WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Mitigating DDOS Attacks from IP-Rotating Bots Using Token-Based Access Control

Miss. Rashmi Ravindra Chaudhari[1], Mr. Vaibhav Tukaram Narkhede[2]

[1]Visiting Faculty, Dept. of Computer Engineering, Government College of Engineering, Jalgaon
[2]UG Student, Dept of Electronics and Telecommunication, Government College of Engineering, Jalgaon
Doi : https://doi.org/10.55248/gengpi.5.1024.2720

A B S T R A C T

IP-rotating bots, in particular, pose a significant threat to service availability as Distributed Denial of Service (DDoS) attack operations become more advanced with the Internet growing long in the tooth. With botnets and rotating IP addresses many simple blocks like IP filtering are bypassed, rate limits can be equally ineffective against these new attack patterns. In order to counteract this rising nature of threat, in this paper, an innovative approach is suggested for mitigation that uses a reverse proxy architecture and token-based access control mechanism. Because this approach forces users to create and verify time-limited, unique tokens, it complicates the process immensely for automated attackers attempting to skirt security layers and further separates legitimate traffic from malicious entities (e.g., bots).

The way the system works is based on a more than single token generation phased along with JavaScript execution then cross-checking through browser, keeping automated bots at bay. The proxy now processes every single request and filters out anything which fails the token validation process, only letting valid users through. This dynamic behavior boosts the entire security model, makes life harder for bots rotating IPs and combats DDoS attacks. In this talk, we will detail the architecture, methodology and broader significance of deploying a token-based defense in modern systems for DDoS mitigation.

Keywords: DDoS mitigation, IP rotation, advanced bots, reverse proxy, token-based access control, JavaScript execution, session tokens, automated attacks, security verification, random URL generation, bot defense, token validation, IP filtering, rate limiting, client-side security, web security

1. INTRODUCTION

The rapid growth of internet-connected devices and reliance on online services have significantly increased the risk and impact of DDoS attacks on organizations worldwide. These attacks, which flood target systems with malicious traffic, can cripple services and cause widespread disruptions. Among the various types of DDoS attacks, the rise of advanced bots using IP rotation to constantly change their addresses has made traditional mitigation techniques like IP blocking and rate limiting less effective. This evolving threat demands more sophisticated defensive strategies.

IP rotation, often supported by botnets, proxies, and VPNs, allows attackers to distribute their traffic across numerous IP addresses, making it difficult to identify and block malicious activity. Conventional network defenses like firewalls, Intrusion Prevention Systems (IPS), and rate limiting can only partially mitigate such attacks. As attackers adapt their techniques, security experts have begun to shift focus toward analyzing user behavior and introducing more adaptive defenses to combat these sophisticated threats.

This paper proposes a novel token-based system integrated with a reverse proxy to counter IP-rotating bots. Instead of relying on static IP-based defenses, the system uses a multi-step verification process where users are required to collect and submit session tokens generated by the server. These tokens are time-sensitive and involve tasks such as executing JavaScript and managing cookies, which pose significant challenges to automated bots. By introducing these additional verification layers, the system disrupts the ability of bots to complete the attack cycle, even when rotating IPs.

The proposed token-based approach focuses on behavioral validation rather than static IP detection. Legitimate users can seamlessly navigate the system, while bots face hurdles due to the time-constrained nature of the token collection process. This strategy provides an adaptive defense mechanism that better addresses the evolving tactics of DDoS attackers and enhances overall system resilience against sophisticated, automated threats like IP-rotating bots.

2. SYSTEM ARCHITECTURE

The proposed system architecture consists of several key components designed to collaboratively enhance security and mitigate DDoS attacks from IP-rotating bots. Each component plays a crucial role in ensuring that users are legitimate and not part of automated bot attacks.

2.1 Reverse Proxy

The reverse proxy is the gateway through which all incoming user traffic passes before reaching the main server. It serves as a buffer between the internet and the protected server, controlling access and managing user sessions. By acting as the first line of defense, the reverse proxy allows for pre-processing of requests before they reach the main infrastructure, which is essential in mitigating DDoS attacks.

