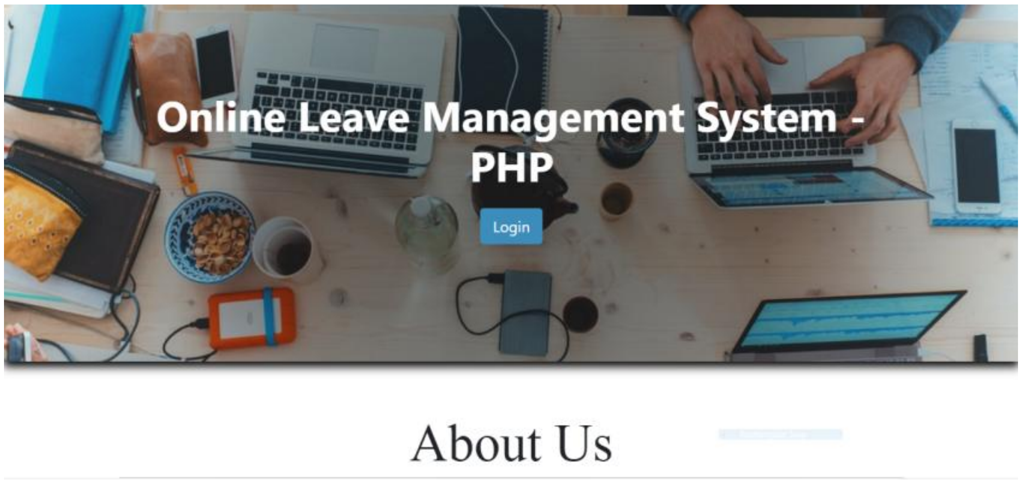
nu11secur1ty Update templeate_report.txt ... on Sep 1, 2021 History

..	
Capture.PNG	last year
PoC-CVE-nu11-03-SQL-baypass-Login-injection.py	last year
README.MD	last year
chromedriver.exe	last year
leave_system_0.zip	last year
templeate_report.txt	last year

README.MD

CVE-nu11-03

Online Leave Management System SQL-Injection-Bypass-Authentication:



Vendor:

- [+] href

Description:

The OLMS - PHP (by: oretnom23) v1.0 is vulnerable in the application /leave_system/classes/Login.php from remote SQL-Injection-Bypass-Authentication m0re info: https://portswigger.net/support/using-sql-injection-to-bypass-authentication. The parameter (username) from the login form is not protected correctly and there is no security and escaping from malicious payloads. When the user will sending a malicious query or malicious payload to the MySQL server he can bypass the login credentials and take control of the administer account.

Broken query:

```
	public function login(){
		extract($_POST);

		$qry = $this->conn->query("SELECT * from users where username = '$username' and password = md5('$password') ");
		if($qry->num_rows > 0){
			foreach($qry->fetch_array() as $k => $v){
				if(!is_numeric($k) && $k != 'password'){
					$this->settings->set_userdata($k,$v);
				}

			}
```

The fix, but not strong enough!

```
  public function login(){
		extract($_POST);
```

```
                    $qry = $this->conn->query("SELECT * from users where username = ('$username') and password = md5('$password') ");
                    if($qry->num_rows > 0){
                            foreach($qry->fetch_array() as $k => $v){
                                    if(!is_numeric($k) && $k != 'password'){
                                            $this->settings->set_userdata($k,$v);
                                    }

                            }
```

Proof:

- [+]href

Conclusion and solution of the problem:

- [+]href

BR

- [+] @nu11secur1ty