

Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

## [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-02/0067.html>

---

*From:* SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

*Date:* 02/17/05

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 17 Feb 2005 16:48:11 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.secureteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.secureteam.com/maillinglist.html>

-----

Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

---

### SUMMARY

### DETAILS

Findjmp2 is a modified version of Findjmp from eEye.com to find jmp, call, push in a loaded DLL. This version includes search for pop/pop/ret set of instructions that is useful to bypass Windows XP SP2 and Windows 2003 stack protection mechanism.

Source Code:

/\*

Findjmp.c

written by Ryan Permeh – ryan at eeye – Summarily modified by I2S-LaB.com

<http://www.eeye.com>

Findjmp2.c (pop/pop/ret scanner, logging to file)

version by A.D – class101 at hat-squad

<http://class101.org>, <http://www.hat-squad.com>

This finds useful jump points in a dll. Once you overflow a buffer, by looking in the various registers, it is likely that you will find a

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

reference to your code. This program will find addresses suitable to overwrite eip that will return to your code.

It should be easy to modify this to search for other good jump points, or specific code patterns within a dll.

It currently supports looking for:

1. jmp reg
2. call reg
3. push reg  
ret

All three options result in the same thing, EIP being set to reg.

It also supports the following registers:

EAX  
EBX  
ECX  
EDX  
ESI  
EDI  
ESP  
EBP

\*/

```
#include <iostream.h>
#include <fstream.h>
#include <Windows.h>
#include <stdio.h>
```

```
FILE *fplog;
```

```
void usage();
void sep();
void iok(BYTE *curpos, char *reg);
void iok2(BYTE *curpos, char *reg);
void ook(BYTE *curpos, char *reg);
void ook2(BYTE *curpos, char *reg);
```

```
DWORD GetRegNum( char *reg );
void findjmp( char *dll, char *reg );
```

```
//This finds useful jump points in a dll. Once you overflow a buffer, by
//looking in the various registers, it is likely that you will find a
//reference to your code. This program will find addresses of suitable
//addresses of eip that will return to your code.
```

```
int main( int argc, char **argv )
{
    if( argc <= 2 )
```

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
usage();

else
{
char dll[512], //holder for the dll to look in
reg[512]; // holder for the register

if ((fplog =fopen("findjmp.txt", "r"))==NULL){
fplog =fopen("findjmp.txt", "w");}
else fplog =fopen("findjmp.txt", "a");
strncpy( dll, argv[1], 512 );
strncpy( reg, argv[2], 512 );
findjmp( dll, reg );
}
return 0;
}

//This prints the usage information.

void usage()
{
printf("\nFindjmp, Eeye, I2S-LaB\nFindjmp2, Hat-Squad\nFindJmp DLL
registre\nEx: findjmp KERNEL32.DLL esp\"
\nCurrently supported registre are: EAX, EBX, ECX, EDX, ESI, EDI,
ESP, EBP\n" );
}

//findjmp is the workhorse. it loads the requested dll, and searches for
//the specific patterns for jmp reg, push reg ret, and call reg

void findjmp( char *dll,char *reg )
{
char reg1[]="eax";char reg2[]="ebx";
char reg3[]="ecx";char reg4[]="edx";
char reg5[]="esi";char reg6[]="edi";
char reg7[]="esp";char reg8[]="ebp";

BYTE jmppat[8][2]={ { 0xFF, 0xE0 }, { 0xFF, 0xE3 }, { 0xFF, 0xE1 }, {
0xFF, 0xE2 },
{ 0xFF, 0xE6 }, { 0xFF, 0xE7 }, { 0xFF, 0xE4 }, { 0xFF, 0xE5 } };
// patterns for jmp ops

BYTE callpat[8][2]={ { 0xFF, 0xD0 }, { 0xFF, 0xD3 }, { 0xFF, 0xD1 }, {
0xFF, 0xD2 },
{ 0xFF, 0xD6 }, { 0xFF, 0xD7 }, { 0xFF, 0xD4 }, { 0xFF, 0xD5 } };
// patterns for call ops

