EMBEDDED

Stopping Brute Force PIN Attacks in BGScript

Last week, I wrote about BLE Security and said that I was able to brute force a device's PIN code in less than 17 minutes. If you're using PIN-based security in your Bluetooth device, I'm sure that sucks to hear. Here is a workaround (in BGScript) I have to help prevent that from happening.

My Suggestion

Taking a cue from bcrypt and other 'slow' hashes, let's just try to make it a pain in the ass to guess PIN codes. The goal is to reduce the 50 attempts per second to something more painful. Depending on how smart your hardware is, you can look into exponential backoff, or mac address-based whitelists/blacklists (although, these can be spoofed), etc etc...

My proposal is super simple (and intended for something like the BLE113 or BLE121LR in BGScript - which is not a powerful language). You get 5 wrong attempts in a row, then everyone is disconnected and locked out for 2 minutes. There you have it. That's it. No more brute force attacks. And here's the code.

```
# PIN security
dim incorrect_attempt_counter   # Number of incorrect PIN attempts on
dim is_connection_allowed       # 0 - ignore connection attempts, 1 -
dim cached_pin_code(8)          # The correct PIN code, populated som

const TICKS_PER_SECOND = 32760
const TOO_MANY_INCORRECT_ATTEMPTS_TIMEOUT_SECONDS = 120
const INCORRECT_ATTEMPTS_LIMIT = 5

procedure setup_connectivity()
    if is_connection_allowed then
        # Put module into discoverable/connectable mode
        call gap_set_mode(gap_general_discoverable, gap_undirected_co
    else
        # Put module into NON-discoverable/NON-connectable mode
        call gap_set_mode(gap_non_discoverable, gap_non_connectable)
    end if
end

event connection_disconnected(handle, result)
    # Determine if we can connect to the device
    call setup_connectivity()

    # ... Other stuff ...
end

event attributes_value(connection, reason, handle, offset, value_len,

    # Check whether pin code was correct,
    # If so, reset the incorrect attempts counter - if not, increment
    if handle = xgatt_open_door then
        if memcmp(value_data(0), cached_pin_code(0), 8) then
            incorrect_attempt_counter = 0
        else
            incorrect_attempt_counter = incorrect_attempt_counter + 1

            # Check the number of incorrect attempts - if it reaches
            if incorrect_attempt_counter >= INCORRECT_ATTEMPTS_LIMIT
                is_connection_allowed = 0
                call connection_disconnect(0)
                call hardware_set_soft_timer(TOO_MANY_INCORRECT_ATTEM
            end if

        end if
    end if

    # ... Other stuff ...
end

event hardware_soft_timer(handle)
    # When this timer goes off, we're re-allowing connections
    if handle = TOO_MANY_INCORRECT_ATTEMPTS_TIMER_HANDLE then
        is_connection_allowed = 1
        incorrect_attempt_counter = 0
        call setup_connectivity()
    end if

    # ... Other stuff ...
end
```

If you get a wrong attempt, then a success, you now have 5 more wrong attempts to play with (as in, the 5 attempt counter resets after a success).

If you think waiting 2 minutes is annoying, well, that's the point! You can make that number whatever you want - but the idea is that you're trying to stop people from breaking in using a brute force attack, and the person who knows the correct code should never run into this situation.

Another downside is that you could say that it makes it possible for a random person to lock you out of your own device, by just continually failing attempts and putting the whole device into lockdown... Yes, absolutely - but again, that's kinda the point. I'd rather EVERYONE get locked out, rather than let a malicious person in. And at least, this way you're well aware something shady is going on.

BGScript Bug

There is a bug I discovered in setup_connectivity going from discoverable to non-discoverable, back to discoverable. I noticed that for some reason, when you do this - the transmit power falls over and even next to a module, you'll see -100db (instead of -30db).

No one has been able to answer this yet, so my ghetto workaround is to just reset the module entirely inside of setup_connectivity, when trying to become discoverable again - so make sure your system_boot is solid. I'll be looking into this bug a bit more, and if it's something else, I'll update the above code.

Feature Photo credit: Ervins Strauhmanis / Foter / CC BY