Web Log Analysis Report

1. Data overview

The logs describe activity on the web server over several days at the end of October and in November 2025.
We have four kinds of data:

- page_stats.csv – how many times each page was requested, plus how many unique IPs and browsers visited it.
- error_timeline.csv – number of errors per hour (bucketed by date and hour like “Thu Nov 13 17”).
- error_types.csv – just one type called `"other"` with a very large count.
- access_timeline.csv – normal (successful) hits per hour (timestamps like “06/Nov/2025:20”).

Together, they show which pages are popular and which time periods look suspicious or attack-like.

2. Popular and interesting pages

From page_stats.csv, some pages clearly stand out:

- `/~achernii/api/signin.php` – 559 hits
  This is by far the most-used dynamic page. It’s a sign-in API, so high traffic here is suspicious, because normal users don’t usually hit an API hundreds of times.
- Static assets of `~achernii`
  - `~achernii/css/style.css` – 222 hits
  - `~achernii/img/logo.png` – 151 hits
  - `~achernii/js/theme.js` – 136 hits
  - `~achernii/js/auth_tools.js` – 246 hits
    These are typical front-end files. High counts here match a lot of page

loads for the `achernii` project.

- Other login pages
  - `~ycao/login.php` – 142 hits
  - `~mznaien/login.php` – 60 hits
  - `~wasin/login.php` – 37 hits
  - `~nakbogha/admin/login.php` – 34 hits
  - `/login.php` (root) – 20 hits
    All of these are authentication endpoints. Seeing many different login and admin pages requested suggests someone (or some script) is systematically probing sites.
- "Maintenance" and admin-style pages
  - `~achernii/maintenance.php` – 98 hits
  - `~mznaien/maintenance.php` – 28 hits
  - `~azinovev/maintenance.html` – 26 hits
  - `~wasin/maintenance.php` – 26 hits
  - `~kjurabaev/maintenance.html` – 11 hits
  - plus `/server-info`, `/admin/dashboard.php`, `~eyavuz/admin/login.php`, etc.
    Attackers often look for maintenance/admin URLs, so high numbers here also look like scanning.
- Log files being accessed directly
  - `~ycao/access_log.txt`, `~ycao/error_log.txt`
  - `~ycao/kchoi_access_log.txt`, `~ycao/kchoi_error_log.txt`

    - `~fribadenei/flask_test_session.log`
      These are not normal public resources. Requests to log files usually mean someone is trying to read internal information or just testing what is exposed.

Overall, a lot of different student projects are touched, but the heaviest concentration of dynamic requests is on `~achernii` and on various login/admin pages.

3. Error timeline – when the server was unhappy

From error_timeline.csv there are three main spikes:

- Thu Nov 13 17 – 25 340 errors
- Thu Nov 13 19 – 3 599 errors
- Thu Nov 13 20 – 2 704 errors

Together, this is a huge wave of errors in a short window on Thu Nov 13 in the early evening.

There are also smaller spikes:

- Mon Nov 10 15 – 816 errors
- Wed Nov 05 22 – 126 errors
- Tue Nov 11 17 – 157 errors

All other time buckets are much smaller (mostly tens of errors). So:

- Thu Nov 13 looks like a big attack or broken script.
- Nov 5, 10, 11 show some medium-sized bursts that could be testing or smaller-scale attacks.

From error_types.csv we only see one category:

- `"other"` – 33 686 errors total

So all these failures fall into a generic bucket. We cannot see exactly which HTTP error codes they are, but the timing pattern is very clear: most of the errors are concentrated around Thu Nov 13 17–20.

4. Access timeline – when traffic was heaviest

From access_timeline.csv, we can see when the server was most active. Some high-hit periods:

- 31/Oct/2025:00–01 – ~55 hits total (27 + 28)
- 01/Nov/2025:10 – 276 hits
- 04/Nov/2025 09–10 – 101 and 105 hits
- 05/Nov/2025 evening – from 16:00 to 22:00 hits steadily rise and peak at 196 (05/Nov/2025:22)
- 06/Nov/2025 late night/evening
    - 03:00 – 221 hits
    - 20:00 – 398 hits
    - 23:00 – 291 hits
- 11/Nov/2025 18:00 – 668 hits
- 13/Nov/2025 20:00 – 445 hits

So clearly traffic is not flat over time. Instead, we see waves of heavy activity:

1. First wave around Nov 5–6 (late afternoon and night).
2. A large spike on Nov 11 at 18:00.
3. Another strong spike on Nov 13 around 20:00, which overlaps with the massive error spike the same day.

The fact that hits and errors both spike on Thu Nov 13 strongly suggests a scripted attack or an automated evaluation script that is generating a lot of failing requests in a very short time.

