Extraordinary Vulnerabilities Discovered in TCL Android TVs, Now World's 3rd Largest TV Manufacturer.

Extraordinary Vulnerabilities Discovered in TCL Android TVs, Now World's 3rd Largest TV Manufacturer.

by Sick Codes — November 9, 2020 - Updated on April 3, 2021 in Security 10

TCL Android TV Vulnerability

The following piece is the culmination of a three-month long investigation into Smart TVs running Android. Having lived through this research experience, I can wholeheartedly say that there were multiple moments that I, and another security researcher that I met along the way, couldn't believe what was happening. On multiple occasions I found myself feeling as though, "you couldn't even make this up…"

I'm a security researcher, a freelance developer, and a hacker.

Please follow me on Twitter @sickcodes here: https://twitter.com/sickcodes

The second researcher in this story is John Jackson: https://twitter.com/johnjhacking, an Application Security Engineer with Shutterstock, and a hacker.

We met about half way through this, and I have included his experience too.

UPDATE: April 2, 2021. TCL now has a VDP! https://src.tcl.com/en/index

Since our research was published in November 2020, many things happened both positive and negative. One very impressive positive outcome is TCL has actually taken on-board many of the suggestions we made regarding their security. In my professional opinion, I think it is one of the most comprehensive Bug Bounty programs I've seen. They made some very impressive changes and John & myself (sickcodes) are pretty impressed: https://src.tcl.com/en/index

Initial Research

Near the end of September, while conducting research into low-end Android boxes, I came across a number of serious flaws in the way in which these devices were being designed.

Without delving into the nuances of each device, all of the Smart TV products are Android based.

There are four types of TV products in the TV market:

1. TV Sticks
2. TV Boxes
3. Smart TVs
4. Android TVs

All of them are ARM based single board computers (SBCs). Most of the dies are 32bit, some are 64bit, but all of them are like a little Raspberry Pi competitor, focusing on GPU performance through the small, but powerful, Mali GPUs.

Some of the products that I investigated were “factory-flawed” and deliberately insecure.

First Blood

On 2020-09-20, I discovered some ridiculous security shortfalls in the TV Sticks.

NOTE: TCL does not make TV sticks that are vulnerable. Only TCL Android TVs are affected. The following vulnerabilities refer to other products that I was testing at the time before finding the TCL vulnerability that is discussed in depth after the nmap scaps below.

Each stick that I tested had at least one of the following major security flaws.

- Port 22 open and allowing SSH access as root:root out of the box
- Port 5555 open and allowing unauthenticated android (adb) as root:root out of the box
- Rooted device, with world-executable su binaries in multiple locations
- Open WiFi network with adb and ssh daemons running

In effect, if you had a thousand of these devices, you could worm through all of them, taking advantage of the dual WiFi, plain-text WAN router credentials, and the ability to then hop from the TV stick, to the router, MITM the router, search for more vulnerable devices from the larger, more powerful router, and truly “surf the internet”.

Proof of Concept

```
# connect to the device's open WiFi network without any password
adb connect 192.168.1.1
adb shell
su
whoami
# root
```

Having witnessed how dismal the security was on these devices, or lack thereof, my plan was to write a really big proof of concept, in the form of an actual shell based worm, that would hop between the 4 or 5 TV sticks that I had.

Speaking to an associate about my idea, we ended up chatting about real Android TVs.

Suddenly, I thought, “If these sticks are the same, just little Rockchip & Amlogic CPUs, then what is so special about Smart TVs?”

