* Pre-authenticated Device Reboot

There is pre-auth vulnerability in the device which let the attacker to reboot the device remotely. The “etc\_ro\web\cgi-bin” directory is somehow misconfigured and will try to execute every file we GET from it. There is a reboot.sh file in the cgi-bin directory which you can see its content below:

#!/bin/sh

echo "Content-type: text/html"

echo ""

echo "<body>rebooting</body>"

reboot &

So we can reboot the device remotely simply by sending a GET request to the following URL, http://device/ etc\_ro/web/cgi-bin/reboot.sh .

* Pre-authenticated Configuration File Reading

Same as the previews vulnerability, there is also an ExportSettings.sh file in the CGI directory which let you to read the device configuration file. There is almost all the Config data’s inside this file, from device password to encryption keys and any other critical data.



You can see content of the ExportSettings.sh below:

#!/bin/sh

#output HTTP header

echo "Pragma: no-cache\n"

echo "Cache-control: no-cache\n"

echo "Content-type: application/octet-stream"

echo "Content-Transfer-Encoding: binary" # "\n" make Un\*x happy

echo "Content-Disposition: attachment; filename=\"RT2880\_Settings.dat\""

echo ""

echo "#The following line must not be removed."

echo "Default"

ralink\_init show 2860 2>/dev/null

* Undocumented Username/Password

While we started reversing the device binaries we want to see is it possible to bypass the login process without knowing the password. So we came up with this in the process of reversing login section.



If you look closely you can see there is two configuration parameters passed to the nvram\_bufget() function. The first one is the normal “Login/Password” fields. But the second seems a little weird. After digging more in the login process, we got a new pair of username and password, “system/system”. This login credential is undocumented from User (in our knowledge) and you can even see or change its default password from a regular user administration panel.

* Pre-authenticated Command Execution

While we continue our journey in reversing the binaries, we came up with this piece of code:



As you can see, it is responsible for handling Ping requests and the “ping\_ipaddr” will contain the IP address you choose for the device to ping (mostly used for network diagnostics).more on the value gathered from “ping\_ipaddr” is sent to there :



It is pretty obvious that the result of “ping\_ipaddr” websGetVar() is directly sent to System() function as a part of ping command.so by specifying your own command as a IP address, you can execute any command on the device host. The vulnerability is remotely trigger-able and you don’t need any authentication to get to the ping path.



* Pre-authenticated Stack Buffer Overflow Vulnerability

If we look closely at the Ping process, we can see there is also a buffer overflow vulnerability.



The “ping\_ipaddr” is directly sent to the sprint() function with any bound checking mechanism which leads to stack based buffer overflow. We can use this vulnerability as DOS or code execution ( we jus have to deal with the MIPS caches ;)

* Plan Text sensitive data storage
* Vulnerable Dnsmasq
* DNS amplification
* SBC-RSP vulnerability
* dimclient and other remote points