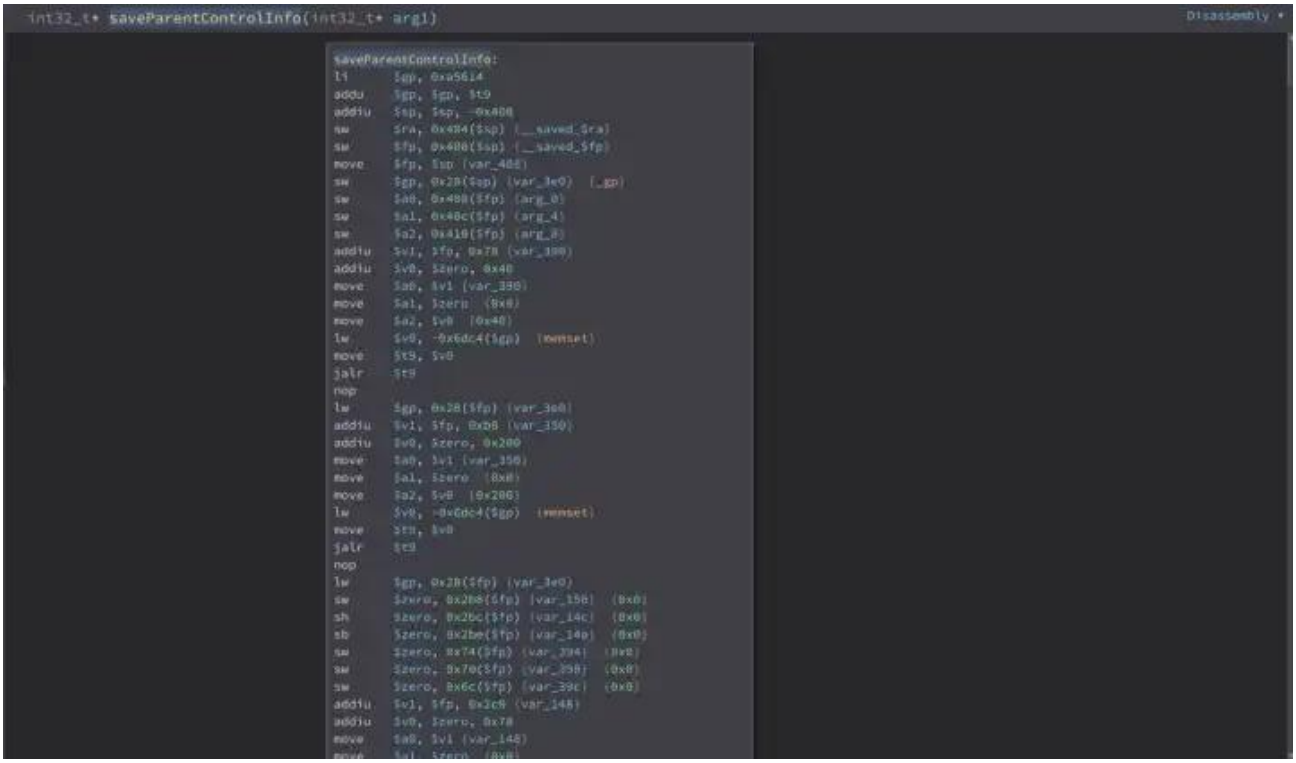
Brandon Roldan Follow
Aug 7, 2021 · 4 min read · Listen

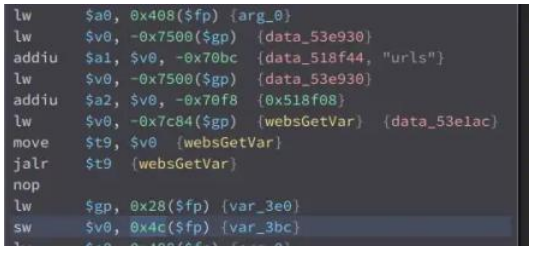
Hacking the Tenda AC10–1200 Router Part 2: Strcpy Buffer Overflow

Hi. This would be another series of writeup where we will try to hack the tenda ac10 1200 and try to get a cve. This writeup is fairly short so lets get started

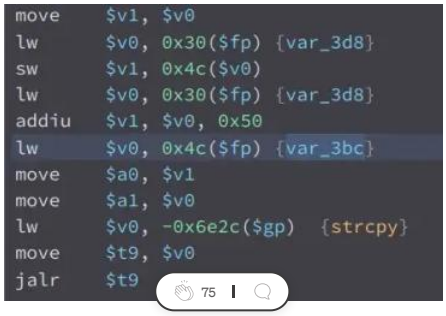
While looking through the functions of tenda, i found this one interesting function `saveParentControlInfo`



What made this function interesting is this.