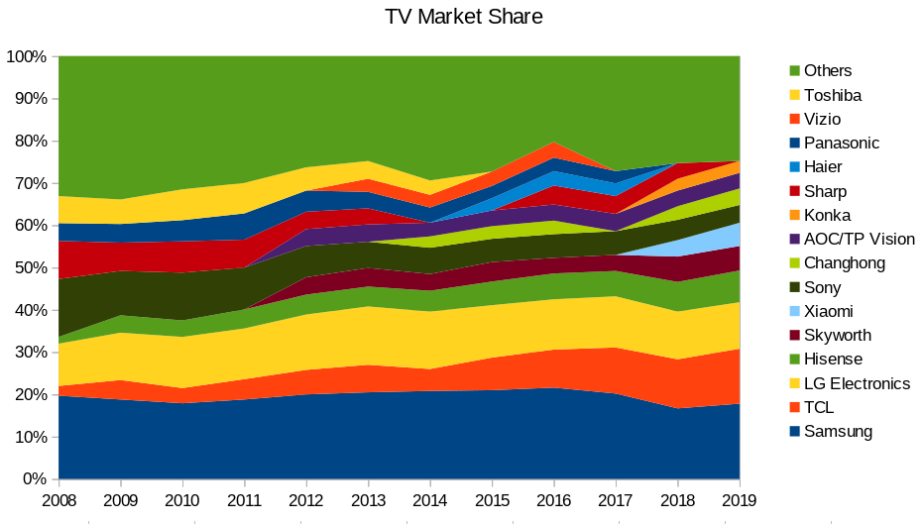
Since I don’t actually have an Android Television to test, I asked my friend what type of Smart TV does he have and is it running Android?

His answer was, “TCL and not sure.”

I hadn’t really heard much about TCL, but it turns out TCL is a huge Chinese electronics manufacturing company.

TCL has been growing their global market share, at a remarkable rate.

According to a Forbes article, they only launched in the United States in 2013 and sales began on Amazon:
https://www.forbes.com/sites/sethporges/2016/11/14/how-a-no-name-chinese-tv-brand-came-to-dominate-the-amazon-charts/?sh=15fd0d52f096



TV market share 2008 to 2020

With their Amazon success, TCL began targeting other large markets.

The key point here is that they aren't Samsung or LG, but they ARE selling millions of TV sets…

We did a remote desktop session and I ran a trivial nmap scan on the TV to see what it was running out of the box.

Here is the nmap scan:

```
Starting Nmap 7.91 ( https://nmap.org ) at 2020-10-16 21:55 UTC
…
Scanning 10.0.0.117 [65535 ports]
Discovered open port 6550/tcp on 10.0.0.117
Discovered open port 8012/tcp on 10.0.0.117
Discovered open port 6466/tcp on 10.0.0.117
Discovered open port 8009/tcp on 10.0.0.117
Discovered open port 9000/tcp on 10.0.0.117
Discovered open port 8443/tcp on 10.0.0.117
Discovered open port 10101/tcp on 10.0.0.117
Discovered open port 46211/tcp on 10.0.0.117
Discovered open port 7989/tcp on 10.0.0.117
Discovered open port 6467/tcp on 10.0.0.117
Discovered open port 6559/tcp on 10.0.0.117
Discovered open port 6553/tcp on 10.0.0.117
Discovered open port 4332/tcp on 10.0.0.117
Discovered open port 8008/tcp on 10.0.0.117
Completed SYN Stealth Scan at 21:56, 20.40s elapsed (65535 total ports)
Initiating Service scan at 21:56
Scanning 14 services on 10.0.0.117
…
Completed Service scan at 21:58, 156.41s elapsed (14 services on 1 host)
Not shown: 65521 closed ports
```

If you nmap your Android mobile phone, you will generally find 0 open TCP ports.

Zero.

So why does a TV need so many open ports?

While there are some reasons why TVs should have open ports, some of the above services warranted much deeper investigation.

Since I was in a remote desktop session, I just entered all the URLs manually into his web browser.

```
http://10.0.0.117:6550
http://10.0.0.117:8012
http://10.0.0.117:6466
http://10.0.0.117:8009
http://10.0.0.117:9000
http://10.0.0.117:8443
http://10.0.0.117:10101
http://10.0.0.117:46211
http://10.0.0.117:7989
http://10.0.0.117:6467
http://10.0.0.117:6559
http://10.0.0.117:6553
```

http://10.0.0.117:4332
http://10.0.0.117:8008

I also tested the https:// editions:

https://10.0.0.117:6550
https://10.0.0.117:8012
https://10.0.0.117:6466
https://10.0.0.117:8009
https://10.0.0.117:9000
https://10.0.0.117:8443
https://10.0.0.117:10101
https://10.0.0.117:46211
https://10.0.0.117:7989
https://10.0.0.117:6467
https://10.0.0.117:6559
https://10.0.0.117:6553
https://10.0.0.117:4332
https://10.0.0.117:8008

Some of the pages were blank white pages. This can indicate an API endpoint.