BYTE pushretpat[8][2]={ { 0x50, 0xC3 }, { 0x53, 0xC3 }, { 0x51, 0xC3 }, {
0x52, 0xC3 },
{ 0x56, 0xC3 }, { 0x57, 0xC3 }, { 0x54, 0xC3 }, { 0x55, 0xC3 } };
// patterns for pushret ops
```

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
BYTE poppat[8][1]={ { 0x58 }, { 0x5B }, { 0x59 }, { 0x5A }, // patterns
for pop,pop,ret
    { 0x5E }, { 0x5F }, { 0x5C }, { 0x5D },,};

BYTE retn[1][1]={ 0xC3 }; // pattern for pop,pop,ret

BYTE retnbis[1][1]={ 0xC2 }; // pattern for pop,pop,ret

HMODULE loadedDLL; //base pointer for the loaded DLL

BYTE *curpos; //current position within the DLL
BYTE *curpos2; //subposition pop,pop,ret

DWORD regnum=GetRegNum(reg); // decimal representation of passed register
DWORD regnum1=GetRegNum(reg1);DWORD regnum2=GetRegNum(reg2);
DWORD regnum3=GetRegNum(reg3);DWORD regnum4=GetRegNum(reg4);
DWORD regnum5=GetRegNum(reg5);DWORD regnum6=GetRegNum(reg6);
DWORD regnum7=GetRegNum(reg7);DWORD regnum8=GetRegNum(reg8);

DWORD numaddr=0; //accumulator for addresses

if( regnum == -1 ) //check if register is useable
{ //it didn't load, time to bail
printf( "There was a problem understanding the register.\n"\
    "Please check that it isa correct IA32 register name\n"\
    "Currently supported are:\n "\
    "EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP\n"\
    );

exit(-1);
}

if( (loadedDLL=LoadLibraryA(dll)) == NULL) // check if DLL loaded
correctly
{ //it didn't load, time to bail
printf( "There was a problem Loading the requested DLL.\n"\
    "Please check that it is in your path and readable\n" );
exit(-1);
}
else
{
sep();
fprintf(fplog,"Findjmp, Eeye, I2S-LaB\nFindjmp2, Hat-Squad\n");
printf("\nFindjmp, Eeye, I2S-LaB\nFindjmp2, Hat-Squad\n");
printf( "Scanning %s for code useable with the %s register\n", dll, reg
); //we loaded the dll correctly, time to scan it
fprintf(fplog,"Scanning %s for code useable with the %s register\n",
dll, reg ); //we loaded the dll correctly, time to scan it
sep();
curpos=(BYTE*)loadedDLL; //set curpos at start of DLL
curpos2=(BYTE*)loadedDLL; //pop,pop,ret subscan.
```

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
__try
{
while(1)
{
Sleep(1/10);
if( !memcmp( curpos, jmppat[regnum], 2) ) //check for jmp match
{
printf( "0x%X\tjmp %s\n", curpos, reg ); // we have a jmp match
fprintf(fplog,"0x%X\tjmp %s\n", curpos, reg ); // we have a jmp match
numaddr++;
}
else if( !memcmp( curpos, callpat[regnum],2) ) //check for call match

{
printf( "0x%X\tcall %s\n", curpos, reg ); // we have a call match
fprintf(fplog,"0x%X\tcall %s\n", curpos, reg );
numaddr++;
}
else if( !memcmp(curpos,pushretpat[regnum], 2) ) //check for push/ret
match
{
printf( "0x%X\tpush %s – ret\n", curpos, reg ); // we have a pushret
match
fprintf(fplog,"0x%X\tpush %s – ret\n", curpos, reg ); // we have a
jmp match
numaddr++;
}
else if( !memcmp(curpos,poppat[regnum], 1) ) //check for pop/pop/ret
match
{
curpos2++;
if( !memcmp(curpos2,poppat[regnum1], 1) )
{
curpos2++;
if( !memcmp(curpos2,retn, 1) )
{
iok(curpos, reg); // we have a popopret match
ook(curpos, reg); // we have a popopret match
numaddr++;
}
}
if( !memcmp(curpos2,retnbis, 1) )
{
iok2(curpos, reg); // we have a popopret match
ook2(curpos, reg); // we have a popopret match
numaddr++;
}
}
curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum2], 1) )
{
curpos2++;
```

Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
if( !memcmp(curpos2,retn, 1) )
{
  iok(curpos, reg); // we have a popopret match
  ook(curpos, reg); // we have a popopret match
  numaddr++;
}
if( !memcmp(curpos2,retnbis, 1) )
{
  iok2(curpos, reg); // we have a popopret match
  ook2(curpos, reg); // we have a popopret match
  numaddr++;
}
curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum3], 1) )
{
  curpos2++;
  if( !memcmp(curpos2,retn, 1) )
  {
    iok(curpos, reg); // we have a popopret match
    ook(curpos, reg); // we have a popopret match
    numaddr++;
  }
  if( !memcmp(curpos2,retnbis, 1) )
  {
    iok2(curpos, reg); // we have a popopret match
    ook2(curpos, reg); // we have a popopret match
    numaddr++;
  }
  curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum4], 1) )
{
  curpos2++;
  if( !memcmp(curpos2,retn, 1) )
  {
    iok(curpos, reg); // we have a popopret match
    ook(curpos, reg); // we have a popopret match
    numaddr++;
  }
  if( !memcmp(curpos2,retnbis, 1) )
  {
    iok2(curpos, reg); // we have a popopret match
    ook2(curpos, reg); // we have a popopret match
    numaddr++;
  }
  curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum5], 1) )
{
  curpos2++;
```

