MOAUB

Abysssec Research

1) Advisory information

Title	: Adobe Acrobat and Reader "newclass" invalid pointer vulnerability
Version	: <= adobe reader 9.3.2
Discovery	: http://www.abysssec.com
Vendor	: http://www.adobe.com
Impact	: Ciritical
Contact	: shahin [at] abysssec.com , info [at] abysssec.com
Twitter	: @abysssec
CVE	: CVE-2010-1297

2) Vulnerable version

S.u.S.E. SUSE Linux Enterprise Desktop 11 SP1
+ Linux kernel 2.6.5
S.u.S.E. SUSE Linux Enterprise Desktop 11
S.u.S.E. SUSE Linux Enterprise Desktop 10 SP3
S.u.S.E. SUSE Linux Enterprise 11 SP1
S.u.S.E. SUSE Linux Enterprise 10 SP3
S.u.S.E. openSUSE 11.2
S.u.S.E. openSUSE 11.1
S.u.S.E. openSUSE 11.0
RedHat Enterprise Linux WS Extras 4
RedHat Enterprise Linux Supplementary 5 server
RedHat Enterprise Linux Extras 4
RedHat Enterprise Linux ES Extras 4
RedHat Enterprise Linux Desktop Supplementary 5 client
RedHat Enterprise Linux AS Extras 4
RedHat Desktop Extras 4
Pardus Linux 2009 0
HP Systems Insight Manager C.05.00.02
HP Systems Insight Manager C 05.00.02
HP Systems Insight Manager 6.0.0.96

HP Systems Insight Manager 5.3 Update 1
HP Systems Insight Manager 5.3
HP Systems Insight Manager 5.2 SP2
HP Systems Insight Manager 5.1 SP1
HP Systems Insight Manager 5.0 SP6
HP Systems Insight Manager 5.0 SP5
HP Systems Insight Manager 5.0 SP3
HP Systems Insight Manager 5.0 SP2
HP Systems Insight Manager 5.0 SP1
HP Systems Insight Manager 5.0
Adobe Reader 9.3.2
Adobe Reader 9.3.1
Adobe Reader 9.1.3
Adobe Reader 9.1.2
Adobe Reader 9.1.1
Adobe Reader 9.3
Adobe Reader 9.2
Adobe Reader 9.1
Adobe Reader 9
Adobe Flex 4.0
Adobe Flex 3.0
Adobe Flash Player Plugin 9.0.31 .0
Adobe Flash Player Plugin 9.0.28 .0
Adobe Flash Player Plugin 9.0.20 .0
Adobe Flash Player Plugin 9.0.16
Adobe Flash Player Plugin 9.0.45.0
Adobe Flash Player Plugin 9.0.18d60
Adobe Flash Player Plugin 9.0.124.0
Adobe Flash Player Plugin 9.0.112.0
Adobe Flash Player Plugin 10.0.12.10
Adobe Flash Player 10.1.51 .66
Adobe Flash Player 10.0.45 2
Adobe Flash Player 10.0.32 18
Adobe Flash Player 10.0.22 .87
Adobe Flash Player 10.0.15 .3
Adobe Flash Player 10.0.12 .36
Adobe Flash Player 10.0.12 .35
Adobe Flash Player 9.0.262
Adobe Flash Player 9.0.246 0
Adobe Flash Player 9.0.152 .0
Adobe Flash Player 9.0.151 .0
Adobe Flash Player 9.0.124 .0
Adobe Flash Player 9.0.48.0
Adobe Flash Player 9.0.47.0
Adobe Flash Player 9.0.45.0
Adobe Flash Player 9.0.31.0
Adobe Flash Player 9.0.28.0
Adobe Flash Player 9.0.260.0

Adobe Flash Player 9.0.246.0
Adobe Flash Player 9.0.159.0
Adobe Flash Player 9.0.115.0
Adobe Flash Player 9
Adobe Flash Player 10.0.42.34
Adobe Flash Player 10
Adobe Flash CS5 Professional 0
Adobe Flash CS4 Professional 0
Adobe Flash CS3 Professional 0
Adobe AIR 1.5.3 .9130
Adobe Acrobat Standard 9.3.2
Adobe Acrobat Standard 9.3.1
Adobe Acrobat Standard 9.1.3
Adobe Acrobat Standard 9.1.2
Adobe Acrobat Standard 9.3
Adobe Acrobat Standard 9.2
Adobe Acrobat Standard 9.1
Adobe Acrobat Standard 9
Adobe Acrobat Professional 9.3.2
Adobe Acrobat Professional 9.3.1
Adobe Acrobat Professional 9.1.3
Adobe Acrobat Professional 9.1.2
Adobe Acrobat Professional 9.3
Adobe Acrobat Professional 9.2
Adobe Acrobat Professional 9.1
Adobe Acrobat Professional 9
Adobe Acrobat 9.3.2
Adobe Acrobat 9.3.1
Adobe Acrobat 9.1.1
Adobe Acrobat 9.3
Adobe Acrobat 9.2

3) Vulnerability information

Class

1- Code execution

Impact

Attackers can exploit this issue to execute arbitrary code or cause denial-of-service conditions.

