Background

Rolling Flaws

What is Rolling Flaws?

A program developed by Derek Jamison for Flipper Zero to play with rolling codes vulnerabilities.

Why Do I Care?

Because testing rolling codes vulnerabilities on real products that are in use can lead to Bad Things™, not only desyncing controllers but even Letting The Smoke Out™. Rolling Flaws allows the Flipper to act as a receiving unit with various kinds of issues.

Frequencies and The Law

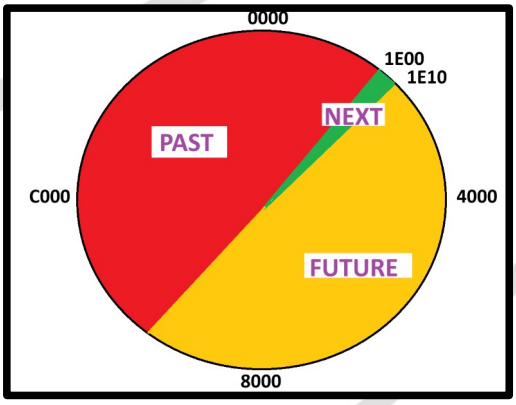
The frequency defaults to 433.92MHz. When used on the Flipper, this complies with the law. Please not not change the frequency. The modulation on the Flipper should be AM650.

Basic Configuration and Fiddling (More Later)

Use the config menu to set up the flaws you want to introduce. I'd advise keeping the protocol as KL(DH) – this is the common KeeLoq 64 protocol with the manufacturer ID as DoorHan. The Fix is a combination of the button pressed and the serial no of the transmitter – leave it at 0x284EE9D5.

The window refers to the number of valid future codes (next), the number of future codes (limited number before the counter has to reset) and the gap.

The gap is what can allow a desynced remote to be resynced as two future codes received within the gap (so 2 of 2 codes or 2 of 4 codes) will reset the counter and resync the remote. This is an attack vector.



Exit the app and restart it to reset the configuration.

Rolling Flaws: Github Jamisonderek

Intercepting a Remote

Rolling Flaws

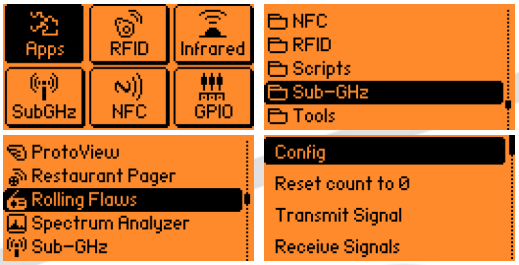
Opening The App

1) Go to Apps

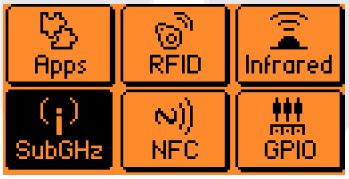
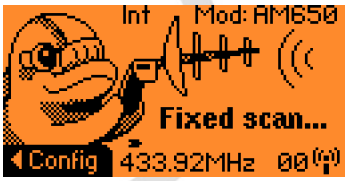
2) Scroll to Sub-GHz

3) Scroll to Rolling Flaws

4) App main menu



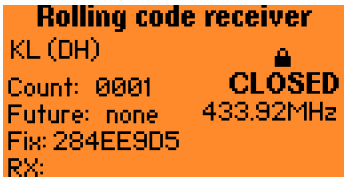
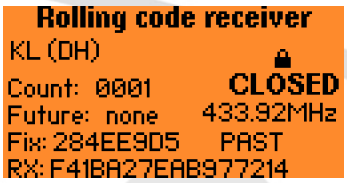
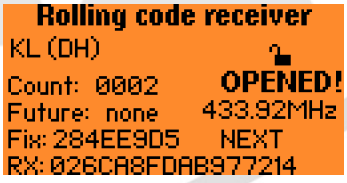
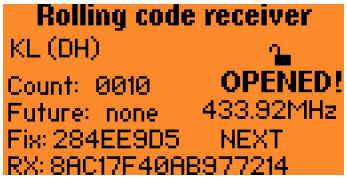
Intercepting Signals

Info	Rolling Flaws Flipper	Hacking Flipper
Once set up, get ready with transmit signal.		
Go to SubGHz		
Select "Read"		
In Config: **Frequency to 433.92** **Modulation AM650**		
Transmit signal		
This should appear… Select.		
You have received the signal… **...PTO**		