We can see here that it get the value of the post parameter `urls` using `websGetVar` then save its value to the variable `var_3bc` . If we follow this variable, we will see this.



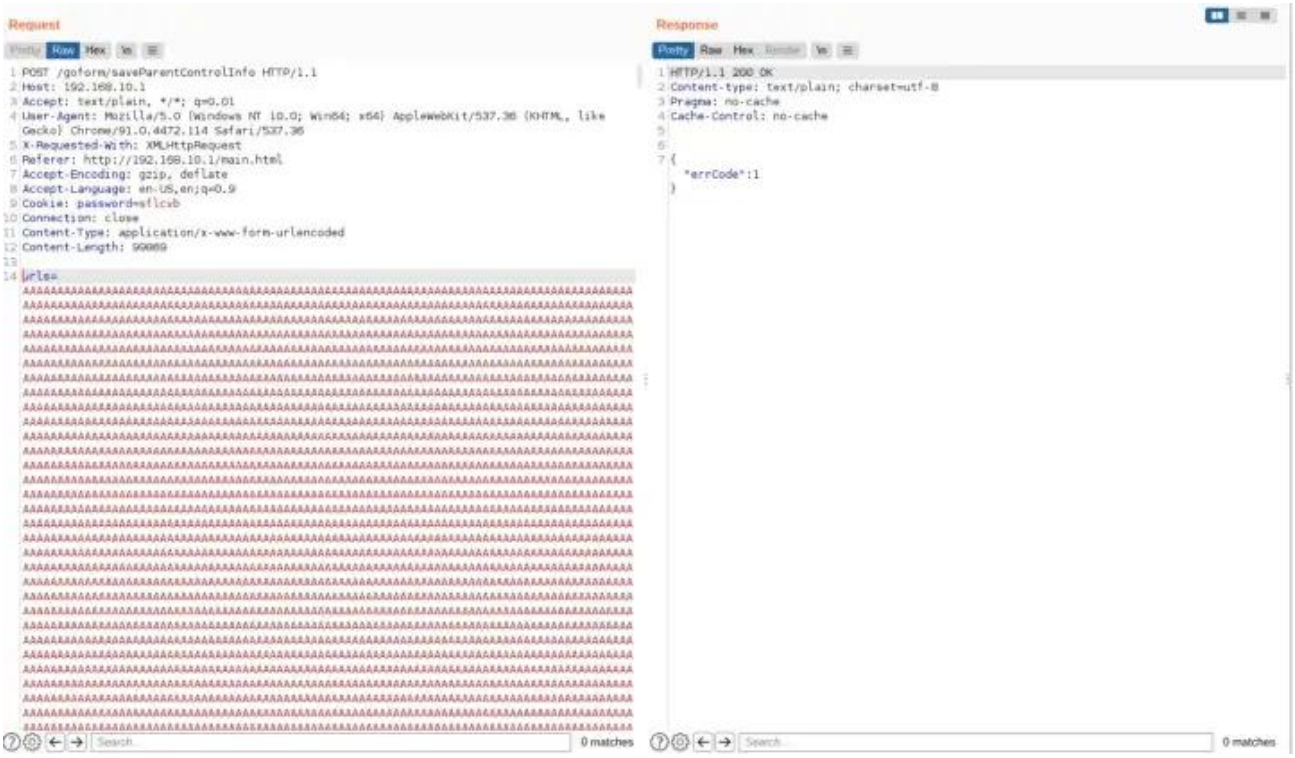
We can see that it is used as an argument to `strcpy`. Now as we all know, `strcpy` is vulnerable to buffer overflow. So lets try it our, if we send a a long string in the `urls` parameter, the server should crash. But first, we have to find out the vulnerable endpoint.