Some of the pages just hang the browser.

Then the rest of the nmap scan came through…

```
PORT STATE SERVICE VERSION
4332/tcp open getty-focus?
6466/tcp open ssl/unknown
| ssl-cert: Subject: commonName=atvremote/BeyondTV2/BeyondTV/BeyondTV2/unknown
| Subject Alternative Name: email:android-tv-remote-support@google.com
| Issuer: commonName=atvremote/BeyondTV2/BeyondTV/BeyondTV2/unknown
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
…
6467/tcp open tcpwrapped
6550/tcp open fg-sysupdate?
| fingerprint-strings:
| NULL:
|_ Version 4
6553/tcp open unknown
6559/tcp open unknown
| fingerprint-strings:
| GenericLines:
…
7989/tcp open unknown
| fingerprint-strings:
| FourOhFourRequest:
| HTTP/1.1 404 Not Found
| Content-Type: text/plain
| Date: Fri, 16 Oct 2020 10:57:17 GMT
| Accept-Ranges: bytes
| Connection: keep-alive
| Content-Length: 26
| Error 404, file not found.
| GenericLines:
| HTTP/1.1 400 Bad Request
| Content-Type: text/plain
| Date: Fri, 16 Oct 2020 10:56:27 GMT
| Connection: keep-alive
| Content-Length: 56
| REQUEST: Syntax error. Usage: GET /example/file.html
| SIPOptions:
| HTTP/1.1 404 Not Found
| Content-Type: text/plain
| Date: Fri, 16 Oct 2020 10:57:37 GMT
| Accept-Ranges: bytes
| Connection: keep-alive
| Content-Length: 26
|_ Error 404, file not found.
8008/tcp open http?
|_http-title: Site doesn\'t have a title (text/html).
8009/tcp open ssl/castv2 Ninja Sphere Chromecast driver
|_ajp-methods: Failed to get a valid response for the OPTION request
8012/tcp open unknown
```

```
8443/tcp open ssl/https-alt\?
|_http-title: Site doesn\'t have a title (text/html).
| ssl-cert: Subject: commonName=/organizationName=Google Inc/stateOrProvinceName=Washington/countryName=US
| Issuer: commonName=TCL TV BeyondTV Realtek RTD2851 Cast ICA/organizationName=Google Inc/stateOrProvinceName=Washington/countryName=US
9000/tcp open ssl/cslistener?
10101/tcp open ssl/ezmeeting-2?
46211/tcp open tcpwrapped
```

Port 7989 was showing a 404 error, yet when I visit 10.0.0.117:7989 in the browser, an error is shown.

http://10.0.0.117:7989 did not return a page in the browser.

TCL directory file structure

What kind of special web server doesn’t show an index page, but shows deeper pages?

I had recently done research on an IoT device that was serving CGI scripts from the / directory, so the first page I thought to test was init.rc

http://10.0.0.117:7989/init.rc

403 Forbidden.

Yikes.

This means that the file is exists but we are not authorized to view it.

Naturally, I tested some other Android directories:

http://10.0.0.117:7989/sdcard

TCL directory file structure vulnerability

If you work with computers, no matter whether it be mobile apps, web apps, websites, back-end, front-end, upend…

Ask yourself this question right now:

> *When in the history of your career…*
> *Have you ever needed to serve the entire filesystem…*
> *over http?*

This becomes a really import question because this custom vendor firmware is currently installed in millions of TCL Android TVs around the world.

My friend who was actually on the phone to me while we were doing the remote desktop, was fairly surprised.

“Why can we see all the files in the TV?”

Port 7989 is not on the list of standard TCP/UDP ports by the Internet Assigned Numbers Authority (IANA), https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

This means, without scanning all 65,535 ports, most scanners will skip that port.

Secondly, the actual root page is blank.

So in order to scan more than 1 page per port, port scan times will exponentially increase…

Curiously, I checked the IANA list for other ports that end with 989:

ftps-data	989	tcp	ftp protocol, data, over TLS/SSL
tr-rsrb-p3	1989	tcp	cisco
mshnet	1989	tcp	MHSnet
zarkov	2989	tcp	ZARKOV
bv-queryengine	3989	tcp	BindView-Query
parallel	4989	tcp	Parallel
wbem-https	5989	tcp	WBEM
sunwebadmins	8989	tcp	Sun
	9989-9989		Unassigned
	10934-10989		Unassigned

Notably, 6989 and 7989 are missing.

