

# **Insufficient Logging & Monitoring**

*White Hats: Team Members*

Banks Cargill - [cargillb@oregonstate.edu](mailto:cargillb@oregonstate.edu)

Frederick Eley - [eleyf@oregonstate.edu](mailto:eleyf@oregonstate.edu)

Timothy Glew - [glewt@oregonstate.edu](mailto:glewt@oregonstate.edu)

## **Description**

Insufficient logging and monitoring, along with a lack of effective integration of incident response, make the final entry in OWASP's top 10 web application security risks list. By integrating systems that automatically log potential points of vulnerability, companies can review and learn from attacks that were successful against their systems. IBM Security reported in 2019, that the average time to detect a breach was 206 days with an additional average time to contain the breach being 73 days (IBM). That is a total of 279 days in which attackers can continue investigating, penetrating, and compromising systems. With proper monitoring and incident response integration, breaches can be detected earlier negating a large amount of potential damage.

There are several events that need to be logged to increase the odds of detecting a breach. To begin with, logs need to be generated in an easily "consumable" format. CLF or common log format is once such standardized text file format that is used by web servers that a centralized log management (CLM) system can consolidate and store in one accessible, user friendly interface. CLMs offer the ability to easily search the logs and generate alerts based on metrics defined.

At a minimum, all login, server-side input validation, and access control failures should be logged. They need to be logged with enough user context to identify any suspicious accounts and need to be stored long enough to allow for a delayed analysis with one year being the generally agreed upon minimum. Any high-value transactions should also have preventative controls for tampering or deletion, such as append-only database tables.

An effective monitoring and alerting system needs to be established so that the logging implemented in the last step can be used to identify suspicious activity. Once this is in place, an incident response and recovery plan should be created or adopted from an existing source such as NIST 800-61 Rev.2(OWASP top 10).

While taking these measures does not reduce the chance of being compromised, it does significantly reduce the impact. For both the application and its users' security, it is critical to implement logging and monitoring. Unlike our other tutorials, this guide will be focusing solely on the defense since logging and monitoring is only defensive in nature.

# How to Defend our Site

## Adding Log Statements

Python comes with a logging module in the standard library that can handle most log outputs. Since we knew from viewing Heroku's logs that anything printed (`sys.stdout` or `sys.stderr`) would appear in the logs, we could simply add log statements that would print to `stderr`. This way our users would not be able to see the statements, but we could use them to track how our users were navigating and utilizing the site. We specified that when the operating system's environment was Heroku, the output would be `stderr` and we would only log INFO level threats or above. This negates DEBUG level threats, which would be used to step through and correct code issues. We will therefore view INFO, WARNING, ERROR, and CRITICAL level logs in Heroku's logs.

```
if os.environ.get('HEROKU') is not None:
    stream_handler = logging.StreamHandler()
    webapp.logger.addHandler(stream_handler)
    webapp.logger.setLevel(logging.INFO)
    webapp.logger.info('Capstone Secure')
```

If we were hosting our site on another server that allowed logging to a file, we would have done so. Unfortunately, it is not something that Heroku allows but you should consider it as a great option if you are setting up a production level web application.

The next thing to handle was what information we wanted present for each log title. We set it for anything above level INFO again and formatted it to show the time, what level it was, the name of the file that called the log, the active thread, and finally the message.

```
logging.basicConfig(level=logging.INFO, format='%(asctime)s %(levelname)s %(name)s %(threadName)s : %(message)s')
```

With this completed, we just needed to add our specific log statements. If we knew this was such a vital part of protecting our webapp from the beginning, we would have been coding with log statements throughout development. Areas we needed to have additional log statements:

- Login
  - Failed login attempts from no matching username in database issue a "WARNING" level log, documenting the username.
  - Failed login attempts, issue "INFO" level log with attempt # and username if the username is in the database
  - If the user fails to login after three attempts, issue a "WARNING" level log
  - Successful logins print "INFO" level log with username
- Confirmation Email
  - If the link is invalid or expired, an "INFO" level log is issued mentioning it
  - If the email is already confirmed, an "INFO" level log is issued with the email