5. Likely attack / behaviour pattern

Putting everything together, a reasonable story looks like this:

1. Automated scanning across many student projects
   Someone (or some script) is crawling many URLs under `/~username`. It touches:
   - index pages (`/`, `~user/`, `index.html`, etc.),
   - login pages (`login.php`, `signin.php`, `admin/login.php`),
   - maintenance pages (`maintenance.php`, `maintenance.html`),
   - admin/dashboard and CGI scripts (`run_query.py`, `form_generator.py`, `process_form.py`),
   - and even log files (`access_log.txt`, `error_log.txt`, etc.).
2. This is typical behaviour of:
   - a vulnerability scanner,
   - a grading/evaluation script that tests each project,
   - or a bot doing brute-force and information gathering.
3. Focus on authentication endpoints
   The extremely high hit count on `~achernii/api/signin.php` (559 hits) and the many other login/admin URLs suggest that login functionality is the main target.
   This could be:
   - brute-forcing passwords,
   - testing input validation,

    - or just repeatedly calling APIs with bad parameters (leading to errors).

4. Huge error spike on Thu Nov 13
   The combination of:
    - massive errors at Thu Nov 13 17–20
    - plus high traffic from the access timeline on the same date
      strongly points to a single intense event.
      This might be:
    - a broken or misconfigured test script that spams invalid requests, or
    - a real attack (e.g., brute-force login, path fuzzing, or some script hitting non-existing endpoints).
5. Because most of the errors are lumped into `"other"`, we cannot say exactly which HTTP status codes are involved, but the pattern is clearly abnormal compared to the rest of the week.
6. No single ordinary user would cause this pattern
   A human user would not:
    - visit dozens of different students' subdirectories,
    - try various log files,
    - access many different login/admin URLs,
    - and generate tens of thousands of errors in a short time.
7. So this is almost certainly scripted / automated traffic.

6. Short summary and recommendations

- There is clear evidence of automated scanning and probing across many student web projects.
- The heaviest focus seems to be:

    - `~achernii` (forum project: many API and JS calls, and a huge number of sign-in requests),
    - several other students' login and admin pages (`~ycao/login.php`, `~mznaien/login.php`, `~nakbogha/admin/login.php`, `~eyavuz/admin/login.php`, etc.),
    - and sensitive files like log files and server info.
- The most critical time window is Thu Nov 13, 17:00–20:00, when both hits and errors spike strongly. That is the most likely attack window or test run.

If this were a real system, I would recommend:

1. Rate limiting on login and sign-in APIs (like `api/signin.php`, `login.php`, `admin/login.php`) to slow down brute-force attempts.
2. Hiding or protecting internal files like log files, test scripts, and `server-info`.
3. Monitor and alert when errors per hour suddenly jump by orders of magnitude (like Thu Nov 13).
4. Centralize authentication instead of having many unprotected login pages scattered under different user directories.

Web Log Analysis Report

1. Data overview

The logs describe activity on the web server over several days at the end of October and in November 2025.
We have four kinds of data:

- page_stats.csv – how many times each page was requested, plus how many unique IPs and browsers visited it.
- error_timeline.csv – number of errors per hour (bucketed by date and hour like "Thu Nov 13 17").
- error_types.csv – just one type called `"other"` with a very large count.

- access_timeline.csv – normal (successful) hits per hour (timestamps like “06/Nov/2025:20”).

Together, they show which pages are popular and which time periods look suspicious or attack-like.

2. Popular and interesting pages

From page_stats.csv, some pages clearly stand out:

- `/~achernii/api/signin.php` – 559 hits
  This is by far the most-used dynamic page. It’s a sign-in API, so high traffic here is suspicious, because normal users don’t usually hit an API hundreds of times.

- Static assets of `~achernii`
  - `~achernii/css/style.css` – 222 hits
  - `~achernii/img/logo.png` – 151 hits
  - `~achernii/js/theme.js` – 136 hits
  - `~achernii/js/auth_tools.js` – 246 hits
    These are typical front-end files. High counts here match a lot of page loads for the `achernii` project.

- Other login pages
  - `~ycao/login.php` – 142 hits
  - `~mznaien/login.php` – 60 hits
  - `~wasin/login.php` – 37 hits
  - `~nakbogha/admin/login.php` – 34 hits
  - `/login.php` (root) – 20 hits
    All of these are authentication endpoints. Seeing many different login and admin pages requested suggests someone (or some script) is

systematically probing sites.

- “Maintenance” and admin-style pages
  - `~achernii/maintenance.php` – 98 hits
  - `~mznaien/maintenance.php` – 28 hits
  - `~azinovev/maintenance.html` – 26 hits
  - `~wasin/maintenance.php` – 26 hits
  - `~kjurabaev/maintenance.html` – 11 hits
  - plus `/server-info`, `/admin/dashboard.php`, `~eyavuz/admin/login.php`, etc.
    Attackers often look for maintenance/admin URLs, so high numbers here also look like scanning.
- Log files being accessed directly
  - `~ycao/access_log.txt`, `~ycao/error_log.txt`
  - `~ycao/kchoi_access_log.txt`, `~ycao/kchoi_error_log.txt`
  - `~fribadenei/flask_test_session.log`
    These are not normal public resources. Requests to log files usually mean someone is trying to read internal information or just testing what is exposed.

