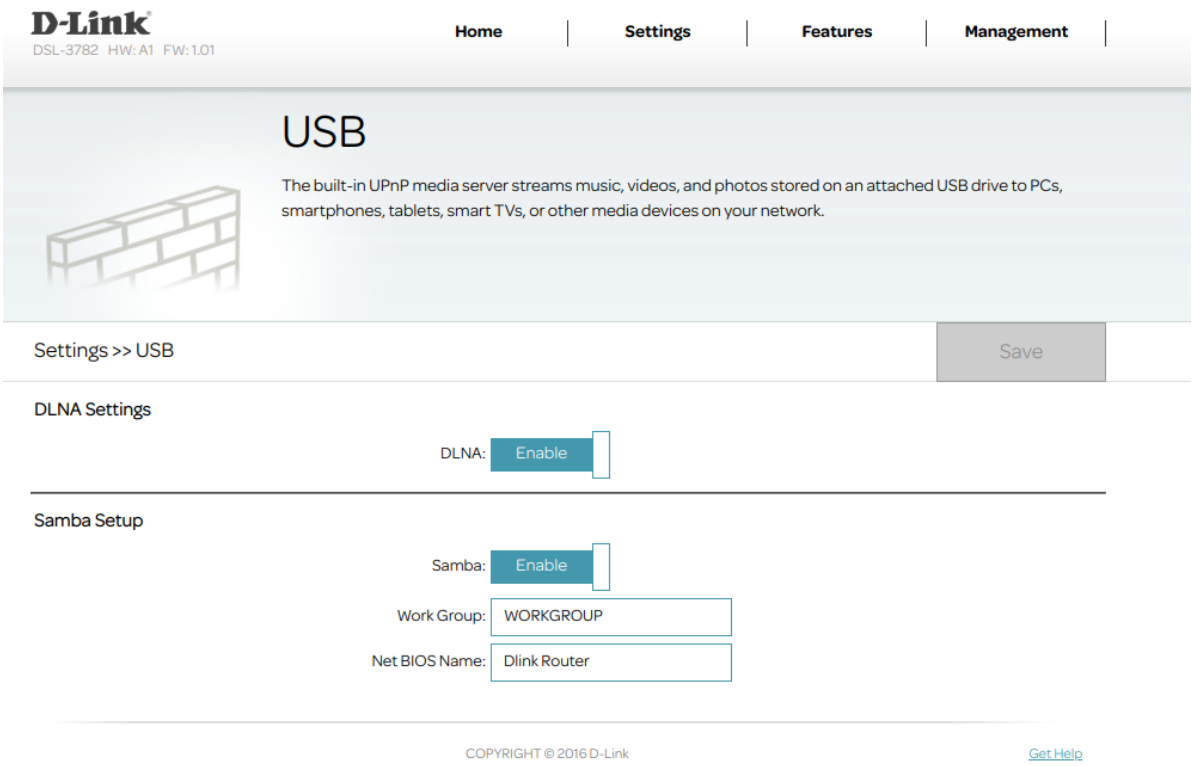
Vendor of the products:  D-Link

Affected products:  DSL-3782 v1.01

# Vulnerability Description

An OS command injection vulnerability was discovered in D-Link DSL-3782 v1.01, triggered by the samba_wg and samba_nbn parameters. This vulnerability allows attackers to execute arbitrary operating system (OS) commands via a crafted packet.

The interface that triggers the vulnerability




# POC

send


You can see the Telnet service has been successfully started and connected, which could potentially lead to a complete compromise of the application and all its data, exposing it to severe security risks.

```
└─$ telnet 192.168.1.1 9993
Trying 192.168.1.1 ...
Connected to 192.168.1.1.
Escape character is '^]'.
# ls
bin         etc         linuxrc     sbin        usr
boaroot     firmadyne   lost+found  tmp         var
dev         lib         proc        userfs
#
```

Similarly, the samba_nbn parameter can also trigger this vulnerability.


```
└─$ telnet 192.168.1.1 9992
Trying 192.168.1.1 ...
Connected to 192.168.1.1.
Escape character is '^]'.
# ls
bin         etc         linuxrc     sbin        usr
boaroot     firmadyne   lost+found  tmp         var
dev         lib         proc        userfs
```

# Code in cfg_manager

By using IDA to analyze cfg_manager, It can be seen that the samba.sh script is concatenated into the system function and executed.

```
87  }
88  if ( getAttrValue(a1, v15, "Active", v13) )
89  {
90    strcpy((char *)v13, "Yes");
91    strcpy((char *)v15, "Account");
92    strcpy((char *)v16, "Entry0");
93    if ( getAttrValue(a1, v15, "username", v17) )
94      strcpy(v17, "Admin");
95    v2 = 1;
96    if ( getAttrValue(v1, v15, "web_passwd", v18) )
97    {
98      v2 = 1;
99      strcpy(v18, "1234");
100   }
101 }
102 else
103 {
104   v2 = 0;
105 }
106 if ( strcmp((const char *)v13, "Yes") )
107   return 0;
108 v4 = fopen((const char *)v14, "w");
109 if ( !v4 )
110   return 0;
111 if ( v2 )
112 {
113   memset(v20, 0, sizeof(v20));
114   sprintf(v20, "%s %s %s %s\n", "/usr/script/samba.sh", "SambaSvr", "MyGroup", v17);
115   fputs(v20, v4);
116   memset(v20, 0, sizeof(v20));
117   sprintf(v20, "%s %s %s\n", "/userfs/bin/smbpasswd -a", v17, v18);
118   fputs(v20, v4);
119   memset(v20, 0, sizeof(v20));
120   sprintf(v20, "%s %s %s %s %s", "/usr/script/samba_add_dir.sh", "Samba_Share", "/", "Yes", "Yes");
121   fputs(v20, v4);
122 }
123 fwrite("/userfs/bin/nmbd -D\n", 1u, 0x14u, v4);
124 fwrite("/userfs/bin/smbd -D\n", 1u, 0x14u, v4);
125 fclose(v4);
126 chmod((const char *)v14, 0x309u);
127 system((const char *)v14);
128 unlink((const char *)v14);
129 return 0;
130 }
```

