verify_email.sh

DOWNLOAD
Path in BitBucket:

Functionality: Given an email address, returns a value that can be used to discern whether the email is a real account.

Qualitative Assessment

- Situations when this script definitively tells us an address is valid
    - in-depth checking is active(not just checking for a valid domain), and the target email server responds with “Recipient OK” or a ‘250’ response.
- Situations when this script definitively tells us an address is invalid
    - the domain portion of an email address is invalid (has no mail exchange service)
    - if the mail exchanger service responds with 550, script returns “invalid”, no email server can respond with a 550 error if an email address is valid. (The server wouldn’t be able to accept legitimate emails)
- Gray areas
    - servers that always say Invalid, because they’re suspicious of scripts like this one
        - script will return an ‘inconclusive’ result in the following cases:
            - server responds with something other than “Recipient OK”, 250, 451, or 550
            - if server responds with a 553 error, strict anti-spam measures are in place and the email’s validity would be unable to be determined.

TODO: Quantitative Assessment that corresponds to the qualitative one

System Requirements:

- Linux/BSD/Unix, Bash (check first line for correct path), expect, nslookup, nc (netcat)
- For optimal functionality, the script should be run from a server with a static IP address because certain email servers like Hotmail and Yahoo will refuse connections from dynamic IP hosts.

Installation: (just unpack the archive and run the script!)

```
tar -xzf verify_email.tar.gz
```

Check the path for the bash binary:

```
which bash
```

Alter line 1 to reflect the correct path

```
#!/bin/bash
```

Usage:

`./verify_email.sh` *`email_address`* `[`*`relay_url`*`] [`*`true`*`|`*`false`*`]`

`email_address`: full email address to be submitted for validation

`relayurl`: optional, but ought to be a url that matches the server running the script. This is the url that is used for the 'HELO' command, some servers require a URL,
(e.g. HELO example.com)

*Argument 3 (**true**|**false**)*: if true, skips the in-depth email account test. This could be useful if all that is required is to determine whether a given email address has a valid domain with an MX (mail exchange) record. If this returns true, there is a fair chance the email is valid. Setting this value to true skips the in-depth email account verification process (which takes several seconds per email).

Return values:
`invalidEmail` : indicates malformed email address

`invalidDomain` : nslookup could not find an MX record for the domain

`valid` : email should be considered valid - if Argument 3 is true, this is also the return for valid email exchange domain.

`invalid` : got 550 response, email is definitely no-good

`inconclusive` : got unexpected response from email server, possibly anti-spam settings prevent this kind of access. In most cases, an inconclusive response should be treated as a valid email.

Procedure:
This script uses three methods to determine the validity of an email address.
1st test: Use a regular expression comparison to test whether it is a correctly formatted email.
2nd test: The script grabs the domain from the email, and tests (with nslookup) if the domain has a mail exchanger
3rd test: Attempts to log-in to the mail exchanger smpt service port (25) and executes commands to test the existence of the email account. After sending "rcpt to: *email*", if the server returns 250 or 451, the email is valid. If the server returns a 550 response, the email definitely does not exist. If it returns something else, then the result is "inconclusive" and it's up to the user to decide whether the email is valid.

Debugging:
If you get unexpected behaviour, uncomment line 89 to output the server response to a file called "outputlog.txt", be aware that, depending of the mail server, responses may include windows newlines (^M), so the unix 'cat' command might not show the whole file - use vi, emacs, or your

favourite text editor to view this file.