Looking at the cross references to `saveParentControlInfo`, i saw a cross references to `formDefineTendDa`

```
lw      $gp, 0x10($fp) {var_10}
lw      $v0, -0x7ae4($gp)  {data_53e34c}
addiu   $a0, $v0, -0x6568  {data_509a98, "saveParentControlInfo"}
lw      $a1, -0x791c($gp)  {saveParentControlInfo}  {data_53e514}
lw      $v0, -0x7c88($gp)  {websFormDefine}  {data_53e1a8}
move    $t9, $v0  {websFormDefine}
jalr    $t9  {websFormDefine}
```

That means, our vulnerable endpoint is `saveParentControlInfo`, so now, we can test it out.

I tried it out in burpsuite and saw this,



`"errCode":1`, that is not what were expecting, lets find out why that happened. After reversing the function once again, i found the `errCode: 1`

```
lw      $v0, -0x7500($gp)  {data_53e930}
addiu   $v0, $v0, -0x6e60  {0x5191a0, "[%d][%s] time string is null!!!!…"  }
move    $a0, $v0  {0x5191a0, "[%d][%s] time string is null!!!!…"  }
addiu   $a1, $zero, 0x21d
lw      $v0, -0x7500($gp)  {data_53e930}
addiu   $a2, $v0, -0x6ca0  {__FUNCTION__.9859, "saveParentControlInfo"}
lw      $v0, -0x6f08($gp)  {printf}
move    $t9, $v0
jalr    $t9
nop
lw      $gp, 0x28($fp) {var_3e0}
lw      $a0, 0x408($fp) {arg_0}
lw      $v0, -0x7500($gp)  {data_53e930}
addiu   $a1, $v0, -0x6e3c  {data_5191c4, "HTTP/1.1 200 OK\nContent-type: t…"  }
lw      $v0, -0x7f8c($gp)  {websWrite}  {data_53dea4}
move    $t9, $v0  {websWrite}
jalr    $t9  {websWrite}
nop
lw      $gp, 0x28($fp) {var_3e0}
lw      $a0, 0x408($fp) {arg_0}
lw      $v0, -0x7500($gp)  {data_53e930}
addiu   $a1, $v0, -0x6dd8  {data_519228, "{"errCode":%d}"}
addiu   $a2, $zero, 1
lw      $v0, -0x7f8c($gp)  {websWrite}  {data_53dea4}
move    $t9, $v0  {websWrite}
jalr    $t9  {websWrite}
nop
lw      $gp, 0x28($fp) {var_3e0}
lw      $a0, 0x408($fp) {arg_0}
addiu   $a1, $zero, 0xc8
lw      $v0, -0x7f38($gp)  {websDone}  {data_53def8}
move    $t9, $v0  {websDone}
jalr    $t9  {websDone}
nop
lw      $gp, 0x28($fp) {var_3e0}  {_gp}
b       0x4a1480
nop
```

