RAAD

Project 3: Modular Intrusion Prevention System
Rong Yu, Amy Yu, Anna Rooks, Doeun Kim

Team Members and Contributions

Rong Yu (rong.yu@stonybrook.edu)

- Designed and implemented the backend, the database and the interactions between the two.
- Implemented the backend API entry points for the front-end and the logger.
- Provided the GitHub repository and the virtual machine that's used to configure all the settings.

Amy Yu (amy.yu.1@stonybrook.edu)

- Implemented a parser to process SSH logs and send relevant information to the backend
- Installed Drupal, MediaWiki, and phpMyAdmin to run on Apache
- Configured Drupal, MediaWiki, and phpMyAdmin to record login attempts
- Wrote an extension for MediaWiki to produce authentication logs

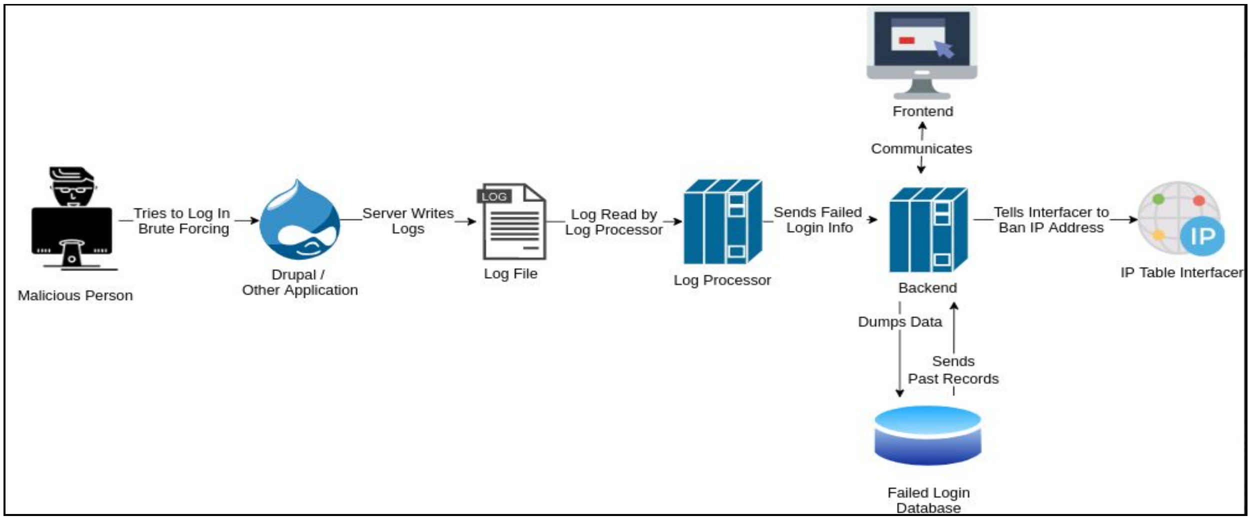
Anna Rooks (Anna.R.Li@stonybrook.edu)

- Wrote a simplified iptables interface to ban/unban IPs
- Wrote configuration script to configure password, start script to start all components
- Provided assistance for general server software usage.

Doeun Kim (doeun.kim@stonybrook.edu)

- Designed the whole UI/UX components of the web application
- Developed front-end side of the web application using HTML, CSS, JavaScript, jQuery, BootStrap, and Python3/Django
- Connected back-end API and front-end UI using AJAX
- Tested the whole web application, debugged front-end issues, and reported back-end API bugs
- Provided Slack workspace and Trello board for the smooth team communication and collaboration

System Architecture



Architectural Design Flow

Our original idea was to do a full Elastic Stack (consisting of Filebeat, Logstash, and Elasticsearch) because Filebeat automatically ships logs to Logstash in real-time and Logstash automatically sends the essential information about IP address to be blocked to the storage Elasticsearch based on the processed logs.

Besides processing logs, we also need to allow for an admin to log in and to track the banned IPs. These functionalities are normally done using a RDBMS or NoSQL DB. However, we soon realized that Elastic Stack might be a bit heavy and an inefficient method for the log processing part. So we decided to find an alternative method.

To replace Filebeat and Logstash, we initially found pyinotify, a Python module that can monitor files on Linux. After using pyinotify, we found out that it detects *if* a file has been changed, but does not show *what* those changes are. We switched to using the subprocess module for calling the Unix `tail -F` command. Moreover, instead of using both Elasticsearch and a standard database, we decided to just store everything on a standard database. Between using an RDBMS, and NoSQL, we chose NoSQL because it maximizes developer productivity.