Rolling Flaws: Github Jamisonderek

Cloning a Remote

Rolling Flaws

Info	Rolling Flaws Flipper	Hacking Flipper
You should have the intercepted remote from the last page.		
Go to receive signals.		
This display shows the received signals and the door status. Note count is 0001.		
Hit send. Again, note the count.		
The door is closed as the count received (0001) is not in the future set.		
Hit send again, we have advanced the count.		
Now the door opens.		
You can now keep mashing send.		
The door will keep on opening. Yay.		

Rolling Flaws: Github Jamisonderek

App Menu Options

Rolling Flaws

Top Level Menu

Config

This is where you can configure the settings. The settings are reset whenever the application restarts.

Reset count to 0

This will reset the count to 0. This is useful when you want to start over or want to test some rollback scenarios.

Transmit Signal

This will transmit the signal. This is useful to capture the next signal.

Receive Signals

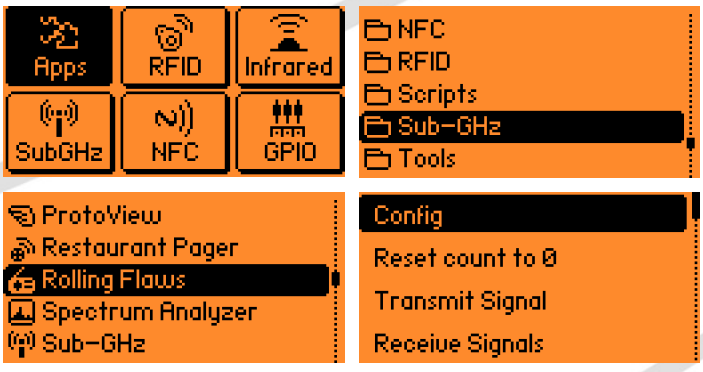
This will receive signals. This is the primary purpose of the application.

Sync Remote

This will sync the configuration using a remote signal. This is useful when you want to pair the Flipper Zero to your remote.

About

This will show information about the application.



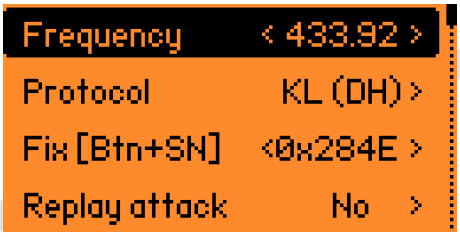
App Menu Options

Rolling Flaws

Config Menu

Frequency

Frequency is the frequency that the receiver and transmitter will use. Leave this at 433.92MHz (defaults to this).



Protocol

- "KL (DH)" is the KeeLoq protocol with the manufacturer key from DoorHan.

- "KL (All)" is the KeeLoq protocol but accepts any manufacturer key known to the receiving unit (a vulnerability with generic receivers).

- "KL(Custom)" is set when doing a 'Sync Remote' with your own remote.

Fix [SN+Btn]

This is the button and serial number to decode.

- 0x20000000 is for a test transmitter.

- 0x284EE9D5 is default and used in sample files from this project.

- Custom is set when doing a 'Sync Remote' operation.

Replay attack

If this is set to "yes" then it is possible to do a replay attack. Requires custom firmware. And yes, this vulnerability is actually present in real world receivers using rolling codes...

Count 0 opens

This will cause the receiver to open when it receives a count of 0.

PTO for more configuration options...

App Menu Options

Rolling Flaws

Config Menu

Window [next]

