All this text wasn’t written by me (Tiago a.k.a TigaxMT).

All of it was transcribed by me from HackerOne video lessons.

All credits and thanks go to them.

Crypto Wrap-Up

DETECTION OF ECB

If you have control of data being encrypted in some way (you don’t have to control all the data), then repetition is the easiest way to determine if ECB mode is in use.
Remember that with ECB, each block is encrypted independently.

If you put 47 ‘A’ characters in a row into a payload that’s encrypted, this will guarantee repetition if a block size of 16 bytes (128 bits) or less is used.
No matter how many characters are before or after those 47 As, you will always have enough to fill at least two blocks.

DETECTION OF BLOCK SIZE

Once we have repetition, determining the block size is trivial: Pick a character somewhere in the middle of your repeated ‘A’s and turn it into a ‘B’.
You will see some number of bytes change, or a gap in the repetition. That number of bytes is your block size.

DETECTION OF OFFSET

If you have an ECB payload and found repetition, the next step to exploitation is to figure out where in the string your data is being placed. To do this, we can exploit repetition again. The number of characters you can change at the beginning of your payload is the offset from the beginning of the whole encrypted string. The number of characters you can change at the end of your payload is the offset from the end.

NOTES ON PADDING ORACLES

1. Most importantly, padding oracles only come into play with CBC mode, due to the XOR chaining nature of the mode.
2. You must be able to manipulate the data. If there’s an HMAC (implemented properly!) then this is a no-go.

SUBNOTE

I say that an HMAC implemented properly prevents padding oracle exploitation. That’s correct, but remember, you **must validate the HMAC before decrypting**.
If it decrypts first, then you can find the padding.

PADDING ORACLE DETECTION

The only reliable way to detect padding oracles is to actually exploit them to an extent.
The best thing you can do is run through all 256 possibilities for the last byte of the second-to-last block- There should be no more than two values that give you different errors; everything else should be padding errors.

PADBUSTER

The Padbuster tool from GDS is great for exploiting padding oracles and can successfully decrypt the data in vulnerable payloads in the vast majority of cases:
http://blog.gdssecurity.com/labs/2010/9/14/automated-padding-oracle-attacks-with-padbuster.html

JUST SAY NO

Unless you and your thousand closest cryptographer friends are working together, Just Say NO to designing your own crypto protocols.

Even then, the likelihood of something going wrong is high.
Keyczar and NaCl (Networking and Crypto Library, not Native Client) both provide high-level APIs for crypto, rather than using primitives. Use them whenever humanly possible.

TLS AND PGP

Generally speaking:

If you have data in flight, use TLS (what used to be SSL).

If you have data at rest, use PGP.

These will cover the vast majority of cases.

MORTAL SINS OF CRYPTO

Do Not:

- MAC-then-Encrypt: You leave yourself open to attacks against your crypto implementation
- Use hashes instead of MACs: You enable hash extension attacks
- Reuse key-IV or key-nonce pairs: You open yourself up to a multitude of issues

- **EVER USE ECB**: Seriously, there’s never a good use case for ECB. You’ve seen how bad it is.