- **Traffic Filtering**: The reverse proxy filters incoming traffic, discarding or flagging suspicious patterns before the request even reaches the token generation phase. This pre-filtering ensures that only eligible traffic proceeds through the system, reducing the load on the main server.
- **Session Management**: When users land on the proxy, it establishes a session, generating unique session tokens and serving the necessary JavaScript for token collection.
- **Scalability**: The reverse proxy can be scaled horizontally to handle large volumes of traffic, making it suitable for handling DDoS attack scenarios where thousands of requests might be initiated simultaneously.

2.2 Token Generation System

The token generation system is responsible for creating session-specific tokens that act as temporary access credentials. Upon landing on the reverse proxy, each user session is assigned a unique token, which serves multiple purposes throughout the validation process.

- **Session Token**: This token is generated using a secure algorithm that guarantees uniqueness and unpredictability. The token is stored on the server and associated with the user session. Its lifetime is limited to one minute, forcing attackers to continually regenerate tokens in real-time, complicating automated attacks.
- **Token Security**: The generation process involves cryptographic techniques to ensure that tokens cannot be easily guessed or forged by bots or attackers.
- **Time-Bound Tokens**: By limiting the validity of tokens to a short time frame, the system introduces a temporal layer of security. Attackers who cannot complete the token collection and submission process within the time frame will have their tokens invalidated, further frustrating bot-based attacks.

2.3 Random URL Generation

As part of the token-based validation process, the reverse proxy generates 3-4 random URLs that each contain a unique identifier or token. These URLs are dynamically created for each session, ensuring that they differ across user interactions.

- **Dynamic URLs**: The randomness in the URLs prevents bots from hardcoding the paths to gather tokens. Each session involves different URL endpoints, which makes it difficult for automated tools to predict or bypass the validation process.
- **Token Embedding**: Each URL contains a token embedded either within the URL itself or as part of the response (e.g., headers or hidden fields in the page source). This embedding ensures that users or their browsers must visit these URLs to retrieve the necessary tokens.
- **Obfuscation**: These URLs can be obfuscated to further hinder bot interaction. Automated bots that rely on predictable URL patterns will struggle to process obfuscated tokens and paths.

2.4 JavaScript-Based Token Collection

The reverse proxy serves a JavaScript script to the user's browser. This script is responsible for navigating to the dynamically generated URLs and gathering the tokens required for verification. This step ensures that human interaction (via the browser) is involved, making it harder for bots to mimic.

- **Browser-Based Execution**: JavaScript is executed within the user's browser, leveraging client-side resources. Many bots, especially those that operate outside of headless browsers, cannot execute JavaScript, providing a strong deterrent against non-human traffic.

- **Automatic Token Collection**: The JavaScript code visits the random URLs within a predefined time limit (usually two seconds) and retrieves the tokens. The process is fully automated from the user's perspective but relies heavily on browser functionality, making it hard for bots to replicate.
- **Token Combination**: After collecting the tokens from the random URLs, the JavaScript combines them using a cryptographic or hashing function. This final combined token is then sent to the reverse proxy for validation.

2.5 Final Token Submission

The combined token generated by the JavaScript is sent back to the reverse proxy through an AJAX request. This final step is crucial for verifying whether the user completed the process correctly within the allowed time frame.

- **Real-Time Submission**: The AJAX request is performed in real-time and must happen within the validity period of the initial session token. If the process exceeds this time limit, the token becomes invalid, and the request is rejected.
- **Cross-Verification**: The reverse proxy checks the final token against the initial session token stored on the server. This cross-verification ensures that the process was followed correctly and the final token matches the one expected from legitimate users.

2.6 Token Validation and Access Control

The reverse proxy is responsible for validating the final token and performing additional security checks before granting access to the main server. This layer of validation is designed to catch sophisticated attacks that might have bypassed earlier stages.