In fact, the range 7983-7997 is unassigned by IANA.

Why would an Android device need a web server running on a non-standard port?

What kind of manufacturer publishes the whole file system of a device?

Contact Tracing - CVE-2020-27403

At this point, I drafted the first CVE Request and sent it off to MITRE.

A copy of the draft advisory was sent to TCL's security@tcl.com email address, requesting that they confirm receipt of the report.

The email bounced, so I submitted a contact form via their website requesting immediate support and escalation to the security department.

As this was a Friday, I was not expecting a reply, but a few days passed by and MITRE did not reply either.

I double checked that the device was not covered by another Certificate Numbering Authority (CNA) and ended up working out, to my surprise, that TCL Corporation owns Alcatel.

In fact, they also own(ed) Blackberry Mobile.

Both of those brands are Open Handset Alliance members, and they are covered by the Google Android CNA.

So I resubmitted the report to the Android CNA.

A few more days passed and no reply from anyone yet.

Then I contacted a friend who put me in touch with a researcher, who had a contact of someone at the company. I emailed them.

Yet again, no reply.

I then publicly requested contact details of anyone at the TCL security team in this thread: https://community.disclose.io/t/looking-for-security-contact-at-tcl-corporation/39

Another researcher named John Jackson replied, and we connected on Twitter.

John had recently gone through a messy public disclosure with a company that threatened to sue him even though they had already stated, in writing, "the bug does not exist."

This is where things get tough.

Requests for Assistance

John: I was browsing the disclose.io boards and saw Sick Codes' post requesting assistance with disclosure for a major vulnerability that he found on TCL's Smart TVs.

Seeing that he had attempted to contact the company through their phone support, contact forms, twitter, and email addresses - it was obvious that he wasn't getting any response.

First and foremost, I replied on the board and connected with him on Twitter to validate the authenticity of the vulnerability.

It seemed like a legitimate issue, the Smart TV was exposing a random port and I could see a bunch of files on the client side. Working with Sick Codes, I dug a little deeper into the filesystem and noted that a lot of sensitive files could be accessed.

Nonetheless, not all of the sensitive files looked accessible and there were certain permission restrictions such as accessing the sensitive configuration files.

Turns out, a built-in directory existed on the TV and maintained root level permissions. The directory was nearly an identical clone of the exposed filesystem, and allowed us to circumvent any issues accessing various critical files

It was clear that utilizing this vulnerability could result in remote code execution or even quick network pivots with the intention of exploiting systems quickly with ransomware.

With a vast list of affected TV models enumerated from the exposed filesystem on the client-side, [or in other words millions of vulnerable TVs] I set out to help Sick Codes in the disclosure process.

Unfortunately, the result of my attempted disclosure was nearly identical to Sick Codes', resulting in no response on Social Media or Email.

I called TCL and talked to a support representative. I urged her that we had a serious vulnerability on our hands and she stated that she had no contact info to the Security team, and didn't even think/know if TCL had a Security team. She asked me if I wanted to disclose the vulnerability over the phone so that she could submit it into the ticketing system: I said no. It's generally not the best practice to expose a highly sensitive vulnerability via a shared ticketed system.

Obviously, we had to approach other methodology so I contacted CERT to ask for assistance in disclosure. CERT did not respond to us initially.

Days later Sick Codes managed to acquire a name and point of contact at TCL. We reached out and after about a week, the company had acknowledged the vulnerability and stated that they would patch the issue. We asked for updates over the course of weeks, but they stopped responding to us. CERT finally responded and told us to disclose the vulnerability if that was the case.

Respectfully, John Jackson

Part 2: Backdoor Update Vulnerability - CVE-2020-28055

Sick Codes here again. It took 13 days for TCL Corporation to reply to our initial report.

Almost 2 weeks to confirm receipt of a security report, from the World's 3rd largest TV manufacturer.

Second to that, but the TCL Security team actually confirmed in the same email, that they had fixed the vulnerability.