Overall, a lot of different student projects are touched, but the heaviest concentration of dynamic requests is on `~achernii` and on various login/admin pages.

3. Error timeline – when the server was unhappy

From error_timeline.csv there are three main spikes:

- Thu Nov 13 17 – 25 340 errors

- Thu Nov 13 19 – 3 599 errors
- Thu Nov 13 20 – 2 704 errors

Together, this is a huge wave of errors in a short window on Thu Nov 13 in the early evening.

There are also smaller spikes:

- Mon Nov 10 15 – 816 errors
- Wed Nov 05 22 – 126 errors
- Tue Nov 11 17 – 157 errors

All other time buckets are much smaller (mostly tens of errors). So:

- Thu Nov 13 looks like a big attack or broken script.
- Nov 5, 10, 11 show some medium-sized bursts that could be testing or smaller-scale attacks.

From error_types.csv we only see one category:

- `"other"` – 33 686 errors total

So all these failures fall into a generic bucket. We cannot see exactly which HTTP error codes they are, but the timing pattern is very clear: most of the errors are concentrated around Thu Nov 13 17–20.

4. Access timeline – when traffic was heaviest

From access_timeline.csv, we can see when the server was most active. Some high-hit periods:

- 31/Oct/2025:00–01 – ~55 hits total (27 + 28)
- 01/Nov/2025:10 – 276 hits

- 04/Nov/2025 09–10 – 101 and 105 hits
- 05/Nov/2025 evening – from 16:00 to 22:00 hits steadily rise and peak at 196 (05/Nov/2025:22)
- 06/Nov/2025 late night/evening
    - 03:00 – 221 hits
    - 20:00 – 398 hits
    - 23:00 – 291 hits
- 11/Nov/2025 18:00 – 668 hits
- 13/Nov/2025 20:00 – 445 hits

So clearly traffic is not flat over time. Instead, we see waves of heavy activity:

1. First wave around Nov 5–6 (late afternoon and night).
2. A large spike on Nov 11 at 18:00.
3. Another strong spike on Nov 13 around 20:00, which overlaps with the massive error spike the same day.

The fact that hits and errors both spike on Thu Nov 13 strongly suggests a scripted attack or an automated evaluation script that is generating a lot of failing requests in a very short time.

5. Likely attack / behaviour pattern

Putting everything together, a reasonable story looks like this:

1. Automated scanning across many student projects
   Someone (or some script) is crawling many URLs under `/~username`. It touches:
    - index pages (`/`, `~user/`, `index.html`, etc.),

- login pages (`login.php`, `signin.php`, `admin/login.php`),
- maintenance pages (`maintenance.php`, `maintenance.html`),
- admin/dashboard and CGI scripts (`run_query.py`, `form_generator.py`, `process_form.py`),
- and even log files (`access_log.txt`, `error_log.txt`, etc.).

2. This is typical behaviour of:
   - a vulnerability scanner,
   - a grading/evaluation script that tests each project,
   - or a bot doing brute-force and information gathering.

3. Focus on authentication endpoints
   The extremely high hit count on `~achernii/api/signin.php` (559 hits) and the many other login/admin URLs suggest that login functionality is the main target.
   This could be:
   - brute-forcing passwords,
   - testing input validation,
   - or just repeatedly calling APIs with bad parameters (leading to errors).

4. Huge error spike on Thu Nov 13
   The combination of:
   - massive errors at Thu Nov 13 17–20
   - plus high traffic from the access timeline on the same date
     strongly points to a single intense event.
     This might be:
   - a broken or misconfigured test script that spams invalid requests, or
   - a real attack (e.g., brute-force login, path fuzzing, or some script hitting non-existing endpoints).

5. Because most of the errors are lumped into `"other"`, we cannot say exactly which HTTP status codes are involved, but the pattern is clearly abnormal compared to the rest of the week.

6. No single ordinary user would cause this pattern
   A human user would not:
   - visit dozens of different students' subdirectories,
   - try various log files,
   - access many different login/admin URLs,
   - and generate tens of thousands of errors in a short time.

7. So this is almost certainly scripted / automated traffic.

6. Short summary and recommendations

- There is clear evidence of automated scanning and probing across many student web projects.
- The heaviest focus seems to be:
  - `~achernii` (forum project: many API and JS calls, and a huge number of sign-in requests),
  - several other students' login and admin pages (`~ycao/login.php`, `~mznaien/login.php`, `~nakbogha/admin/login.php`, `~eyavuz/admin/login.php`, etc.),
  - and sensitive files like log files and server info.
- The most critical time window is Thu Nov 13, 17:00–20:00, when both hits and errors spike strongly. That is the most likely attack window or test run.

If this were a real system, I would recommend:

1. Rate limiting on login and sign-in APIs (like `api/signin.php`, `login.php`, `admin/login.php`) to slow down brute-force attempts.

2. Hiding or protecting internal files like log files, test scripts, and `server-info`.

3. Monitor and alert when errors per hour suddenly jump by orders of magnitude (like Thu Nov 13).

4. Centralize authentication instead of having many unprotected login pages scattered under different user directories.