Remotely Exploitable

Yes

Locally Exploitable

Yes

4) Vulnerabilities detail

authplay.dll is responsible for processing flash contents in pdf files. Through processing of the newclass(bytecode 0x58) command it faces a memory corruption error.

By running the newfunction command, a new class will be created. This command takes an argument. The value of this argument is an index from classinfo structure. (For further information about this command refer to ActionScript Virtual Machine 2 (AVM2) Overview).

Here is part of the code in the sub_30292F10 function that process this command:

```
.text:30242DF1          lea    edx, [esp+18h+arg_4] ; jumptable 30242ACB case 84
.text:30242DF5          push   edx
.text:30242DF6          call   sub_301C82B0
.text:30242DFB          mov    ecx, [esp+1Ch+arg_10]
.text:30242DFF          mov    edx, [ecx+9Ch]
.text:30242E05          mov    eax, [edx+eax*4]
.text:30242E08          mov    ecx, [esp+1Ch+arg_0]
.text:30242E0C          add    esp, 4
.text:30242E0F          push   eax
.text:30242E10          mov    eax, ds:off_303F8088[esi*4]
.text:30242E17          push   offset asc_30362C14 ; " "
.text:30242E1C          push   eax
.text:30242E1D          call   sub_3025BF20
.text:30242E22          mov    ecx, eax
.text:30242E24          call   sub_3025BF20
.text:30242E29          mov    ecx, eax
.text:30242E2B          call   sub_3025C2B0
.text:30242E30          pop    edi
```

```
.text:30242E31              pop     esi
.text:30242E32              pop     ebp
.text:30242E33              pop     ebx
.text:30242E34              add     esp, 8
.text:30242E37              retn    14h
```

At the beginning of this code sub_301C82B0 is called. This function takes a pointer to the buffer that contains newclass command as an argument:

```
.text:301C82B0              push    esi
.text:301C82B1              mov     esi, [esp+4+arg_0]
.text:301C82B5              mov     ecx, [esi]
.text:301C82B7              movzx   eax, byte ptr [ecx]
.text:301C82BA              test    al, al
.text:301C82BC              js      short loc_301C82C3
.text:301C82BE              inc     ecx
.text:301C82BF              mov     [esi], ecx
.text:301C82C1              pop     esi
.text:301C82C2              retn
.text:301C82C3
.text:301C82C3 loc_301C82C3:                        ; CODE XREF: sub_301C82B0+Cj
.text:301C82C3              movzx   edx, byte ptr [ecx+1]
.text:301C82C7              shl     edx, 7
.text:301C82CA              and     eax, 7Fh
.text:301C82CD              or      edx, eax
.text:301C82CF              test    edx, 4000h
.text:301C82D5              jnz     short loc_301C82E0
.text:301C82D7              add     ecx, 2
.text:301C82DA              mov     [esi], ecx
.text:301C82DC              mov     eax, edx
.text:301C82DE              pop     esi
.text:301C82DF              retn
....
```

In this function the first byte after bytecode 58 which is equal to newclass command is read. If it is greater than zero the next bytes also will be read. The value of the second byte is multiplied by 128 and added with the value of the first byte. If the result is greater than 16384 it will go on the third byte. This process is continued until the fifth bye after bytecode 0x58.

There problem here is not properly checking these values. sub_301C82B0 functions return the above result. After executing the sub_301C82B0 function remaining code will be followed in sub_30292F10 function. then value of edx is added to the return value of sub_301C82B0 function and is stored in a buffer.

A little later sub_3025C2B0 function is called:

```
.text:3025C2B0          push   esi
.text:3025C2B1          mov    esi, ecx
.text:3025C2B3          mov    ecx, [esp+4+arg_0]
.text:3025C2B7          test   ecx, ecx
.text:3025C2B9          jz     short loc_3025C2D2
.text:3025C2BB          mov    eax, [ecx]
.text:3025C2BD          mov    edx, [esi+0Ch]
.text:3025C2C0          mov    eax, [eax+8]
.text:3025C2C3          push   edx
.text:3025C2C4          call   eax
```

sub_3025C2B0 function takes the returned value of vulnerable function as its only argument. Value of eax register is called and because value of this register is related to its argument so it is possible to change to any address.

Exploit

Exploiting this bug is difficult but possible because o the DEP (permanent) in Adobe Reader. According to the above explanation I will present the way of exploitation.

As we discussed sub_301C82B0 function return some controllable value:

```
.text:30242AEA          call   sub_301C82B0
.text:30242AEF          mov    edi, [esp+1Ch+arg_10]
.text:30242AF3          mov    esi, eax
.text:30242AF5          mov    eax, [edi+38h]
.text:30242AF8          mov    eax, [eax+esi*4]
```

We should set values after bytecode 0x58 which in result the return value of sub_301C82B0 and finally result of [edx+eax*4]expression direct us to our controllable code. To reach this point we change 5byes after bytecode 0x58 so edx+eax*4 expression points to controllable data. Our controllable data can be name of the class which is a long string.

Check http://www.exploit-db.com/exploits/14853/ for a full PoC.