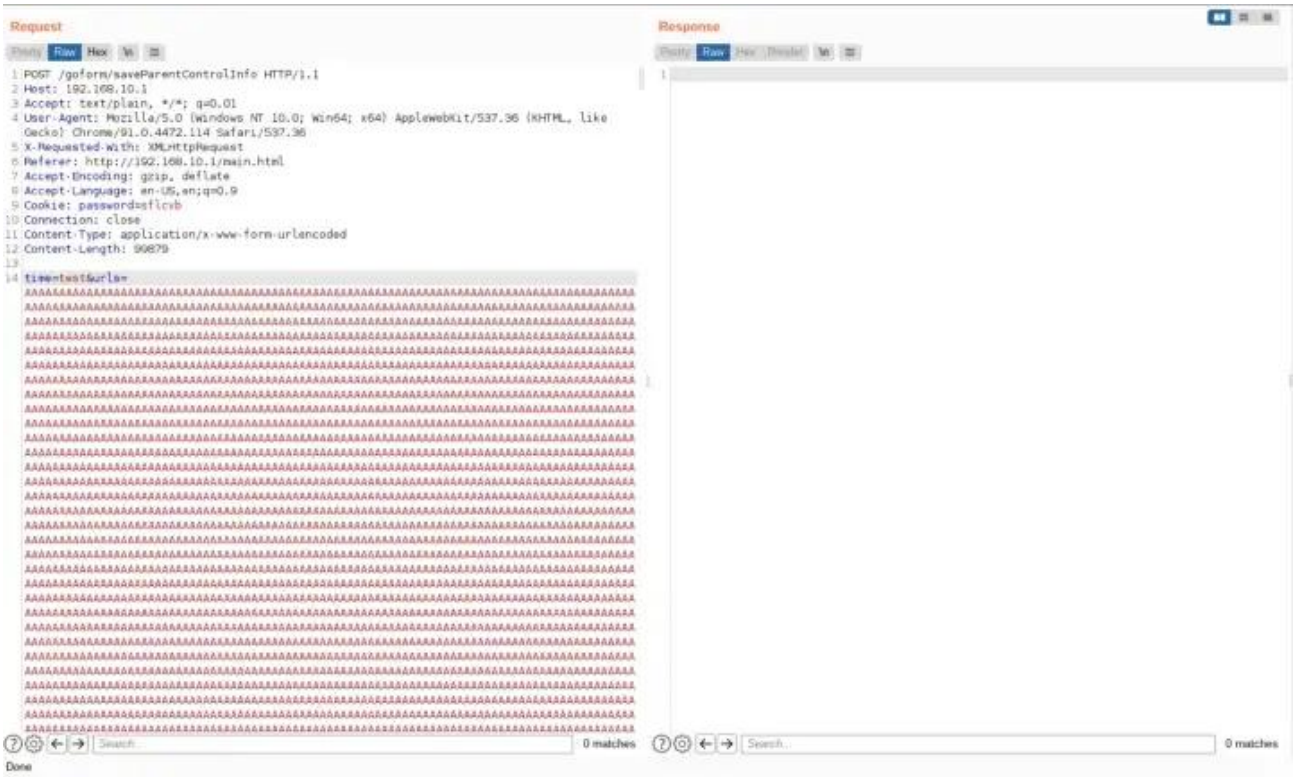
We can see the error code string there. Now lets see what causes it to jump there.

```
lw      $v0, 0x54($fp) {var_3b4}
lbu     $v0, ($v0)
bne     $v0, $zero, 0x4a0b94
nop
```