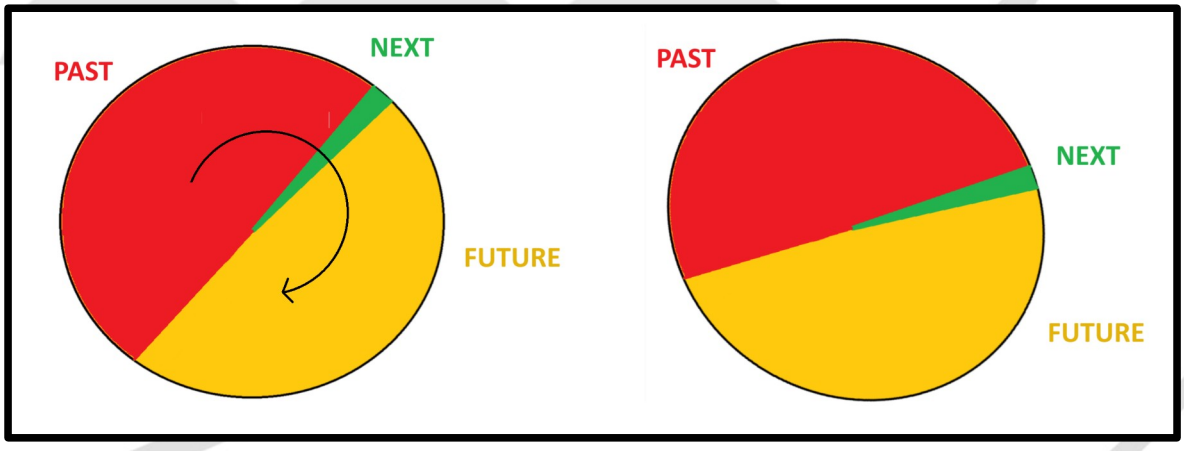
This is how many counts forward from the existing count are considered acceptable (default 16). As codes are used, the next window advances (the green segment below moves further round).

Window [future]

This is how many counts forward from the existing count are considered future. For example, if the current count is 0x0001 (1) and the window is 32768, then 0x5011 (20497) would be considered a future count, but 0xEC00 (60416) would be considered a past count (39919 ahead, so beyond the 32768 future window). This is because rolling codes just roll around after they're all used, so it prevents a code from a past rotation being considered a future code.

Window [gap]

This is how close two future counts need to be from each other for them to be considered within the gap. When a second count is received and is within the gap, the next count will be advanced to the last count sent.



It's not just the green "next" segment that moves around as codes are used; all the segments rotate around.

App Menu Options

Config Menu

The below are two more advanced options.

SN00/cfw (requires custom firmware)
If this is set to "yes" then if the decoded data serial number bytes match 0x00, then any serial number will be considered a match. If this is set to "no" then the serial number must match exactly. For this feature to validate the serial number, you will need custom firmware.

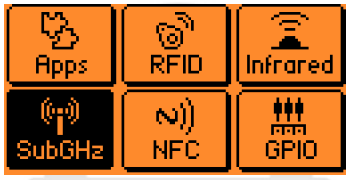
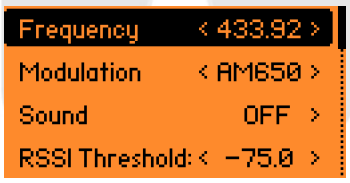
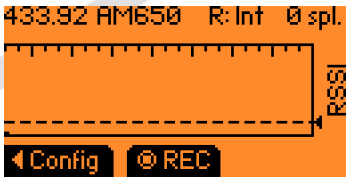
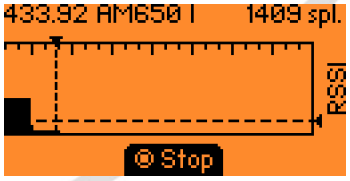
SN bits/cfw (requires custom firmware)
By default the firmware only checks 8 bits of the serial number. If this is set to "10 (dec)" and you have custom firmware, then 10 bits from the decoded data will need to match the serial number.

Raw Replay Attack 1/2

Rolling Flaws

This is a replay attack which does not need to decode any part of the signal, just record any signal on the frequency and modulation specified and replay.

When transmitting, keep Flippers >15cm apart.

Info	Rolling Flaws Flipper	Hacking Flipper
Open SubGHz		
Select “Read Raw”		
Configure as per the image.		
Open a fresh Rolling Flaws. In the config menu, enable a replay attack.		
Transmit a signal. This is so the count is not 0 (that’s a different flaw!).		
Go back to the record screen and press record.		
Transmit again. This should send count 0001.		
You should record a signal. Press stop.		