- **Token Expiration**: The reverse proxy checks that the token submitted by the user is still valid and has not expired. This check ensures that attackers cannot reuse tokens or submit tokens generated in previous sessions.
- **Browser and Cookie Checks**: The proxy validates session cookies and browser details, such as user agent, screen size, and browser fingerprinting information, to ensure that the session is consistent across different stages of the process. These checks help detect bots that might switch browsers or IPs during the process.
- **Timing Constraints**: The system verifies whether the user completed the token collection and submission within the 2-second window. This time constraint is a key factor in mitigating bot attacks, as automated systems may struggle to meet such precise timing requirements.

2.7 Main Server Access

Once the reverse proxy has successfully validated the final token and passed all other security checks, the user is granted access to the main server. This final step ensures that only legitimate users are allowed through, while suspicious or bot-like activity is blocked.

- **Selective Access**: Users who pass the verification process are given access to the full resources of the main server. This selective access reduces the load on the server by filtering out bot traffic at the proxy level.
- **Logging and Analytics**: The reverse proxy logs every interaction, enabling administrators to track patterns in traffic and potentially identify new attack vectors. This data can be used for further refining security measures and mitigating future attacks.

3. METHODOLOGY

The methodology of the proposed system revolves around the careful orchestration of several security layers, each designed to make it increasingly difficult for bots, particularly those utilizing IP-rotation techniques, to successfully execute DDoS attacks. By incorporating dynamic token generation, JavaScript execution, and time-sensitive verification, the system ensures that only legitimate users can proceed to the main server. The following steps break down the operation of the system.

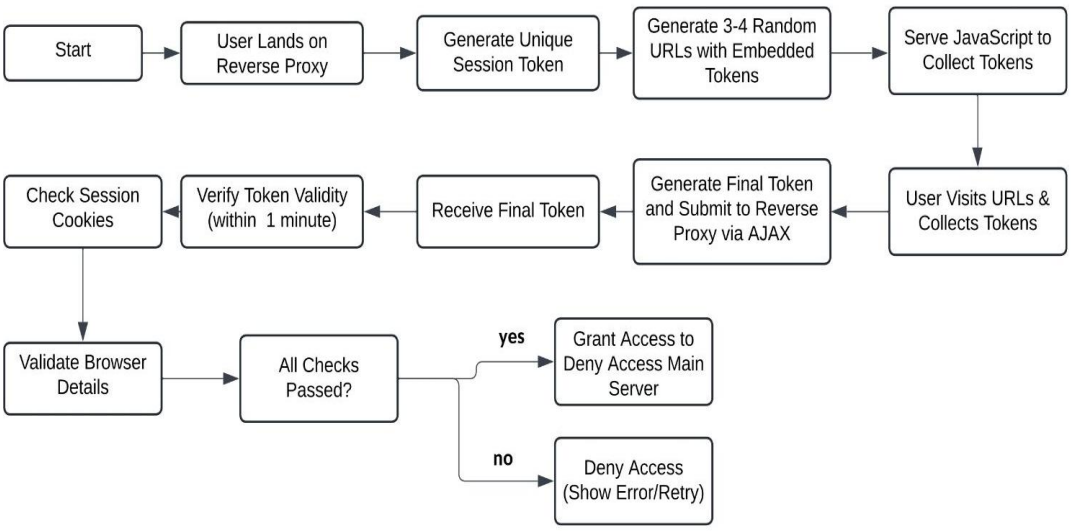

Fig.01 Flowchart for User Interaction and Token Generation and Verification Process

3.1 User Landing

When a user attempts to access the target domain, they are initially routed to the reverse proxy, which functions as the intermediary between the user and the main server. The reverse proxy serves as the primary entry point for all incoming requests and is responsible for managing the authentication and security mechanisms that follow.

As the user lands on the reverse proxy, they are served a basic landing page. This page initiates the token generation process while the user remains unaware of the underlying mechanisms. The entire process is designed to occur within a short, unnoticeable window (approximately two seconds), during which the user's browser initiates the JavaScript code that facilitates token collection.

The routing through the reverse proxy ensures that direct access to the main server is prevented, thereby filtering traffic at the earliest point. Additionally, this structure provides the flexibility to impose specific rules on incoming traffic, such as rate limiting, geolocation filtering, and IP monitoring. However, the primary defense is built around the token-based mechanism.

