Novell Distributed Print Services
Remote Integer Overflow

15-May-2006

Summary

Novell Distributed Print Services (DPS) offers various printing services to networks using an RPC-based protocol between clients and servers. The library that facilitates this communication is present in all Novell products that support DPS and will be referred to as the DPRPC library.

There is a code defect in the variable-length array decoding portion of the DPRPC library. This defect is an integer overflow resulting from the multiplication of array-member count and element size. This multiplication can be abused to cause a small heap allocation and copy an arbitrary number of bytes to an arbitrary memory location. By overwriting specific data structures attackers may reliably execute arbitrary code. Also, since this flaw resides in a basic type exported from shared APIs, there are many vulnerable instances available to attackers.

Impact

These vulnerabilities are present by default in DPS software. Successful exploitation of these vulnerabilities results in remote code execution with the full privileges of the DPS-communicating process. By default, the privileges are equivalent to the super-user. Due to certain features in the DPRPC library, exploits that leverage these features can be made reliable. Further, correct network-based detection of this vulnerability requires decoding the full DPS protocol since the vulnerability resides in extensions of a basic type.

Affected software

Novell Netware (All versions)
Novell Open Enterprise Server (All NetWare based versions)
Novell Netware Client for Windows (All versions)

Credit

These vulnerabilities were researched by Ryan Smith and Alex Wheeler.

Contact

advisories@hustlelabs.com

SOME RIGHTS RESERVED

Details

The DPRPC library exports a function (ndps_xdr_array) that decodes an XDR encoded variable-length array. These arrays are encoded first with an XDR-encoded unsigned long, designating the number of elements in the array, followed by the elements of the array. Location 1 in ndps_xdr_array retrieves the number of elements in the array. Location 2 compares the value to an argument that designates the maximum number of array elements to decode, and if the value is less than or equal to the maximum value, continues processing. Next, location 3 in ndps_xdr_array multiplies the size of an array element by the number of elements in the XDR stream and subsequently allocates that amount of memory. At location 4, each element from the XDR stream is decoded into the allocated memory space.

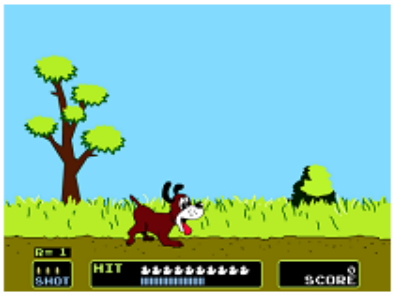
```
ndps_xdr_array  proc near

var_4           = dword ptr -4
xdr_stream      = dword ptr  8
arg_4           = dword ptr  0Ch
arg_8           = dword ptr  10h
in_ulMaxElements= dword ptr  14h
in_ulSzElement  = dword ptr  18h
in_fpDecode     = dword ptr  1Ch

                ...

                mov     ebx, [ebp+arg_8]
                push    esi
                push    edi
                mov     edi, [ebp+arg_4]
                push    ebx
                push    [ebp+xdr_stream]
                mov     [ebp+var_4], 1
                mov     esi, [edi]
                call    ndps_xdr_u_int          <-- 1
                test    eax, eax
                jz      short loc_592024E0
                mov     ebx, [ebx]
                cmp     ebx, [ebp+in_ulMaxElements]   <-- 2
                jbe     short loc_592024B4
                mov     eax, [ebp+xdr_stream]
                cmp     dword ptr [eax], 2
                jnz     short loc_592024E0

                ...

;
loc_592024C8:                           ; CODE XREF: ndps_xdr_a
                test    ebx, ebx
                jz      short loc_592024C3
                mov     eax, ebx                <-- 3
                imul    eax, [ebp+in_ulSzElement]
                push    eax
                call    allocator
                mov     esi, eax
                test    esi, esi
                mov     [edi], esi
                jnz     short loc_592024E4

                ...

loc_592024EC:                           ; CODE XREF: ndps_xdr_a
                push    esi                     <-- 4
                push    [ebp+xdr_stream]
                call    [ebp+in_fpDecode]
                add     esi, [ebp+in_ulSzElement]
                mov     [ebp+var_4], eax
                test    eax, eax
                jz      short loc_59202505
                inc     [ebp+arg_4]
                cmp     [ebp+arg_4], ebx
                jb      short loc_592024EC

                ...

ndps_xdr_array  endp
```

SOME RIGHTS RESERVED

An integer overflow may occur at location 3 when the product of element size and number of elements cannot fit within 32 bits. If an overflow does occur, it will result in a memory allocation of insufficient size.