- If the email gets confirmed, an “INFO” level log is issued with the email
- Password Recovery
  - If the email entered is not in the database, a “WARNING” level log is issued with the email the user entered. This could be an attacker attempting to determine email addresses in our system
  - If the email was not confirmed before attempting the password reset, an “INFO” level log is issued with the email
- Password Reset
  - If the link is invalid or expired, an “INFO” level log is issued
  - If a password is reset, an “INFO” level log is issued with the email
- List Functions
  - List is added, an “INFO” level log is issued with user id, list name, and list description
  - List is deleted, an “INFO” level log is issued with user id and list id.
  - List is updated, an “INFO” level log is issued with user id, list id, list name, and list description
  - User views a list, an “INFO” level log is issued with the user id
  - If a user attempts to view a list not theirs, a “WARNING” level log is issued with the current user’s id
- Task Functions
  - User adds a task, an “INFO” level log is issued with the user id and list id
  - User deletes a task, an “INFO” level log is issued with the user id, list id, and task id
  - User views tasks, and “INFO” level log is issued with the user id
  - User updates a task, an “INFO” level log is issued with the user id and task id

Here are a few examples of how the logs appear in our application code:

```
webapp.logger.INFO("Login unsuccessful, attempt #s, username: %s", result[0][6]+1, username )
```

```
webapp.logger.WARNING("Invalid access to view list. userid: %s", current_user.id)
```

```
webapp.logger.INFO("Confirmation link invalid or expired")
```

You’ll notice as you look through the code and these examples, there are times we log information based on results of requests to the database and times that we log with information provided by the user in form submission. This should allow us to better track users that are searching for vulnerabilities and help us take preventative measures.

## Centralized Log Management

Heroku only stores its most recent 1500 lines of logs so we needed to find a way to store them for a set period and be able to search and set notifications based on results. Papertrail is a cloud hosted log management system that when added to a Heroku application, will pull log files, and store them for 7 days for free. It also enables searching and setting alerts for specific findings in the files. Here is an example of what the log file looks like in the Papertrail UI:


*You can easily see that user\_id 3 is adding a list at 9:24 am on May 30th.*

```
May 30 09:24:40 osu-capstone-project app/web.1 Executing SELECT * FROM users WHERE `user_id` = '3' with ()
May 30 09:24:40 osu-capstone-project app/web.1 2020-05-30 16:24:40,438 INFO starter_website.webapp MainThread :
Adding list. user id: 3, list_name: To do today, list_desc: May 30th'
May 30 09:24:40 osu-capstone-project app/web.1 10.81.229.85 - - [30/May/2020:16:24:40 +0000] "POST /add_list
HTTP/1.1" 302 217 "https://osu-capstone-project.herokuapp.com/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36"
May 30 09:24:40 osu-capstone-project app/web.1 Executing SELECT * FROM users WHERE `user_id` = '3' with ()
May 30 09:24:40 osu-capstone-project app/web.1 Executing SELECT `username` FROM users WHERE `user_id` = '3' with ()
May 30 09:24:40 osu-capstone-project app/web.1 Executing SELECT * FROM `lists` WHERE `user_id` = '3' with ()
May 30 09:24:40 osu-capstone-project app/web.1 10.81.229.85 - - [30/May/2020:16:24:40 +0000] "GET /home HTTP/1.1"
200 3432 "https://osu-capstone-project.herokuapp.com/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36"
May 30 09:24:40 osu-capstone-project heroku/router at=info method=GET path="/home" host=osu-capstone-
project.herokuapp.com request_id=82e9e232-15ca-462f-92ab-64e6675e8929 fwd="173.53.115.228" dyno=web.1 connect=2ms
service=85ms status=200 bytes=3915 protocol=https
```

When searching, you can specify what you want to find. As a test, let's try to find when users logged in to the site.

