을 누르면 shellcode.exe_ 와 같은 형식으로 다운 받을 수 있다

이렇게 받은 exe 파일을 실행 시켜보니 뭐 별다른 것을 눈으로 확인 할 수 없어 ollydbg에 넣어서 실행 시켜 보았다.

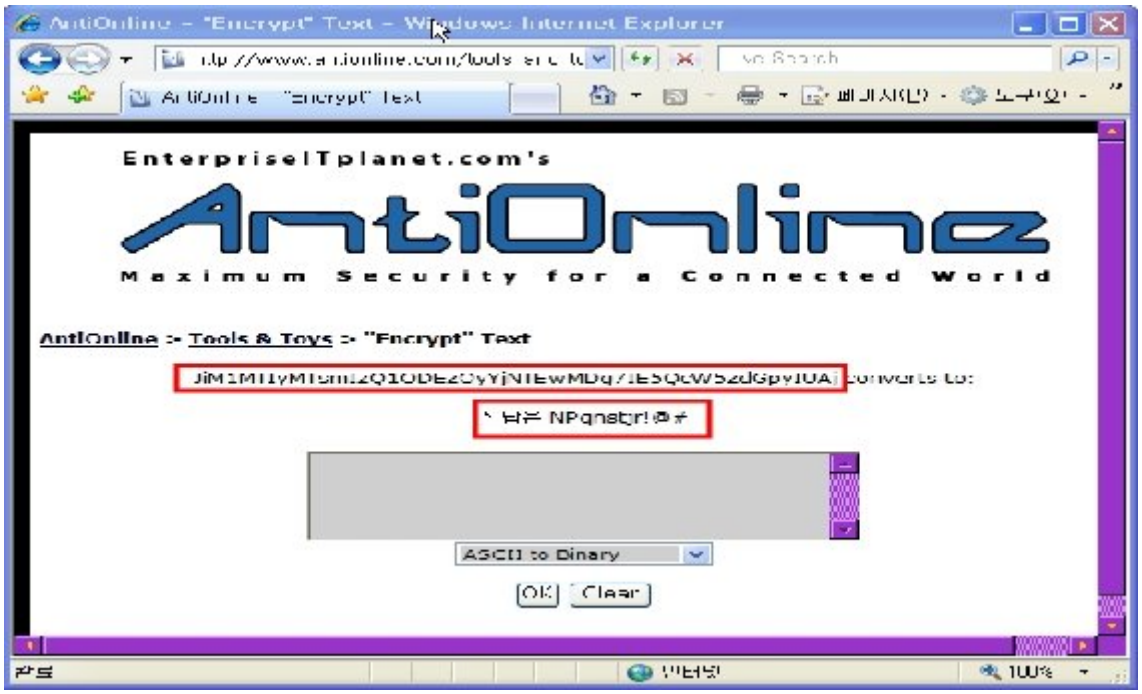


위와 같은 결과가 나왔다.

ASCII "/bin/sh"
ASCII "-c"
ASCII echo "JiM1MTIyMTsmIzQ1ODEzOyYjNTEwMDg7IE5QcW5zdGpyIUAj"

자세히 보니 echo로 뭔가를 찍은 것을 보고 echo로 찍은 값이 뭔지 보기 위해 다음의 싸이트에 가서 하나씩 decrypt 하기 시작했다.

http://www.antionline.com/tools-and-toys/encrypt-text/



차례대로 변환해본 결과 Base 64 Decode 에서 위와 같은 결과를 얻을 수 있었다.

친절 하게도 정답은 NPqnstjr!@# 이라는 결과 나타났다.