Insufficient Logging & Monitoring

White Hats: Team Members
Banks Cargill - cargillb@oregonstate.edu
Frederick Eley - eleyf@oregonstate.edu
Timothy Glew - 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 are able to 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 a number of 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.

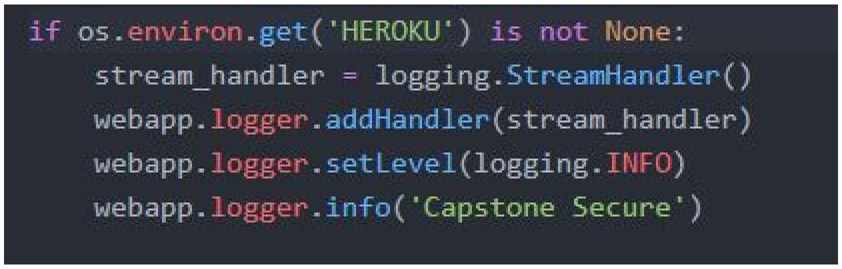
An effective monitoring and alerting system needs to be established so that the logging implemented in the last step can be used to identify suspicious activity. Once this is in place, an incident response and recovery plan should be created or adopted from an existing source such as NIST 800-61 Rev.2(OWASP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

While taking these measures doesn't 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

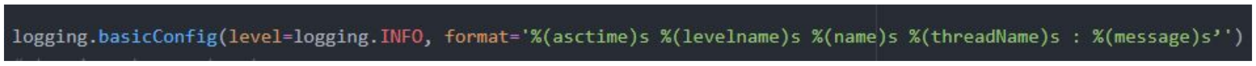
Python comes with a logging module in the standard library that can handle most log outputs. Since we knew from viewing Heroku’s logs that anything printed (sys.stdout or sys.stderr) would appear in the logs, we could simply add log statements that would print to stderr. This way our users wouldn’t be able to see the statements but we could use them to track how our user’s 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’s 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 get 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 is an area where someone could be trying to

Adding this on Saturday(BC)
[Detailed explanation (with screenshots or code snippets) of how we have patched the vulnerability]

References

General:
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://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/