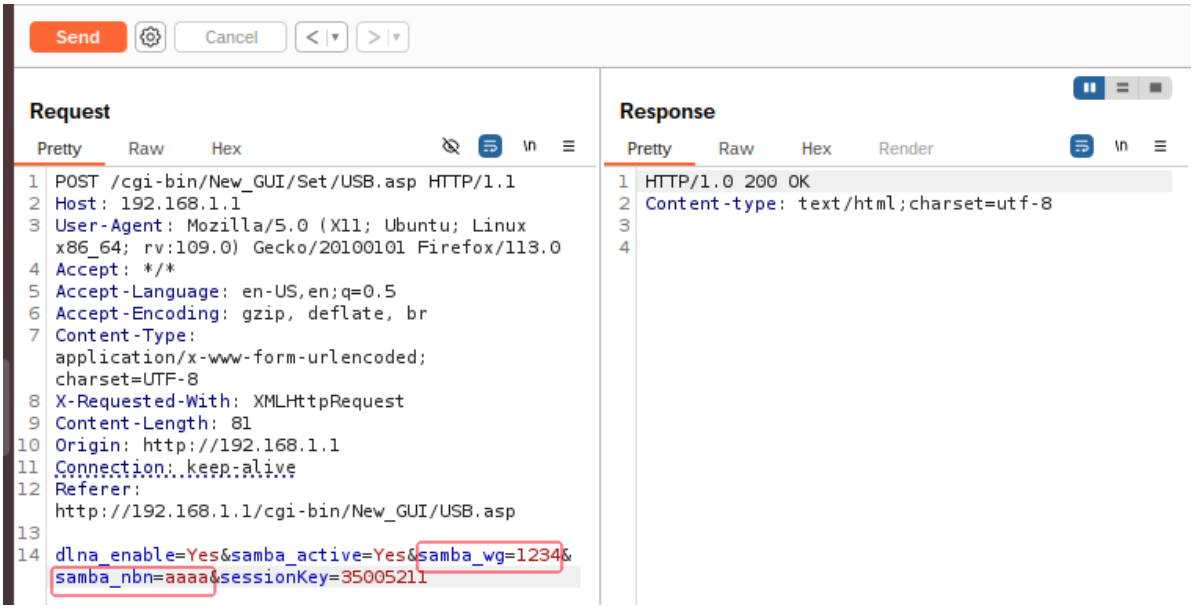
Take a look at the content of samba.sh.

```
/usr/script # cat samba.sh
#!/bin/sh

SAMBA_FILE=/etc/samba/smb.conf

if [ ! -n "$3" ]; then
        echo "insufficient arguments!"
        echo "Usage: $0 <netbios_name> <workgroup> <guest>"
        echo "Example: $0 RT2880 Ralink admin"
        exit 0
fi

NETBIOS_NAME="$1"
WORKGROUP="$2"
GUEST="$3"

echo "[global]
netbios name = $NETBIOS_NAME
server string = Samba Server
workgroup = $WORKGROUP
security = user
guest account = $GUEST
log file = /var/log.samba
socket options = SO_RCVBUF=32768 SO_SNDBUF=16384
encrypt passwords = yes
use spne go = no
client use spnego = no
disable spoolss = yes
smb passwd file = /etc/samba/smbpasswd
host msdfs = no
strict allocate = No
os level = 20
log level = 1
max log size = 50
null passwords = yes
mangling method = hash
dos charset = UTF8
unix charset = UTF8
display charset = UTF8" > $SAMBA_FILE
```

In the samba.sh script, you can see that the NETBIOS_NAME and WORKGROUP parameters are received and written to the file, and the final configuration  information is written into the smb.conf file.

Take a look at the content of smb.conf.

```
/tmp/etc/samba # cat smb.conf
[global]
netbios name = aaaa
server string = Samba Server
workgroup = 1234
security = user
guest account = admin
log file = /var/log.samba
socket options = SO_RCVBUF=32768 SO_SNDBUF=16384
encrypt passwords = yes
use spne go = no
client use spnego = no
disable spoolss = yes
smb passwd file = /etc/samba/smbpasswd
host msdfs = no
strict allocate = No
os level = 20
log level = 1
max log size = 50
null passwords = yes
mangling method = hash
dos charset = UTF8
unix charset = UTF8
display charset = UTF8
```




It can be seen that it is exactly the content we entered in the front  end, which confirms that the attacker can construct special inputs to  carry out an OS command injection.