3.2 Session Token Creation

Upon landing on the reverse proxy, a unique session token is generated and assigned to each user. This token is created using a secure random number generator, which ensures that each token is unique, unpredictable, and resistant to potential collisions or reuse. The token generation process uses cryptographic methods to enhance security, making it extremely difficult for bots to guess or predict valid tokens.

Once the session token is generated, it is stored server-side, typically in a secure, encrypted database or memory cache, with an expiration time of one minute. This expiration time serves as a critical security measure, ensuring that even if a token is intercepted or misused, its utility is severely limited. The reverse proxy associates the token with the specific session and browser instance, preventing bots from reusing tokens across multiple IPs or sessions.

This token is a central component of the verification process. It is temporary and acts as a key for the user to navigate through the various security challenges imposed by the reverse proxy. The token's time-limited nature further complicates bot operations, as they must operate within the narrow validity window.

3.3 Random URL Generation

To further obfuscate and complicate the automated collection of tokens, the reverse proxy generates a series of random URLs upon session creation. These URLs (typically 3-4 per session) are unique to each user and are dynamically created for every request. Each URL contains its own unique token, embedded either in the URL itself (e.g., as query parameters) or within the content of the page (e.g., hidden fields, headers, or JavaScript variables).

These random URLs serve two critical purposes. First, they add another layer of security by diversifying the token collection process, requiring users (or bots) to interact with multiple endpoints rather than a single static URL. Second, they spread the token collection task across several locations, forcing bots to navigate multiple URLs and gather the necessary tokens within a constrained time window.

Because these URLs are randomly generated for each session and request, bots cannot predict or pre-fetch these URLs in advance. They must follow the token generation process dynamically, further complicating their ability to automate the attack.

3.4 JavaScript Execution

At the core of the system's security is the JavaScript code served by the reverse proxy. Once the user lands on the reverse proxy, the JavaScript code is executed within the browser. The JavaScript's primary function is to visit the random URLs generated by the proxy and collect the tokens embedded within each one.

The JavaScript automatically visits the URLs in rapid succession, collecting the tokens from each page. Bots that do not support JavaScript execution will fail at this step, as they are unable to retrieve the necessary tokens. This reliance on client-side execution introduces a significant hurdle for bots, which traditionally operate without the ability to run complex JavaScript.

Moreover, the system imposes a strict time limit on the token collection process. The JavaScript is designed to operate within a short time frame (e.g., two seconds), meaning that the entire token-gathering operation must be completed quickly. If the bot or attacker fails to collect the tokens within this time limit, the session token expires, and the process must be restarted.

By using JavaScript execution to gather the tokens, the system shifts part of the computational burden to the client, reducing the load on the server while increasing the complexity of the attack vector. This client-side process ensures that human users can navigate the challenge with ease, while bots face substantial difficulties in completing the task within the allotted time.

3.5 Token Submission

Once the JavaScript code has successfully collected all the tokens from the random URLs, it combines them to generate a final token. This final token is a unique identifier, formed by hashing or concatenating the individual tokens collected from the random URLs. The combined token adds an additional layer of complexity, as bots must correctly assemble the tokens to generate the valid final token.

After the final token is generated, the JavaScript submits it to the reverse proxy via an AJAX request. This asynchronous communication allows the browser to send the token without refreshing the page, streamlining the user experience. The token submission process is essential for verifying the user's identity and validating their session.

Because the final token is derived from multiple sources, automated bots face the challenge of not only collecting tokens from different URLs but also correctly assembling them into the final token. Any failure to collect or combine the tokens results in an invalid submission, preventing the bot from gaining access to the main server.