Serious questions must be raised at this point. Why did it take two weeks to reply, and how was it so easy to fix?

What Android service was stopped in the fix?

Will the TV function as expected without this overly permissive directory allowing anyone on the same network to download files on the TV?

It was at this time, I decided to try find some more vulnerabilities while we had the attention of the security team.

I went digging through .rc files looking for vendor specific changes.

Indeed, there were critical changes made by TCL to several folders on the TV file system that should be absolutely locked down.

In the file:

/system/vendor/etc/init/hw/init.rtd285o.rc

I found the following critical errors:

```
#tcl save data directory
mkdir /data/vendor/tcl 0777
#tcl oad data directory
mkdir /data/vendor/upgrade 0777
mkdir /var/TerminalManager 0777
```

This means that these folders are now writable, by all users on the file system.

/data/vendor/tcl contains critical files to operate the TV.

These files should not be writable by arbitrary users, or potentially malicious apps or APKs.

One can only guess what consequences having read & write access to the TCL updates folder might have…

Secondly, what I found in "TerminalManager_Remote" was a little head-turning:

TerminalManager is an App, named TerminalManager_Remote.apk

Unpacking the APK contains the following configuration file.

```
[cwmp]
enable=1;
soap_env=SOAP-ENV
soap_enc=SOAP-ENC
acs_auth=1
cpe_auth=0
event_filename=/etc/cwmpevent.bin
###CN official url
acs_url=http://admin.acs.huan.tv/acs
logserver_url=http://service.acs.huan.tv/acs/uploadLog
fileserver_url=http://admin.acs.huan.tv/file/uploadFile
screenserver_url=http://admin.acs.huan.tv/file/uploadScreen
longconnect_url=http://socket.acs.huan.tv
logserver_url_stat=http://service.acs.huan.tv/acs/uploadLogs
###HK area official url
HK_acs_url=http://as.admin.acs.huan.tv/acs
HK_logserver_url=http://as.service.acs.huan.tv/acs/uploadLog
HK_fileserver_url=http://as.admin.acs.huan.tv/file/uploadFile
HK_screenserver_url=http://as.admin.acs.huan.tv/file/uploadScreen
HK_longconnect_url=http://as.socket.acs.huan.tv
HK_logserver_url_stat=http://as.service.acs.huan.tv/acs/uploadLogs
###TW area official url
TW_acs_url=http://as.admin.acs.huan.tv/acs
TW_logserver_url=http://as.service.acs.huan.tv/acs/uploadLog
TW_fileserver_url=http://as.admin.acs.huan.tv/file/uploadFile
TW_screenserver_url=http://as.admin.acs.huan.tv/file/uploadScreen
TW_longconnect_url=http://as.socket.acs.huan.tv
TW_logserver_url_stat=http://as.service.acs.huan.tv/acs/uploadLogs
###AP area official url
AP_acs_url=http://as.admin.acs.huan.tv/acs
AP_logserver_url=http://as.service.acs.huan.tv/acs/uploadLog
AP_fileserver_url=http://as.admin.acs.huan.tv/file/uploadFile
AP_screenserver_url=http://as.admin.acs.huan.tv/file/uploadScreen
AP_longconnect_url=http://as.socket.acs.huan.tv
AP_logserver_url_stat=http://as.service.acs.huan.tv/acs/uploadLogs