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
if( !memcmp(curpos2,retn, 1) )
{
iok(curpos, reg); // we have a popopret match
ook(curpos, reg); // we have a popopret match
numaddr++;
}
if( !memcmp(curpos2,retnbis, 1) )
{
iok2(curpos, reg); // we have a popopret match
ook2(curpos, reg); // we have a popopret match
numaddr++;
}
curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum6], 1) )
{
curpos2++;
if( !memcmp(curpos2,retn, 1) )
{
iok(curpos, reg); // we have a popopret match
ook(curpos, reg); // we have a popopret match
numaddr++;
}
}
if( !memcmp(curpos2,retnbis, 1) )
{
iok2(curpos, reg); // we have a popopret match
ook2(curpos, reg); // we have a popopret match
numaddr++;
}
curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum7], 1) )
{
curpos2++;
if( !memcmp(curpos2,retn, 1) )
{
iok(curpos, reg); // we have a popopret match
ook(curpos, reg); // we have a popopret match
numaddr++;
}
}
if( !memcmp(curpos2,retnbis, 1) )
{
iok2(curpos, reg); // we have a popopret match
ook2(curpos, reg); // we have a popopret match
numaddr++;
}
curpos2--;curpos2--;goto loop;
}
if( !memcmp(curpos2,poppat[regnum8], 1) )
{
curpos2++;
```

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
if( !memcmp(curpos2,retn, 1) )
{
    iok(curpos, reg); // we have a popopret match
    ook(curpos, reg); // we have a popopret match
    numaddr++;
}
if( !memcmp(curpos2,retnbis, 1) )
{
    iok2(curpos, reg); // we have a popopret match
    ook2(curpos, reg); // we have a popopret match
    numaddr++;
}
curpos2--;curpos2--;goto loop;
}
curpos2--;
}
loop:
    curpos++;
    curpos2++;
}
}
__except(1)
{
    sep();
    fprintf( fplog,"Finished Scanning %s for code useable with the %s
register\n", dll, reg );
    printf( "Finished Scanning %s for code useable with the %s register\n",
dll, reg );
    printf( "Found %d usable addresses\n", numaddr );
    fprintf( fplog,"Found %d usable addresses\n", numaddr);sep();fprintf(
fplog,"\n\n\n");
}
}

}

DWORD GetRegNum( char *reg )
{
    DWORD ret=-1;
    if( !strcmp( reg, "eax" ) )
    {
        ret=0;
    }
    else if( !strcmp( reg, "ebx" ) )
    {
        ret=1;
    }
    else if( !strcmp( reg, "ecx" ) )
    {
        ret=2;
    }
}
```

## Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
else if( !strcmp( reg, "edx" )
{
ret=3;
}
else if( !strcmp( reg, "esi" )
{
ret=4;
}
else if( !strcmp( reg, "edi" )
{
ret=5;
}
else if( !strcmp( reg, "esp" )
{
ret=6;
}
else if( !strcmp( reg, "ebp" )
{
ret=7;
}

return ret; //return our decimal register number
}

void sep()
{

fprintf(fplog,"-----\n")
}

void iok(BYTE *curpos, char *reg)
{
printf( "0x%X\tpop %s – pop – ret\n", curpos, reg ); // we have a
popopret match
}

void iok2(BYTE *curpos, char *reg)
{
printf( "0x%X\tpop %s – pop – retbis\n", curpos, reg ); // we have a
popopret match
}

void ook(BYTE *curpos, char *reg)
{
fprintf(fplog,"0x%X\tpop %s – pop – ret\n", curpos, reg ); // we have a
jmp match
}

void ook2(BYTE *curpos, char *reg)
{
fprintf(fplog,"0x%X\tpop %s – pop – retbis\n", curpos, reg ); // we have
```

Securiteam: [TOOL] Findjmp2 – Find jmp, call, push in a Loaded DLL (With pop/pop/ret)

```
a jmp match  
}
```

```
// EOF
```

Download information:

The original source code and binary can be found at:

<http://www.hat-squad.com/~class101/36/55/002/Findjmp2.zip>

<http://www.hat-squad.com/~class101/36/55/002/Findjmp2.zip> (ZIP Passwd:  
byclass101).

#### ADDITIONAL INFORMATION

To keep updated with the tool visit the project's homepage at:

<http://www.hat-squad.com/en/000157.html>

<http://www.hat-squad.com/en/000157.html>

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

[list-unsubscribe@securiteam.com](mailto:list-unsubscribe@securiteam.com)

In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto:list-subscribe@securiteam.com)

=====

=====

#### DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.