Rolling Flaws: Github Jamisonderek

Raw Replay Attack 2/2

Rolling Flaws

Info	Rolling Flaws Flipper	Hacking Flipper
Reset the count to 0. We have to do this because of app behaviour.	Config Reset count to 0 Transmit Signal Receive Signals	
Send another, so we are back to where we started.	Config Reset count to 0 Transmit Signal Receive Signals	
Back to receive signals.	Reset count to 0 Transmit Signal Receive Signals Sync Remote	
And we see the door is closed and count is 0001.	Rolling code receiver KL (DH) Count: 0001 Future: none Fix: 284EE9D5 RX: CLOSED 433.92MHz	
We can now click send.		433.92 AM650 I 1613 spl. RSSI Erase Send Save
Sending….		433.92 AM650 I 1409 spl. Hold to repeat
Having seen count 0001 again, it has opened with the reason “REPLAY”.	Rolling code receiver KL (DH) Count: 0001 Future: none Fix: 284EE9D5 RX: F41BA27EAB977214 OPENED! 433.92MHz REPLAY	

You can try this again with replay turned off in the config menu.

The captured code will be invalidated if the real remote is pressed again, so it’s of no long term use. **Rarely, attempting this will brick a real remote**, which is why we use Rolling Flaws.

This attack works on Bill and Ben (door contact alarms) but they use static codes, so once you have saved the signal, it’ll always work.

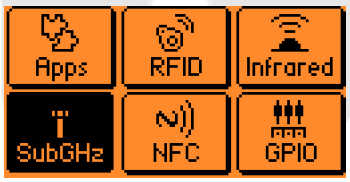
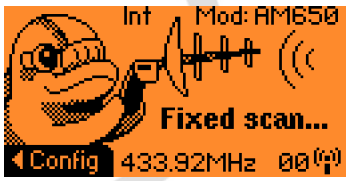
Rolling Flaws: Github Jamisonderek

Replay Attack 1/2

Rolling Flaws

We are going to decode the signal we get from the remote, as the custom Flipper firmware can do that. It's simpler for the user than the raw replay. It doesn't work with all rolling codes. **This will almost certainly brick real remotes.**

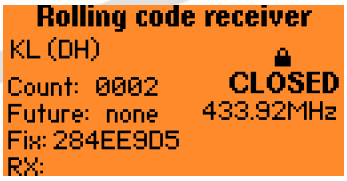
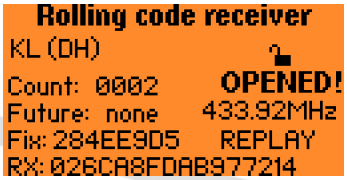
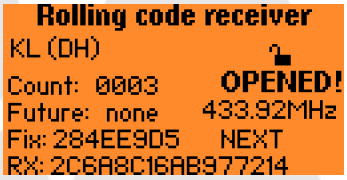
Ensure the Flippers are >15cm apart when transmitting.

Info	Rolling Flaws Flipper	Hacking Flipper
Open a fresh Rolling flaws and enable replay in the config menu.		
Transmit a signal so the count is not 0 (that's a different flaw!).		
Open SubGHz.		
Go to "Read"		
Set the frequency to 433.92MHz and modulation to AM650.		
Transmit another remote signal.		
It should have captured a signal, which you can select to open.		
The details should look like this – note the count.		

Rolling Flaws: Github Jamisonderek

Replay Attack 2/2

Rolling Flaws

Info	Rolling Flaws Flipper	Hacking Flipper
Go to “receive signals”.		
The display should look like this. Note the count compared to the hacking Flipper.		
Send with count 0002.		
Opened and the count hasn’t advanced. Reason is “replay”.		
You can keep sending with advancing counts.		
Now it’ll keep opening as it’s sending the next code in the sequence.		

In this case, the Flipper has essentially taken over from the real remote, which will likely stop working. It may be possible to resync by keeping sending transmissions from it until the count advances past the Flipper. In some cases the remotes will need deleting and reprogramming into the receiver. In extreme cases, the system can just black list that remote entirely, having detected the replay attack.

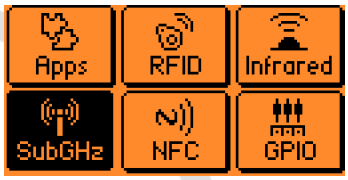
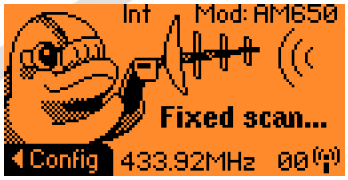
This is why it’s a bad idea to use this against kit in use.