3.6 Verification Process

Upon receiving the final token from the client, the reverse proxy initiates the verification process. The reverse proxy checks the submitted token against the session token stored on the server. The verification process includes several key checks:

1. **Token Validity**: The reverse proxy ensures that the submitted token has not expired. Tokens are only valid for one minute after generation, meaning that attackers must act quickly to exploit them. Once the token expires, it is invalidated, and the user must restart the session.
2. **Cookie Verification**: The reverse proxy checks for the presence of session cookies, which were issued when the session was created. The session cookies ensure that the request originated from the same browser instance and user, providing additional assurance against session hijacking or token reuse.
3. **Browser Details Validation**: The reverse proxy compares the browser details provided during the session initiation (e.g., user agent, screen size, installed plugins) with the details submitted with the final token. Any significant discrepancies, such as a change in user agent or screen size, could indicate bot activity or session tampering.

These checks ensure that the user's session is authentic and that the token submission originated from the same browser instance. If all checks pass, the user is granted access to the main server.

3.7 Access Control

Once the verification process is completed, the reverse proxy grants or denies access to the main server. If the final token is valid and all verification checks pass, the reverse proxy allows the user to proceed. If any of the checks fail, the reverse proxy denies access, presenting the user with an error message or prompting them to retry.

This multi-layered verification process complicates the efforts of bots and automated systems. The reliance on unique tokens, dynamic URLs, and JavaScript execution creates a complex challenge for attackers, particularly those using IP-rotating bots. As the system forces bots to complete tasks within narrow time frames, their ability to conduct large-scale DDoS attacks is severely hindered.

4. SECURITY ANALYSIS

The proposed approach offers several security advantages that enhance the overall integrity and availability of the system against various types of threats, particularly DDoS attacks from IP-rotating bots.

4.1 Increased Complexity

One of the primary advantages of this system is the inherent complexity introduced by the token generation and validation process. Each user interaction necessitates the creation of a unique session token, which is valid only for a short time frame. This method presents significant challenges for automated bots that typically rely on predefined patterns or predictable sequences to execute their attacks.

- **Obfuscation of Intent**: The need to collect multiple tokens from various random URLs complicates the attack process. Automated bots must adapt their strategies to gather the necessary tokens, often requiring sophisticated programming to imitate legitimate user behavior.
- **Resource-Intensive**: The complexity and resource demands placed on bots deter many lower-tier attackers, as they would require more advanced capabilities and computing power to bypass the multi-token requirement. This additional resource burden can lead to higher costs for the attackers, making it less appealing to pursue such an attack.

4.2 JavaScript Dependency

The proposed system's dependency on JavaScript execution serves as a robust barrier against non-browser clients and less sophisticated bots.

- **Browser Behavior Simulation**: Most automated scraping tools and bots do not execute JavaScript, rendering them ineffective when confronted with the proposed model. The requirement for JavaScript to gather and submit tokens means that only users with compliant browsers can access the system.
- **Enhanced Detection of Malicious Activity**: By implementing JavaScript-based checks, the system can better differentiate between human users and bots. This ability to detect non-compliance enables more effective monitoring and potential blocking of suspicious activities, thus reinforcing security.

4.3 Dynamic URL Access

The utilization of dynamic, temporary URLs enhances the system's security posture against automated data collection methods.

- **Token-Embedded URLs**: Each URL provided to the user contains a unique token that is essential for access. As these tokens are generated dynamically and are specific to each session, traditional scraping tools—which often rely on static URLs—are thwarted. This reduces the likelihood of successful DDoS attacks since bots cannot predict or pre-emptively gather the necessary URLs.
- **Session Isolation**: The temporary nature of these URLs, combined with the session-specific tokens, ensures that each user's interaction is isolated from others. This session isolation prevents attackers from leveraging previously obtained tokens or URLs, requiring them to continually adapt to changing access points.
- **Layered Security Model**: By combining dynamic URL generation with token validation and JavaScript execution, the proposed approach establishes a multi-layered security model. Each layer serves as an additional obstacle for potential attackers, significantly complicating their efforts to disrupt service or gather data illicitly.

5. CONCLUSION

The integration of a token-based access control mechanism within a reverse proxy framework presents a robust solution to mitigate DDoS attacks from IP-rotating bots. This approach utilizes a unique combination of session tokens and JavaScript execution, effectively creating multiple barriers that hinder unauthorized access while facilitating legitimate user interactions.