###AU area official url
AU_acs_url=http://as.admin.acs.huan.tv/acs
```

```
AU_logserver_url=http://as.service.acs.huan.tv/acs/uploadLog
AU_fileserver_url=http://as.admin.acs.huan.tv/file/uploadFile
AU_screenserver_url=http://as.admin.acs.huan.tv/file/uploadScreen
AU_longconnect_url=http://as.socket.acs.huan.tv
AU_logserver_url_stat=http://as.service.acs.huan.tv/acs/uploadLogs
###ME area official url
ME_acs_url=http://eu.admin.acs.huan.tv/acs
ME_logserver_url=http://eu.service.acs.huan.tv/acs/uploadLog
ME_fileserver_url=http://eu.admin.acs.huan.tv/file/uploadFile
ME_screenserver_url=http://eu.admin.acs.huan.tv/file/uploadScreen
ME_longconnect_url=http://eu.socket.acs.huan.tv
ME_logserver_url_stat=http://eu.service.acs.huan.tv/acs/uploadLogs
###EU area official url
EU_acs_url=http://eu.admin.acs.huan.tv/acs
EU_logserver_url=http://eu.service.acs.huan.tv/acs/uploadLog
EU_fileserver_url=http://eu.admin.acs.huan.tv/file/uploadFile
EU_screenserver_url=http://eu.admin.acs.huan.tv/file/uploadScreen
EU_longconnect_url=http://eu.socket.acs.huan.tv
EU_logserver_url_stat=http://eu.service.acs.huan.tv/acs/uploadLogs
###AF area official url
AF_acs_url=http://eu.admin.acs.huan.tv/acs
AF_logserver_url=http://eu.service.acs.huan.tv/acs/uploadLog
AF_fileserver_url=http://eu.admin.acs.huan.tv/file/uploadFile
AF_screenserver_url=http://eu.admin.acs.huan.tv/file/uploadScreen
AF_longconnect_url=http://eu.socket.acs.huan.tv
AF_logserver_url_stat=http://eu.service.acs.huan.tv/acs/uploadLogs
###NA area official url
NA_acs_url=http://na.admin.acs.huan.tv/acs
NA_logserver_url=http://na.service.acs.huan.tv/acs/uploadLog
NA_fileserver_url=http://na.admin.acs.huan.tv/file/uploadFile
NA_screenserver_url=http://na.admin.acs.huan.tv/file/uploadScreen
NA_longconnect_url=http://na.socket.acs.huan.tv
NA_logserver_url_stat=http://na.service.acs.huan.tv/acs/uploadLogs
###LA area official url
LA_acs_url=http://na.admin.acs.huan.tv/acs
LA_logserver_url=http://na.service.acs.huan.tv/acs/uploadLog
LA_fileserver_url=http://na.admin.acs.huan.tv/file/uploadFile
LA_screenserver_url=http://na.admin.acs.huan.tv/file/uploadScreen
LA_longconnect_url=http://na.socket.acs.huan.tv
LA_logserver_url_stat=http://na.service.acs.huan.tv/acs/uploadLogs
###test url
test_acs_url=http://103.235.237.119:8080/acs_manager/acs
test_longconnect_url=http://103.235.237.119:6488
test_logserver_url=http://103.235.237.119:8080/acs_web/acs/uploadLog
test_fileserver_url=http://103.235.237.119:8080/acs_manager/file/uploadFile
test_screenserver_url=http://103.235.237.119:8080/acs_manager/file/uploadScreen
test_logserver_url_stat=http://103.235.237.119:8080/acs_web/acs/uploadLogs
ca_file=/etc/ca.pem
ca_password=123456
cpe_manufacture=ChinaNMS
cpe_oui=A00001
cpe_sn=30000000000000000
cpe_name=000000
cpe_pc=OT2800
cpe_specver=V1.0
cpe_hwver=V1.0
cpe_version=V1.2.7.29
cpe_username=cwmp
cpe_password=cwmp
acs_username=app1
acs_password=123456
[cwmpd]
httpd_port=5400
http_timeout=-1
```