*Here we can see that user "username" logged in on May 28th at 18:48 and user "secure" logged in on May 30th at 9:23.*

```
May 28 18:48:28 osu-capstone-project app/web.1 2020-05-29 01:48:28,185 INFO starter_website.webapp MainThread : Login
successful, username: username'
May 30 09:23:49 osu-capstone-project app/web.1 2020-05-30 16:23:49,182 INFO starter_website.webapp MainThread : Login
successful, username: secure'
```




## Notifications

The last thing we wanted to accomplish is enabling some sort of log management system so that we would be notified when something strange was happening on our site. We accomplished this using Papertrail. We set alerts that when found, are sent to our emails. Papertrail allows the user to set the frequency and times for the alerts and since our site isn't receiving much traffic, we decided once per day is enough. We wanted alerts for three main occurrences:

1. When the site goes down
2. Anytime that "WARNING" is found in the logs
3. Anytime "Error" is found in the logs

Here is an example of what the alert setup looks like for warnings:

Edit Alert

Name: [WARNING](#) 

Query:

Alert Conditions


Frequency

☐ 1 minute ☐ 10 minutes ☐ 1 hour ☒ 1 day


Next alert: Sunday, May 31 at 5:00 AM (approximately)

Trigger when

☒ at least  new events match

☐ no new events match 

Send events with timestamps in



Event timestamps will use this zone

Email Details

Recipients

Separate multiple addresses with commas.

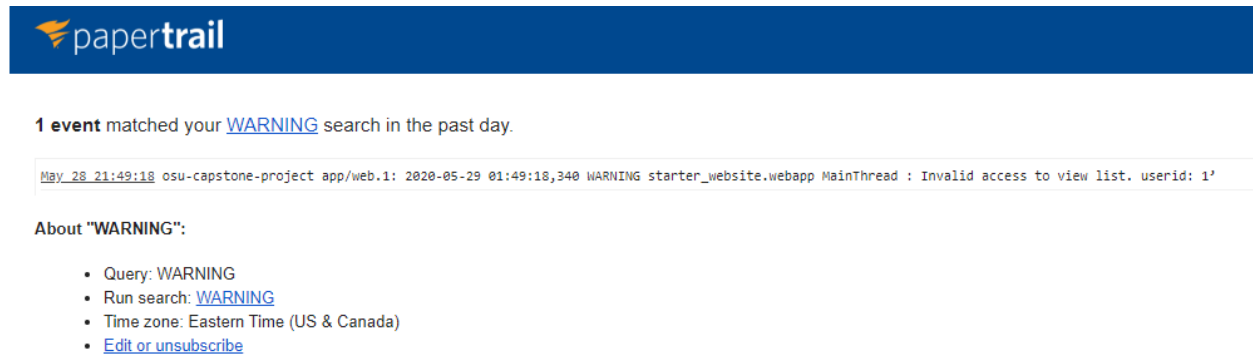
Save Changes

Send Test Data

Deactivate

Delete

And the resulting email that we received from a warning:



It is easy to see how logging and management of the logs is an essential part of any web application, and even more so for ones that manage sensitive user data.

## References

### *General*

[https://owasp.org/www-project-top-ten/OWASP\\_Top\\_Ten\\_2017/Top\\_10-2017\\_A10-Insufficient\\_Logging%252526Monitoring](https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A10-Insufficient_Logging%252526Monitoring)

[https://cheatsheetseries.owasp.org/cheatsheets/Logging\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html)

[https://www.ibm.com/downloads/cas/ZBZLY7KL?\\_ga=2.148238199.1762516747.1577395260-1128561362.1577395260](https://www.ibm.com/downloads/cas/ZBZLY7KL?_ga=2.148238199.1762516747.1577395260-1128561362.1577395260)

### *Heroku logging*

<https://devcenter.heroku.com/articles/logging>

<https://elements.heroku.com/addons/papertrail>

<https://help.papertrailapp.com/kb/configuration/configuring-centralized-logging-from-python-apps/#logging-via-the-heroku-add-on>

### *Python logging*

<https://docs.python.org/3/library/logging.html>

<https://docs.python.org/3/howto/logging.html>

<https://flask.palletsprojects.com/en/1.1.x/logging/>