Additionally, we decided to use Django for the front-end, Flask for the back-end, and python-iptables for the IP table interface. To be able to utilize these frameworks and modules with a single version of Python, we've decided to use Python version 3.6.

Architecture Explanation (According to Previous Diagram)

Log File

- Drupal, phpMyAdmin, MediaWiki, and SSH all generate authentication logs whenever someone attempts to log in, both in failure and in success.
- This means whenever a malicious user tries to get authenticated, the log file is updated.

Log Processor

- The log processor reads the logs generated, parses it for the relevant info, (the IP address, the time of authentication, which application they tried to login into, whether it failed or not), and passes it to the backend to act upon (maybe ban), and to store.

Backend (Data processor)

- The backend is basically the connector, glue and translator of the whole system.
- It maintains a history of failed attempts and their types in a database, and whenever an IP crosses the threshold for any single type (e.g. SSH, Drupal, etc.), it utilizes the IP Table Interface to ban that IP.
- It maintains stateful information about the application, like the thresholds, the banned IP addresses, along with the time set for their whitelisting.
    - The info is stored in a database so even if the application crashes, upon restart, the timed-out blacklisted IP addresses could be whitelisted.

Failed Login Database

- \`records\` table stores all the relevant info for all failed login attempts.
- \`ban\` table stores all the banned IP addresses, and the time in which they should be whitelisted.
- \`hash\` table stores the hash of the admin's password, and the salt.

IP Table Interfacer (System Control)

- The IP Table Management is the system that actually blacklists and whitelists IP addresses by interfacing with the Linux firewalls, to commit and delete rules.

Frontend (Service)

- Generates the admin web interface, enabling the changing of thresholds, changing of passwords, and the addition and the removal of IP address from the blacklist.

System Components

Log Processor

Technologies

- subprocess https://docs.python.org/3/library/subprocess.html
    - A Python module that is used to run the Unix tail -F command, which displays new log messages in real-time, to monitor log files
    - Replaces our initial implementation using the Python module pyinotify: https://pypi.org/project/pyinotify/
    - Reference: https://stackoverflow.com/questions/12523044/how-can-i-tail-a-log-file-in-python

Functions

- Detects changes in auth.log and other appropriate log files
- Parses new log messages to identify successful and failed login attempts
- Sends source IP address and timestamp of login attempts to the backend

Backend

Technologies

- Flask
    - Super lightweight framework used for the backend server.
    - Technically not supposed to be a production server, because it can only handle one request, but only an admin is supposed to have access, so that point becomes moot.
- MongoDB
    - The alternative to MongoDB would have been either SQL or some other less known NoSQL.
        - We went with MongoDB because it was the most popular and the easiest to set up and use.
- PyMongo
    - Most popular python library that interfaces with MongoDB directly.
- threading.Timer
    - Python library to create timed events that would call some function with some set arguments at x time in the future.
    - Alternative would have been something like Python Advanced Scheduler, but threading.Timer is something built into python already, there’s no point in using an external library when an internal one works.

Functions

- Abstracts away the database aspect of the project, because anything that requires the database goes through and is handled by the server.
- Provides API for front end to query the thresholds, reset the password, and to log in.
- Provides API for the logger to dump log data, e.g. who failed to log in at what thing.
    - After every data dump, checks to see if it was successful or not, and if it wasn't, if that failure triggers a ban.
    - Uses the IP table interfacer to ban an IP address.
- Stores all stateful info, like the thresholds, the IP addresses that have been banned, and the time in which to free them, in the DB.

Frontend

Technologies

- HTML5/CSS3/JavaScript
    - Used to form and decorate UI components and make special effects on web pages.
- BootStrap
    - The tutorial website https://getbootstrap.com/ was referred.
    - Chosen to use their well-designed and pretty buttons and alert messages by importing https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js and https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css through the script and link tags.
- jQuery
    - To get the jQuery file, entered https://jquery.com/download/, clicked "Download the compressed, production jQuery 3.4.1," and downloaded the compressed jQuery file by doing right-click, click "Save As," and download as JavaScript file. Imported jQuery file through script tags in all HTML head tags.
    - Used to simplify the API server request code blocks and DOM manipulation code blocks.
- Python3/Django
    - Two tutorial websites, https://www.djangoproject.com/start/ and https://www.tutorialspoint.com/django/index.htm, were referred to in order to learn how to do web development with Django.
    - Chosen to accelerate web application development with easy scale extension and clean modularization.
    - Utilized internal Python modules.

