MOAUB

Abysssec Research

1) Advisory information

Title	: Adobe Acrobat and Reader "pushstring" Memory Corruption
Version	: Adobe Reader 9.3.2
Analysis	: http://www.abysssec.com
Vendor	: http://www.adobe.com
Impact	: Med/High
Contact	: shahin [at] abysssec.com , info [at] abysssec.com
Twitter	: @abysssec
CVE	: CVE-2010-2201

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. 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
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 8.2.2
Adobe Reader 8.2.1
Adobe Reader 8.1.7
Adobe Reader 8.1.6
Adobe Reader 8.1.5
Adobe Reader 8.1.4
Adobe Reader 8.1.3
Adobe Reader 8.1.2
Adobe Reader 8.1.1
Adobe Reader 7.1.4
Adobe Reader 7.1.3
Adobe Reader 7.1.2
Adobe Reader 7.1.1
Adobe Reader 7.0.9
Adobe Reader 7.0.8
Adobe Reader 7.0.7
Adobe Reader 7.0.6
Adobe Reader 7.0.5
Adobe Reader 7.0.4
Adobe Reader 7.0.3
Adobe Reader 7.0.2
Adobe Reader 7.0.1
Adobe Reader 7.0
Adobe Reader 9.3
Adobe Reader 9.2
Adobe Reader 9.1
Adobe Reader 9
Adobe Reader 8.2
Adobe Reader 8.1.2 Security Updat
Adobe Reader 8.1
Adobe Reader 8.0
Adobe Reader 7.1
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 8.2.2
Adobe Acrobat Standard 8.2.1
Adobe Acrobat Standard 8.1.7
Adobe Acrobat Standard 8.1.6
Adobe Acrobat Standard 8.1.4
Adobe Acrobat Standard 8.1.3
Adobe Acrobat Standard 8.1.2
Adobe Acrobat Standard 8.1.1
Adobe Acrobat Standard 7.1.4
Adobe Acrobat Standard 7.1.3
Adobe Acrobat Standard 7.1.1
Adobe Acrobat Standard 7.0.8

Adobe Acrobat Standard 7.0.7
Adobe Acrobat Standard 7.0.6
Adobe Acrobat Standard 7.0.5
Adobe Acrobat Standard 7.0.4
Adobe Acrobat Standard 7.0.3
Adobe Acrobat Standard 7.0.2
Adobe Acrobat Standard 7.0.1
Adobe Acrobat Standard 7.0
Adobe Acrobat Standard 9.3
Adobe Acrobat Standard 9.2
Adobe Acrobat Standard 9.1
Adobe Acrobat Standard 9
Adobe Acrobat Standard 8.2
Adobe Acrobat Standard 8.1
Adobe Acrobat Standard 8.0
Adobe Acrobat Standard 7.1
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 8.2.2
Adobe Acrobat Professional 8.2.1
Adobe Acrobat Professional 8.1.7
Adobe Acrobat Professional 8.1.6
Adobe Acrobat Professional 8.1.4
Adobe Acrobat Professional 8.1.3
Adobe Acrobat Professional 8.1.2
Adobe Acrobat Professional 8.1.1
Adobe Acrobat Professional 7.1.4
Adobe Acrobat Professional 7.1.3
Adobe Acrobat Professional 7.1.1
Adobe Acrobat Professional 7.0.9
Adobe Acrobat Professional 7.0.8
Adobe Acrobat Professional 7.0.7
Adobe Acrobat Professional 7.0.6
Adobe Acrobat Professional 7.0.5
Adobe Acrobat Professional 7.0.4
Adobe Acrobat Professional 7.0.3
Adobe Acrobat Professional 7.0.2
Adobe Acrobat Professional 7.0.1
Adobe Acrobat Professional 7.0
Adobe Acrobat Professional 9.3
Adobe Acrobat Professional 9.2
Adobe Acrobat Professional 9.1
Adobe Acrobat Professional 9
Adobe Acrobat Professional 8.2
Adobe Acrobat Professional 8.1.2 Security Updat
Adobe Acrobat Professional 8.1