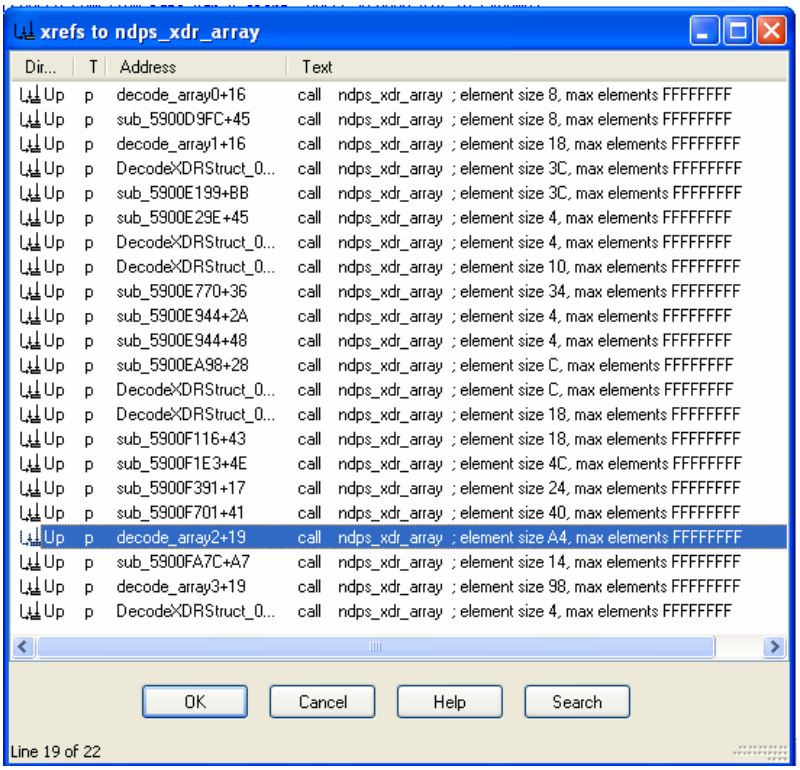
In sub_5900E770, code using the ndps_xdr_array function is not as judicious as it needs to be with the maximum elements parameter. When decoding this array of variable length structures, an integer overflow will occur in the calculation of the amount of memory required if the element count in the XDR stream is greater than 4EC4EC4.

```
sub_5900E770    proc near               ; CODE XREF: DecodeXDRStr
var_4           = dword ptr -4
arg_0           = dword ptr  4
arg_4           = dword ptr  8

                push    esi
                mov     esi, [esp+4+arg_4]
                push    esi
                push    [esp+8+arg_0]
                call    ndps_xdr_enum
                test    eax, eax
                jnz     short loc_5900E787

loc_5900E783:                           ; CODE XREF: sub_5900E770
                xor     eax, eax
                jmp     short loc_5900E7BA
; ---------------------------------------------------------------------------

loc_5900E787:                           ; CODE XREF: sub_5900E770
                mov     eax, [esi]
                sub     eax, 0
                jz      short loc_5900E7AD
                dec     eax
                jnz     short loc_5900E783
                push    offset sub_5900E74C ; #fpDecode
                push    34h             ; ulSzElement
                lea     eax, [esi+4]
                push    0FFFFFFFFh      ; ulMaxElements
                add     esi, 8
                push    eax             ; #ulSpecifiedArrayLength
                push    esi             ; #ulCntDecodedElements
                push    [esp+18h+arg_0] ; xdr_stream
                call    ndps_xdr_array
                jmp     short loc_5900E7BA
; ---------------------------------------------------------------------------

loc_5900E7AD:                           ; CODE XREF: sub_5900E770
                add     esi, 4
                push    esi
                push    [esp+8+arg_0]
                call    DecodeXDRStruct_0_2 ; enum,(bytes,byteArr

loc_5900E7BA:                           ; CODE XREF: sub_5900E770
                                        ; sub_5900E770+3B↓j
                pop     esi
                retn    8
sub_5900E770    endp
```



SOME RIGHTS RESERVED

There are many other invocations of the ndps_xdr_array decoding function that are susceptible to an integer overflow. The following image is a list of cross-references to the ndps_xdr_array function and the parameters of concern. As shown, all 22 invocations of the process importing this library are vulnerable. This is list is representative of the vulnerable invocations in only one library that uses the DPRPC library.



SOME RIGHTS RESERVED

Remediation

In order for the calculation to not result in an integer overflow, all callers of this function should pass a value for the maximum number of elements to be less-than or equal-to 0xFFFFFFFF divided by the element size (instead of just 0xFFFFFFFF). Please refer to the following links for information and patches:
Server:
http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=9145&sliceId=SAL_Public&dialogID=3455056&stateId=0%200%203453353

http://support.novell.com/cgi-bin/search/searchtid.cgi?/2973700.htm

Client:
http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=1076&sliceId=SAL_Public&dialogID=3455056&stateId=0%200%203453353

http://support.novell.com/cgi-bin/search/searchtid.cgi?/2973719.htm

Timeline of Events

01-May-2006 Vendor notification
04-May-2006 Initial patch for servers
09-May-2006 Initial patch for clients
10-May-2006 Due to miscommunication, vendor released the client patch
15-May-2006 Coordinated disclosure of vulnerability

SOME RIGHTS RESERVED

Attributions

Duck Hunt images were taken from screenshots of a flash application published at http://www.johnnyslack.com/duckhunt/.

Code and cross-reference screenshots captured using IDA (http://www.datarescue.com).

Flawed code, and patch information obtained from Novell (http://www.novell.com).

The Creative Commons license-notification image borrowed from http://www.creativecommons.org.

License

This work is licensed under the Creative Commons Attribution 2.5 License. To view a copy of this license, visit http://creativecommons.org/licenses/by/2.5/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Attribution should be provided both in the form of a link or reference to http://www.hustlelabs.com and a copy of the researchers' names listed under the *Credit* section of this document.

All other trademarks and copyrights referenced in this document are the property of their respective owners.