        - requests(https://2.python-requests.org/en/master/) and json(https://docs.python.org/3.6/library/json.html) were used to call Backend APIs and get results in JSON format.
        - urllib3(https://urllib3.readthedocs.io/en/latest/) was used to disable getting an InsecureRequestWarning exception due to usage of unverified SSL certificate.
        - hashlib(https://docs.python.org/3.6/library/hashlib.html?highlight=hashlib) and hmac(https://docs.python.org/3.6/library/hmac.html?highlight=hmac) were used to create the hashed data of concatenated password with salt by using HMAC hashing algorithm in order to store the password securely in our database.
    - Utilized external Django package.
        - django-sslserver(https://github.com/teddziuba/django-sslserver) was used to run a web application over https.

Functions

- Contains features of login, changing password, viewing blocked IP address list, unblocking the IP address, adding blacklist IP address, getting current thresholds, and changing thresholds. The use case for each feature is uploaded in https://github.com/RongYu98/RAAD/blob/master/Frontend_Use_Cases.pdf.
- Communicate with Back-end API application to get, set, update, and delete data from the database.
- Render web pages with data provided by Back-end API application.

IP Table Interfacer

Technologies

- Linux iptables
    - Firewall used in interfacing to "ban" IPs
- python-iptables https://github.com/ldx/python-iptables
    - Used to communicate with iptables indirectly, with a simplified interface

Functions

- Add a rule to the tables to drop packets from a provided IP address
- Remove all rules associated with a specified IP address

Guides

Installation Guide

Note: Development and testing were performed on Ubuntu 18.04

1. Install `git`

2. Clone the GitHub repository.

   ```
   $ git clone https://github.com/RongYu98/RAAD
   ```

3. Install `python3.6`, `pip3`, and `x86_64-linux-gnu-gcc.`

4. Install python3 dependencies

   ```
   $ sudo pip3 install -r <path_to_RAAD>/requirements.txt
   ```

5. Install MongoDB, and ensure it is running. In addition, ensure that it's connectable through localhost only, and is available without credentials.

6. Install one or more of the supported applications
   a. OpenSSh
   b. Drupal
   c. MediaWiki
   d. phpMyAdmin (any version below 4.8.0 does not support authentication logging)

7. Configure Drupal/MediaWiki/phpMyAdmin to produce log files
   a. Drupal
      i. Log into the administrative user for Drupal
      ii. Click on Extend at the top (or go to <your_Drupal_URL>/admin/modules)
      iii. Enable the module Syslog (select the checkbox and click Install)
   b. MediaWiki
      i. From the `RAAD` GitHub repository, move the `RAADMW` directory to your MediaWiki extensions directory
      ii. Locate MediaWiki's configuration settings file `LocalSettings.php` and add these two lines of code at the end:

      ```
      $RAADMWfile = "$IP/extensions/RAADMW/RAADMW.log";
      require_once( "$IP/extensions/RAADMW/RAADMW.php" );
      ```

      iii. Make sure the extension has permission to write to the log file

      ```
      $ sudo chown -R www-data:www-data <log_directory>
      $ sudo chmod -R 755 <log_directory>
      ```

   c. phpMyAdmin

i. Check if your phpMyAdmin version is 4.8.0 or above
ii. $cfg['AuthLog'] should be set to "auto" by default
iii. If phpMyAdmin does not produce authentication log files, then modify the configuration settings file config.inc.php in phpMyAdmin's directory and change $cfg['AuthLog'] to "syslog"

Test Guide

Note: You must have two different machines that have different public IP addresses

1. Run the config script on the virtual machine hosting Drupal, MediaWiki, or phpMyAdmin

   ```
   $ sudo ./configure.sh
   ```

   a. It will ask you for a password, that will be your login password, with the username being root.

2. Run the start script on the same machine to start the monitoring

   ```
   $ sudo ./start.sh
   ```

   a. Alternatively, from within the folder, you can call these three commands, in separate terminals instead.

   ```
   $ python3 logs/notify.py
   ```

   ```
   $ sudo python3 backend/app.py
   ```

   ```
   $ python3 front/manage.py runsslserver 0.0.0.0:8000
   ```

3. For the admin interface, on a browser, go to https://x.x.x.x:8000/admin, where "x.x.x.x" is the IP address of the virtual machine, e.g. https://130.245.169.159:8000/admin .
4. On a separate machine, that you do not mind being banned on, try to ssh in and fail continuously.
5. You can lower the thresholds to make it faster through the web application.
6. You will know you are banned when the ssh request takes too long and times out. Additionally, if you refresh the admin page, your ip address will show up there as well.
7. Remove yourself, and try failing to log in on the Drupal, MediaWiki or phpMyAdmin page. You will also be banned.
8. Please note that if you ban yourself on the machine running the RAAD application, you will still be blocked, and you may possibly remain permanently blacklisted until someone restarts the application if it crashed and either whitelists you or let it auto time out.