Adobe Acrobat Professional 8.0
Adobe Acrobat Professional 7.1
Adobe Acrobat Professional 6.0
Adobe Acrobat 9.3.2
Adobe Acrobat 9.3.1
Adobe Acrobat 9.1.1
Adobe Acrobat 8.2.2
Adobe Acrobat 7.0.9
Adobe Acrobat 7.0.3
Adobe Acrobat 7.0.2
Adobe Acrobat 7.0.1
Adobe Acrobat 7.0
Adobe Acrobat 6.0.5
Adobe Acrobat 6.0.4
Adobe Acrobat 6.0.3
Adobe Acrobat 6.0.2
Adobe Acrobat 6.0.1
Adobe Acrobat 6.0
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

This vulnerability show itself through the processing of flash contents in pdf files. The authplay.dll module which is responsible for processing flash contents in pdf during the processing of pushstring(bytecode 0x2c) face a memory corruption problem.

By executing the pushstring command , a string value is pushed on the stack. This command take an argument. The value of this argument is an index of string in the constant pool structure. (for further information about this command refer to ActionScript Virtual Machine 2 (AVM2) Overview)

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

```
.text:30242AD2         mov    ecx, ds:off_303F8088[esi*4] ; jumptable 30242ACB cases 40,237
.text:30242AD9         mov    ebx, [esp+18h+arg_0]
.text:30242ADD         push   ecx
.text:30242ADE         mov    ecx, ebx
.text:30242AE0         call   sub_3025BF20
.text:30242AE5         lea    edx, [esp+18h+arg_4]
.text:30242AE9         push   edx
.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]
.text:30242AFB         add    esp, 4
.text:30242AFE         or     eax, 2
.text:30242B01         push   eax
```

In the beginning of this code sub_301C82B0 is called. This function take an argument that is a pointer to buffer containing pushstring command 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 first byte after the bytecode 0x2c equal to pushstring command is rad. If it is greater than zero next byte is read too. Value of the next byte is multiplied by 128 and added to the value of first byte. If the result is greater than 16384(4000h) go to the third bye. This procedure continues until the fifth bye after bytecode 0x2c.

Problem here is not properly checking this value. sub_301C82B0 function return the above result. After executing of sub_301C82B0 function the execution follows in the sub_30292F10 function. And then value of eax register is added to the return value of sub_301C82B0 and is stored In some buffer.

The stored value in the buffer is under our control and is used in the next instruction which can corrupt memory or calling and invalid address.

```
.text:30241E30             push    esi
.text:30241E31             mov     esi, [esp+4+arg_0]
.text:30241E35             push    edi
....
.text:30241E47             cmp     eax, 6        ; switch 7 cases
.text:30241E4A             ja      loc_30241EE2   ; default
.text:30241E4A                                 ; jumptable 30241E50 case 2
.text:30241E50             jmp     ds:off_30241F00[eax*4] ; switch jump
.text:30241E57
.text:30241E57 loc_30241E57:                    ; DATA XREF: .text:off_30241F00o
.text:30241E57             mov     ecx, esi      ; jumptable 30241E50 case 0
.text:30241E59             and     ecx, 0FFFFFFF8h
```

```
.text:30241E5C             mov     edx, [ecx]
.text:30241E5E             mov     eax, [edx+84h]
.text:30241E64             push    edi
.text:30241E65             call    eax
```

Exploit

Exploiting this bug is difficult but possible because of 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 0x2c which in result the return value of sub_301C82B0 and finally result of [eax+esi*4] expression direct us to our controllable code. Then take the advantages of other codes that use this value to gain control of the program. After gaining control of the execution we should take the stack and bypassing the DEP by implementing the ROP method to execute the shellcode.