Implementing, Analyzing, and Defending Against Password Spraying Attacks on HTTP-based Authentication Systems

Roo Case ’25 and Amadou Touré ‘25 | Advisor: Jeff Ondich
Department of Computer Science, Carleton College

Terminology

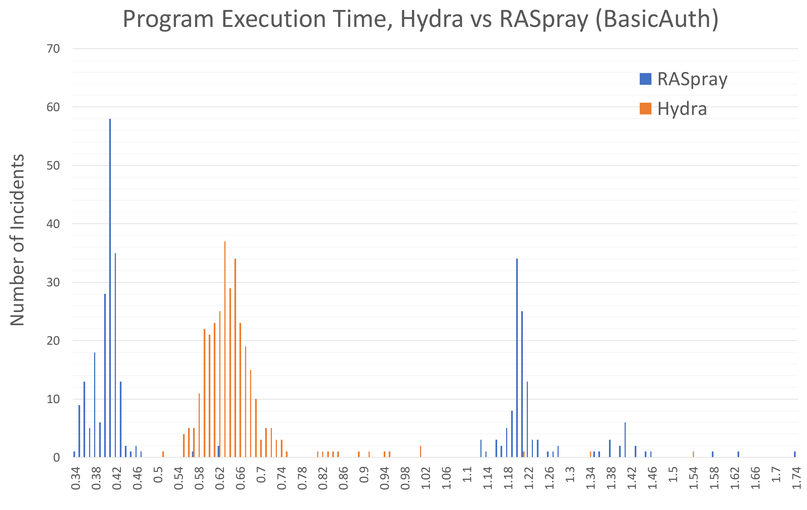
Password spraying:
“Throwing a whole bunch of usernames and passwords at a server and seeing what lets you in.”

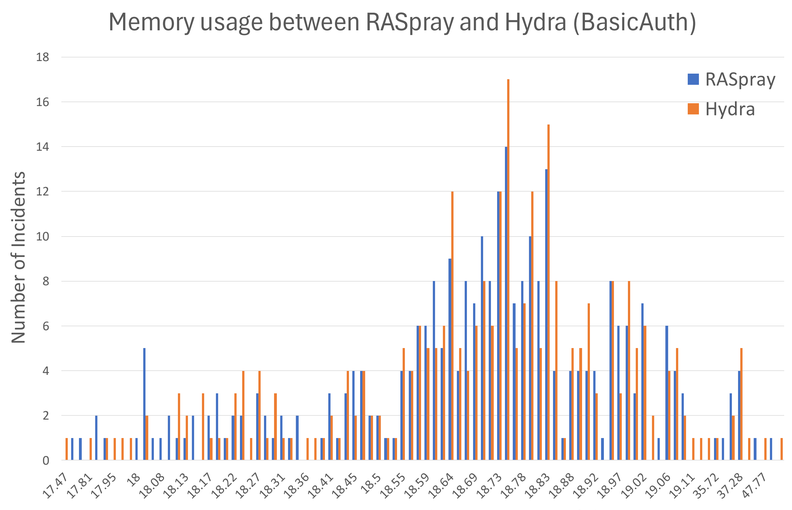
HTTP Authentication Systems:
Various ways of logging in on a website. Some are manually created by the server owner; others are built by the server itself.

Research Goals

- Assess the effectiveness of password-spraying attacks against different server configurations.
- Identify and implement evasion techniques that minimize detection risks.
- Propose practical attack and mitigation strategies to enhance the security of authentication systems.

Figures and Results





Methods

Tool - Amadou

- Dubbed **RASpray** and developed in **Python.**
- Uses “requests” library for HTTP requests and ”beautifulsoup4” for web page parsing.
- Uses a list of usernames and passwords provided by the user, applying user-defined filters like length and character types.
- Optional **brute-forcing** if initial spraying fails.
- Utilizes **threading** to considerably improve efficiency and speed up password spraying attempts across multiple requests.

```
RA SPRAY
Options:
    -h, --help                  Show this help message and exit
    -u, --users                 Specify the file containing usernames
    -p, --pass                  Specify the file containing passwords
    -i, --ip                    Specify the target IP address
    --version                   Show the tool version and exit

    Username Criteria:
    --username-min-len          Specify the minimum length of usernames
    --username-max-len          Specify the maximum length of usernames
    --username-uppercase        Require at least 1 uppercase letter in usernames
    --username-lowercase        Require at least 1 lowercase letter in usernames
    --username-numbers          Require at least 1 number in usernames
    --username-special-chars    Require at least 1 special character in usernames (e.g., !, @, #)

    Password Criteria:
    --password-min-len          Specify the minimum length of passwords
    --password-max-len          Specify the maximum length of passwords
    --password-uppercase        Require at least 1 uppercase letter in passwords
    --password-lowercase        Require at least 1 lowercase letter in passwords
    --password-numbers          Require at least 1 number in passwords
    --password-special-chars    Require at least 1 special character in passwords (e.g., !, @, #)
```

Server – Roo

- Created a server using AWS Lightsail servers.
- Runs Debian, a Linux distribution.
- Apache2, a popular open-source server tool, runs the HTTP server operations.
- Uses Nginx as a rate-limiting tool for load balance for Apache. It accepts initial connections and forwards them to Apache on an internal port.

Login Techniques:
- **BasicAuth** – The most rudimentary login. Apache handles all elements, no additional code is needed
- **PHP-based authentication –** PHP handles login functions. The login script checks for the correct username and password and assigns a password token. The subsequent webpage checks to see if a token has been assigned.
- **Attempt-based lockout**: Add-on to PHP authentication that allows 10 attempts on one username in any 10 minutes.
- **Google reCAPTCHA** – An Implementation of Google's reCAPTCHA API to limit non-human ability to use the login page.

Conclusion

Attack / Pen Testing

- Use randomized time intervals to avoid detection by rate-limiting mechanisms.
- Gather a list of valid / likely valid usernames (see references) beforehand to increase likelihood of successful spraying.
- Target weak implementations through script automation.

Defense / Server Administration

- Add small pauses in the authentication process: from 0.25 to 0.5 seconds is a reasonable delay.
- Use reCAPTCHA helps defend against many bot attacks
- Anti-DDOS tools can also help defend against high-speed spraying attacks from a single source.
- BasicAuth should not be used for sensitive data storage.

Acknowledgments

We are grateful to our advisor, Jeff Ondich, for his invaluable guidance throughout this project. A special thank you to our comps peers, friends, and family members for their support.

References & GitHub Code