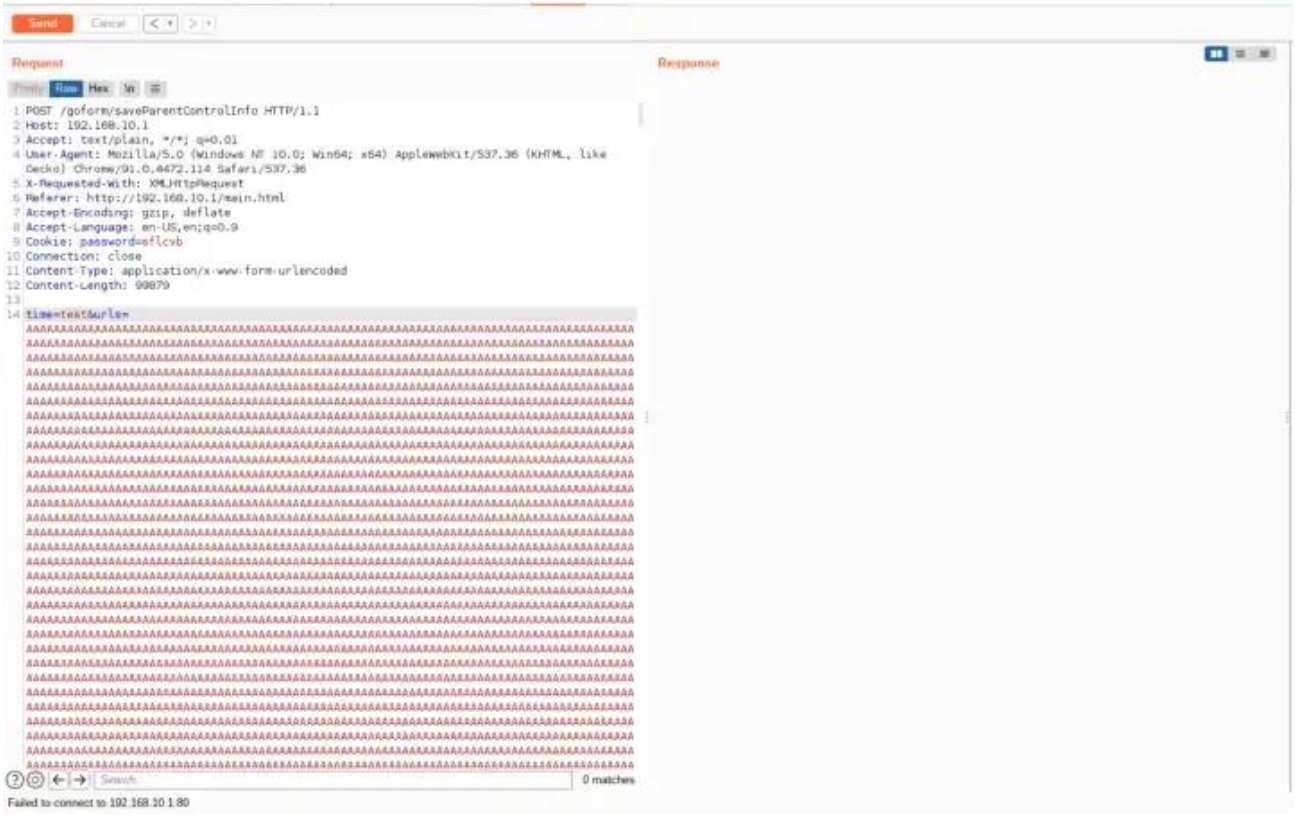
Here, we can see that it checks if the `var_3b4` is equals to zero, if it is, it will jump to the errCode. if we follow this `var_3b4`,

```
lw      $a0, 0x408($fp) {arg_0}
lw      $v0, -0x7500($gp)  {data_53e930}
addiu   $a1, $v0, -0x70b4  {data_518f4c, "time"}
lw      $v0, -0x7500($gp)  {data_53e930}
addiu   $a2, $v0, -0x70f8  {0x518f08}
lw      $v0, -0x7c84($gp)  {websGetVar}  {data_53e1ac}
move    $t9, $v0  {websGetVar}
jalr    $t9  {websGetVar}
nop
lw      $gp, 0x28($fp) {var_3e0}
sw      $v0, 0x54($fp) {var_3b4}
```

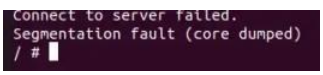
We can see that it is the output of `websGetVar` with time parameter. In our last attempt, we didnt send a time parameter so it is equal to null which caused the jump to the errCode. Lets try it again but this time, lets provide a time parameter



No response, lets try it again



Failed to connect. That means we successfully crashed the web server. We can confirm it even more by looking at the emulation



So we have a buffer overflow confirmed.

Sadly, we cant overwrite the program pointer since this is a heap overflow. If we go back to the vulnerable strcpy

```
lw      $v0, 0x30($fp) {var_3d8}
addiu   $v1, $v0, 0x50
lw      $v0, 0x4c($fp) {var_3bc}
move    $a0, $v1
move    $a1, $v0
lw      $v0, -0x6e2c($gp)  {strcpy}
move    $t9, $v0
jalr    $t9
```

We can see that in the first argument ***$a0***, it uses the variable `var_3d8 + 0x50`. If we trace back this `var_3d8`, we can see that it is the output of `malloc`

```
addiu   $a0, $zero, 0x254
lw      $v0, -0x6de4($gp)  {malloc}
move    $t9, $v0
jalr    $t9
nop
lw      $gp, 0x28($fp) {var_3e0}
sw      $v0, 0x30($fp) {var_3d8}
```

meaning, it is pointing to the heap, not the stack, thats why we cant overwrite the program pointer with what we want. However, with heap overflow, we can overwrite the other data in the heap. But i will end the writeup now.

Other parameter is also vulnerable like deviceId and time but i didnt talked about them since it is already reported and is already a cve CVE-2020–13393. This one is not a cve yet tho so this is the one that i focused in this writeup.

I tried contacting tenda but they didnt responded so i decided to publish this writeup now.

Thanks for reading.

Join the discord server: https://discord.gg/bugbounty

Hacking Cve Infosec Hack Io T