Exploiting Gaps 1/2

Rolling Flaws

The gap value is the number of future codes that need to be received to reset the count to that future value. So we could copy a signal for later use, but the legitimate remote is going to be used in the mean time. We don't want to risk a replay detection and so we take the signal, go away and progress the count well into the future (but within the future window). It will be ahead of the real remote. We can then send 2 signals in a row and the receiver will accept our remote as the real one and sync to the new value.

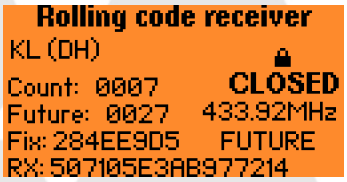
The gap feature can require 1 signal, 2 of 2, 2 or 3 or 2 of 4 signals.

Info	Rolling Flaws Flipper	Hacking Flipper
Open a new Rolling Flaws In config, set the gap. Guide uses default (2).		
Transmit a few signals so we're simulating real use. Let it finish vibrating.		
Open SubGHz		
Go to "read"		
Set frequency to 433.92MHz and modulation to AM650.		
Transmit another signal.		
We have captured it... ...PTO		

Rolling Flaws: Github Jamisonderek

Exploiting Gaps 2/2

Rolling Flaws

Info	Rolling Flaws Flipper	Hacking Flipper
Select the captured signal and you'll see more details.		
We're going to send a few more signals now to simulate use of the real remote.		
Now we're going to be "out of range" and advance the count ~20.		
We're "back in range" and we can receive the Flipper's signals.		
Send two more signals from the hacking Flipper.		
The first one will say "future" and remain closed.		
The second will open as we have had two future codes in a row.		

We have exploited a feature which is there to mitigate against the count being advanced on the remote but not the receiver (e.g. kid mashing the button lots). **The real remote will now be out of sync** and the Flipper has taken over. It may well be possible to resync the real remote using the same method, but not guaranteed.

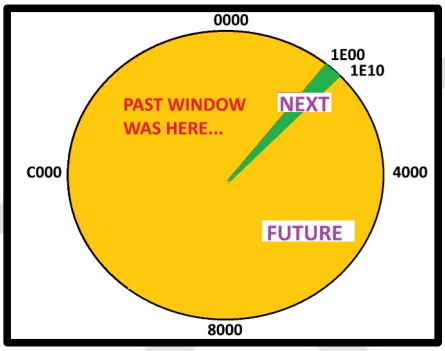
Rolling Flaws: Github Jamisonderek

Rollback Attack 1/3

Rolling Flaws

On some receivers, the future window is set to “all”, meaning any codes that aren’t within the next window are considered future codes. This is silly but it happens.

This means we can capture codes which should have expired and send two of them, exploiting the gap feature. We can even do this using a raw record if we can grab two codes in a row.



Info

Open a fresh Rolling Flaws. In the config, set the future window to “all”.

Transmit a few signals to get the count up.

Open SubGHz.

Select read raw.

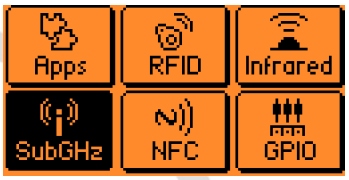
Set the config up as this.

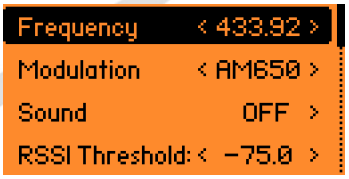
Go back to the record screen and press record.

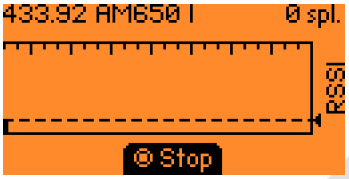
When ready, transmit two signals, a few seconds apart.

Rolling Flaws Flipper

Hacking Flipper







Rolling Flaws: Github Jamisonderek

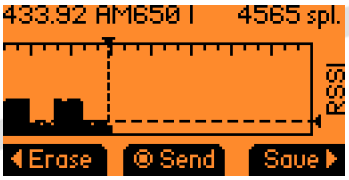
Rollback Attack 2/3

Rolling Flaws

Info

You should get a couple of signals like this. Press stop.