- **Enhanced Security Features**

By leveraging session tokens that are dynamically generated and validated, the proposed system significantly enhances security. The reliance on JavaScript for critical functions means that non-compliant bots—which often lack the capability to execute JavaScript—are rendered ineffective. This not only protects against common automated threats but also deters more sophisticated attacks that attempt to exploit traditional security measures.

The multi-layered verification process embedded within the system challenges the capabilities of advanced automated attackers. Each layer of security adds complexity, making it increasingly difficult for bots to succeed. As a result, the system not only safeguards against DDoS attacks but also reinforces the overall integrity and availability of the service.

- **Future Directions**

While the proposed approach demonstrates significant promise, future research will focus on several key areas to enhance its effectiveness and applicability:

1. **Real-World Applications**: Conducting comprehensive field tests will be essential to validate the system's performance in live environments. Evaluating its effectiveness against a variety of DDoS attack vectors and bot strategies will provide insights into its robustness and reliability.
2. **Performance Under Attack Scenarios**: Testing the system under various simulated attack scenarios will help identify potential vulnerabilities and performance bottlenecks. This data will be critical for fine-tuning the system and ensuring optimal performance even during periods of high traffic.
3. **Token Generation Refinements**: Exploring advancements in token generation techniques will be vital. This could include implementing machine learning algorithms to predict and mitigate emerging threats based on historical attack patterns, thereby further enhancing the system's resilience.
4. **Adaptive Mechanisms**: As cyber threats evolve, so too must our defenses. Future iterations of this system may incorporate adaptive mechanisms that respond in real-time to attack vectors. By dynamically adjusting security parameters based on traffic analysis and threat intelligence, the system can maintain high levels of protection.
5. **User Experience Considerations**: While security is paramount, maintaining a seamless user experience is equally important. Future research will also aim to balance security measures with user convenience, ensuring that legitimate users are not unduly hindered while accessing services.

In conclusion, the proposed token-based access control mechanism within a reverse proxy framework is a promising solution that effectively addresses the challenges posed by DDoS attacks from IP-rotating bots. By continuously refining this approach and adapting to the ever-changing threat landscape, we can enhance digital security and maintain the integrity of online services in an increasingly hostile environment.

6. REFERENCES

[1] Mirkovic, J., & Reiher, P. (2004). A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2), 39-53.

[2] Bhatia, S., & Choudhary, M. (2015). Survey of DDoS Attacks and Defense Mechanisms. *International Journal of Computer Applications*, 116(14), 22-27.

[3] Zhang, Y., & Lee, R. (2018). DDoS Attack Defense Mechanisms: A Survey. *IEEE Access*, 6, 21856-21873.

[4] Chen, T., & Zhao, J. (2019). A Novel Approach to Mitigate DDoS Attacks Based on Tokenization. *Journal of Network and Computer Applications*, 129, 20-30.

[5] Shuja, J., & Ahmed, I. (2020). An Analysis of DDoS Attacks and Countermeasures in Cloud Computing. *IEEE Access*, 8, 88815-88828. doi:10.1109/ACCESS.2020.2994558.

[6] Li, W., Wu, H., & Wu, J. (2020). A Survey of Defense Mechanisms Against DDoS Attacks in Cloud Computing. *IEEE Transactions on Cloud Computing*, 8(1), 177-190. doi:10.1109/TCC.2018.2878975.

[7] Alharbi, A., & Alnofaie, A. (2021). A Survey of DDoS Attack Mitigation Techniques in Cloud Computing. *Future Generation Computer Systems*, 115, 586-600. doi:10.1016/j.future.2020.09.007.

[8] Kaur, A., & Singh, S. (2021). DDoS Attack Mitigation Techniques: A Review. *International Journal of Information Security*, 20(2), 153-179. doi:10.1007/s10207-020-00501-6.

[9] Hossain, M., & Iqbal, M. (2022). A Comprehensive Review of DDoS Attacks and Their Mitigation Techniques. *Computers & Security*, 114, 103645. doi:10.1016/j.cose.2021.103645.