Inside this APK is is netcwmp. Which included the following administration controls:

https://github.com/netcwmp/netcwmp.git

SOAP/XML Parser
SSL
HTTP Server
HTTP Client
Ini config file Parser
Digest Authentication
GetRPCMethods

Inform
SetParameterValues
GetParameterValues
GetParameterNames
Download
Upload
AddObject
DeleteObject
FactoryReset
TransferComplete
Reboot
TR-069 Object Models Interface

Config File:
acs_auth: ACS auth CPE
cpe_auth: CPE auth ACS
acs_url: ACS URL
cpe_manufacture: Manufacture
cpe_oui: OUI
cpe_sn: CPE Serial Number
cpe_name: CPE Name
cpe_pc: CPE Product Class
cpe_username: InternetGatewayDevice.ManagementServer.ConnectionRequestUsername
cpe_password: InternetGatewayDevice.ManagementServer.ConnectionRequestPassword
acs_username: InternetGatewayDevice.ManagementServer.Username
acs_password: InternetGatewayDevice.ManagementServer.Password
httpd_port: Http server listen port
ca_file: ca pem file
ca_password: ca password

As much as I'd like to answer the big question here, I don't know if I can.

I don't have the answer. TCL does, however.

The TV that I conducted preliminary tests on was silently patched. No update warning was sent.

The fact that they were able to identify the vulnerability, and have patched it on their side automatically, without user input is also lightly striking.

Please make your own conclusions from our research.

If you want more research write-ups and investigations, then keep us both on Twitter.

Follow me @SickCodes on Twitter: https://twitter.com/sickcodes

Follow @johnjhacking on Twitter: https://twitter.com/johnjhacking

Shortly after the confirmation email, MITRE issued us two CVE ID's for the vulnerabilities discovered:

CVE-2020-27403

https://sick.codes/sick-2020-009

TCL Android Smart TV (All) - Exposure of Information Through Directory Listing - TCL Android TV Filesystem Browsable to Unauthenticated Attackers Over the Adjacent Network on Port 7989

CVE-2020-28055

https://sick.codes/SICK-2020-012

TCL Android Smart TV (All) - Incorrect Permission Assignment for Critical Vendor Resources - TCL Android TV Vendor Configuration & Upgrade Folders World Writable to Local Attacker

Comments 10

Pingback: Weekendowa Lektura: odcinek 391 [2020-11-15]. Bierzcie i czytajcie | Zaufana Trzecia Strona

Pingback: Security Holes Opened Back Door To TCL Android Smart TVs | The Security Ledger with Paul F. Roberts

juha 2 years ago

http://192.168.1.111:49153 (Philips 55PUS7503/12 Smart Android TV) gives only a “404 page not found” error. Tried to extend that address with Android directories and init.rc but they all gave a “404 page not found” error.

http://192.168.1.111:9080 gives “Status=OK” message but nothing else. Then there are few other ports that give an error message “didn’t send any data” when tried on a web browser.

Reply

admin 2 years ago

This means you are okay.

`Status=OK` is some api that is used, health checking.

If nothing showed /init.rc as 403 forbidden, then it’s most likely that you are not vulnerable.

Reply

Jiml8 2 years ago

I am just starting to root around in my hisense 65” qled android TV. I don’t recall the model number but it is a 2020 device. I have not found anything as egregious as what is described here, though I am still exploring the open ports I have found (which is what brought me here).

To use an android TV safely in spite of its vulnerabilities, put it on its own VLAN and also put a VPN proxy on that VLAN. Direct the TV to use the VPN proxy as its gateway. On the upstream router, block all access to the VLAN from either direction, except for access required by the VPN proxy. Then, connect the TV to the VLAN using an ethernet connection. This likely will require you to use a trunking switch, but will lock your TV down completely.

No inbound connections from anywhere will arrive at the TV; the VPN will prevent that because the external VPN server (the other end of your tunnel) will have no idea how to route any inbound traffic that is aimed at your TV, and the address of that remote server will show as the address of your TV. This prevents any remote control of the TV.

I do this, and in addition I run a DNS server on my network (pihole) which is configured to block all the tracking that the TV tries to do.

Reply

admin 2 years ago

Very nice setup! Fully isolating devices like you have done should be much more commonplace these days. Thanks for the comment!

Reply

Patrick Dolan 2 years ago

I have a
Hisense 65” H9F Android TV.

It recently updated to Android 10.

I’m wondering if this TV has security issues. I always get messages that say there’s vulnerabilities if I cast something to the TV.
Should I stop using it.
is there a safer way to use Android on the TV or my better off with the Roku TV.
Any help would be appreciated as I want to buy a new TV.

Reply

admin 2 years ago

Roku TV’s are fine. We spoke to Roku and they are not vulnerable to this issue. This specifically affects TCL Android TVs.

TCL Roku TVs are fine, even though they most likely come from the same factory.

Reply

Shepherdess 2 years ago

I think it’s obvious what they are doing: mass surveillance for their government, which of course is a majority owner of every company which operates inside of the country.

Reply

admin 2 years ago

I can’t say with certainty anything, but they do have to absolute ability to control the TV remotely.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name

Email

Website

Privacy - Terms

I am human

POST COMMENT

@sickcodes

@sickcodes

@sickcodes

Discord Server

sickcodes.slack.com

t.me/sickcodeschat

./contact_form