kioubit.dn42 Crypto #2 Part 1 Write-up

2026-04-08

Challenge: http://kioubit.dn42/challenge/ch2/

> **Note:** This challenge is hosted on the dn42 network. You may need a peering connection or a VPN tunnel into dn42 to access the link above.

At first glance

The challenge presents a custom CAPTCHA. When it is activated, the frontend JavaScript loads a session from:

- `api/getSessionForUser?username=guest&is_guest=true`

and then verifies it with:

- `api/checkSolution?session=...&solution=...`

A sample session looked like this, with the ciphertext abbreviated for readability:

```
{
  "Captcha": "🦌 + 44455",
  "SessionData": {
    "Encrypted": "tWY4tD...qGb9mdkz7",
    "Metadata": "JmNhcHRjaGFfaW5kZXg9OSZpc192ZXJpZmllZD1mYWxzZQ=="
  },
  "SessionDataHmac": "PGq09Z41GoVbhFr6fsoubxQHrSc7+wAlwubzYXPktCA="
}
```

The base64-decoded metadata is:

```
&captcha_index=9&is_verified=false
```

At this stage, the obvious ideas were:

1. try to get a session with `is_guest=false`.
2. tamper with `is_verified` in the metadata.
3. replace `Captcha` with a trivial one.

None of these worked.

1. the server responds with “You are only allowed to create guest sessions using this api endpoint”.
2. `MAC authentication failed`.
3. whether the CAPTCHA is modified or not, the server returns `Incorrect captcha solution`.

The `captcha.js` file contains a hint: “A cryptographic solution is required which involves looking through the protocol used to verify the captcha response”

Looking into the encryption scheme

I created several sessions with different usernames and compared the `Encrypted` fields. Some examples are listed below:

username	Encrypted
`a`	`tWY4...c4z5rJtOHe7Mqm267...oSIL5wamyo4=`
`b`	`tWY4...c4z4zmzRkkePoAAFU...oSIL5wamyo4=`
`aa`	`tWY4...c4z4xyzyxFwktLoup...SL+TLJpOp6c=`
`aaaaaaaaaaaaaaaaaaaaaaaaa`	`tWY4...c4zwQ...AnNWK...FYlLmh...`
`aaaaaaaaaaaaaaabaaaaaaaaa`	`tWY4...c4zwQ...AnMvM...atjLmh...`

Observations:

1. The first block was the same for all usernames.
2. Some trailing blocks were identical when username length was the same.
3. Changing a part of the username only affected certain blocks.

These observations suggest that the plaintext is encrypted in ECB mode, with a structure something like this:

```
prefix | username | suffix
```

Recovering the suffix

Since the prefix and suffix are constant and the username is arbitrary, we can use a classic byte-at-a-time attack to recover the suffix.

```
&source=web&solution=12513026260501710149&guest_account=true
```

> **Note:** The solution changes over time, although I do not know how often or what triggers the change. Solve your own session to get the current solution.

Getting on the leaderboard

Once the hidden solution was recovered, the rest was simple:

1. Request a fresh session for the scoreboard name.
2. Submit the value to `checkSolution`.

The server returned a new verified session with `is_verified=true`.

Using that session with `api/controlPanel` returned:

```
OK - Logged in as guest user
Congratulations. You partially solved the challenge!
Username: Iris
Your username has been added to the leaderboard
```

And that is the end of the story.

Edited on 2026-04-08