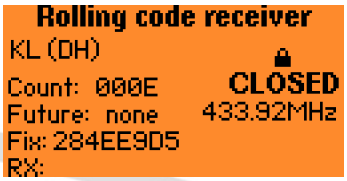
Rolling Flaws Flipper **Hacking Flipper**



Send a few more to simulate normal use.

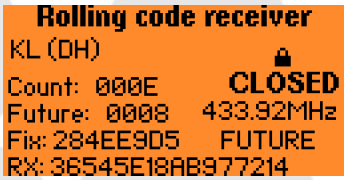
Go to transmit signal.

You'll see this screen, note the count it's expecting in my example is 000E (14).

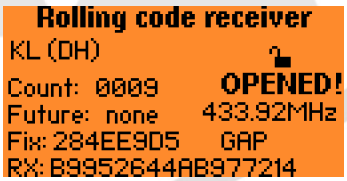


When ready, send your raw recording.

This will pop up with signal 1 and then be replaced with…



...this one, which is showing the count has gone back to 0009 and the door is open.



What happened?

PTO to find out, as there are a couple of things going on here.

Rollback Attack 3/3

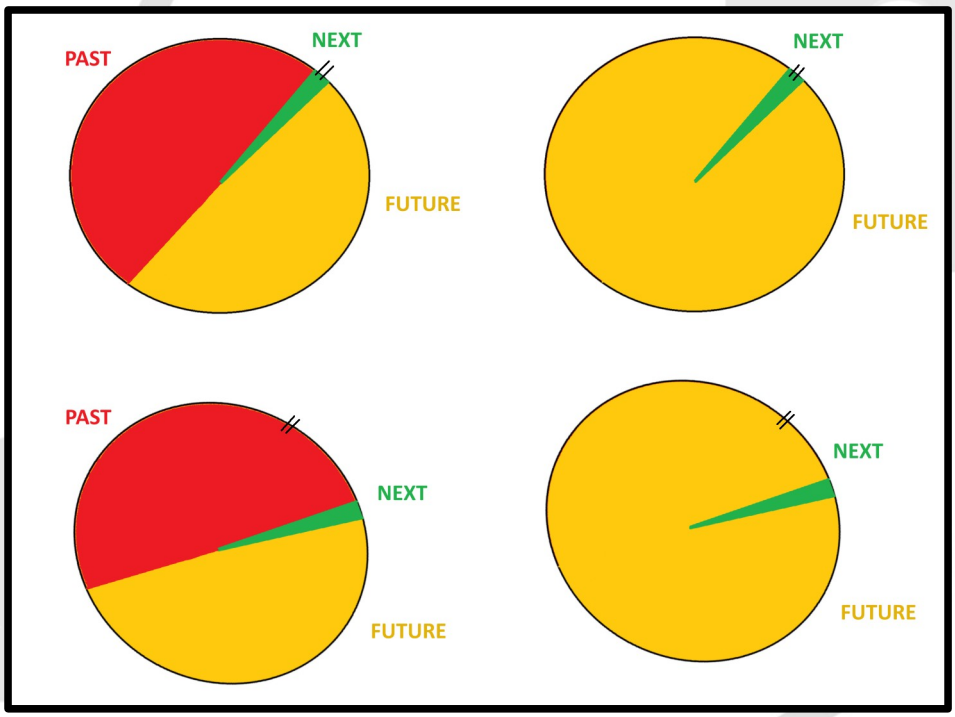
Rolling Flaws

What happened?

We exploited two issues in this attack.

One was the **removal of the "past" window** of codes, which means **any codes that aren't in the "next"** window are **classed as future** codes to be available as the system goes round and round the limited set of codes it can use (64,000).

The second is the **gap function**, where a remote might advance past the "next" window away from the receiver, for whatever reason (child mashing remote is one). This means that **two future codes will resync the receiver to the remote**.



In the above, we have the **beginning state at the top**, where we **record two codes** using the Flipper (black lines). As the system **invalidates old codes**, the "next" window moves along. When we **replay the older codes**, they should be **classed as being from the past**. But, with the **future codes set to "all"** those replayed codes are **considered future codes**.

Then **playing two of them in a row** says the to receiver that the count on the **remote** was **advanced beyond "next"** and the **receiver** should be **resynced** to the **current remote count** – this is the **"gap"** feature.