Challenge Collapsar Attack

Challenge Collapsar Attack is a type of ddos attack that sends frequent HTTP requests to web servers overwhelming the resource usage to exhaustion rendering the web servers non responsive. CC attack is often incorporated with scripts through open proxy servers increasing the number of IPs to attack from and preserving anonymity of the attacker.

A simple way to detect cc attacks is to monitor individual URIs for receiving an unusual amount of high traffic from a single IP or sets of IPs and inspect parameters of the URIs for validating requests.

In a CC attack, the attacker uses many hosts over the internet and sends disguised requests to a target host. Overtime processing a request to a webpage increases its CPU usage and web servers become slower and slower to respond, which if left unattended creates a deadlock for resources and eventually the target host becomes unresponsive.

A good way to reduce cc attack is to impose a request rate limit or throttling on URIs to block IP addresses temporarily from sending requests to the server. If the server does not require to serve all traffic IP or domain whitelisting can be a good preventive measure to stop cc attack. As well as, implementing network firewall, security tools and in applicable cases using content delivery networks or CDN to stop cc attacks. Furthermore, you can also apply network filters blocking malicious sources. DNS services such as Cloudflare provide excellent tools to monitor attacks and prevent them before forwarding traffic to your webserver. Implementing challenge resolver in the WAF level is another way of protection against cc attacks.

SYN Flood attack

A SYN Flood attack is a half-open type ddos attack. A SYN Flood attack repeatedly initiates connection opening packets with the aim to overwhelm all available ports of a target machine, blocking connection requests from legitimate traffic.

A SYN flood attack can be detected by monitoring the network layer or with tcpdump for the number of connections and ports waiting for an ACK packet and setting up alerting system whenever such connections overcomes a certain threshold

SYN flood attacks work by exploiting the handshake process of a TCP connection. Under normal conditions, TCP connection exhibits three distinct processes in order to make a connection. First, the client sends a SYN packet to the server in order to initiate the connection. The server then responds to that initial packet with a SYN/ACK packet, in order to acknowledge the communication. The server then responds to that initial packet with a SYN/ACK packet, in order to acknowledge the communication. To create denial-of-service, an attacker exploits the fact that after an initial SYN packet has been received, the server will respond back with one or more SYN/ACK packets and wait for the final step in the handshake. Here's how it works:

- The attacker sends a high volume of SYN packets to the targeted server, often with spoofed IP addresses.

- The server then responds to each one of the connection requests and leaves an open port ready to receive the response.
- While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally.

In networking, when a server is leaving a connection open but the machine on the other side of the connection is not, the connection is considered half-open. In this type of DDoS attack, the targeted server is continuously leaving open connections and waiting for each connection to timeout before the ports become available again. The result is that this type of attack can be considered a "half-open attack".

SYN flood vulnerability has been known for a long time and a number of mitigation pathways have been utilized. A few approaches include:

- **Increasing Backlog queue**
  Each operating system on a targeted device has a certain number of half-open connections that it will allow. One response to high volumes of SYN packets is to increase the maximum number of possible half-open connections the operating system will allow. In order to successfully increase the maximum backlog, the system must reserve additional memory resources to deal with all the new requests. If the system does not have enough memory to be able to handle the increased backlog queue size, system performance will be negatively impacted, but that still may be better than denial-of-service.
- **Recycling the Oldest Half-Open TCP connection**
  Another mitigation strategy involves overwriting the oldest half-open connection once the backlog has been filled. This strategy requires that the legitimate connections can be fully established in less time than the backlog can be filled with malicious SYN packets. This particular defense fails when the attack volume is increased, or if the backlog size is too small to be practical.
- **SYN cookies**
  This strategy involves the creation of a cookie by the server. In order to avoid the risk of dropping connections when the backlog has been filled, the server responds to each connection request with a SYN-ACK packet but then drops the SYN request from the backlog, removing the request from memory and leaving the port open and ready to make a new connection. If the connection is a legitimate request, and a final ACK packet is sent from the client machine back to the server, the server will then reconstruct (with some limitations) the SYN backlog queue entry. While this mitigation effort does lose some information about the TCP connection, it is better than allowing denial-of-service to occur to legitimate users as a result of an attack.

SSL Attack

An SSL attack targets the SSL handshake protocol either by sending worthless data to the SSL server which will result in connection issues for legitimate users or by abusing the SSL handshake protocol itself. An TLS connection works by verifying an SSL certificate against a certificate authority (CA). SSL attack sends garbage data to validate against CA. The SSL protocol is computationally expensive and it generates extra workload on the server to process garbage data as a legitimate handshake. Firewalls don't help in this case because they are usually not capable of differentiating between valid and invalid SSL handshake packets.

It is not easy to detect ddos ssl attacks. Implementing network monitoring and intrusion detection systems (IDS) or intrusion prevention systems (IPS) to analyze incoming traffic patterns. Look for sudden spikes in SSL/TLS handshake requests or a high number of incomplete handshakes. Furthermore, use anomaly detection techniques to identify unusual or unexpected behavior in SSL/TLS handshake patterns. Set up alerts for significant deviations from normal traffic patterns. As well as, monitor SSL/TLS logs on your server to track handshake failures, error codes, and patterns of failed connections. Analyze these logs regularly for signs of excessive handshake failures.

During the SSL/TLS handshake process, the server and client exchange cryptographic keys and negotiate encryption settings. This process requires computational resources, including CPU cycles and memory. In an SSL exhaustion attack, an attacker sends a large number of incomplete or malicious handshake requests to the server. These requests consume server resources, causing CPU and memory usage to spike, potentially leading to resource exhaustion. As the server processes a high volume of incoming handshake requests, it may create a backlog of pending connections. Legitimate clients trying to establish secure connections may experience delays or timeouts due to the increased waiting time caused by the backlog. The server's increased workload and resource consumption can result in slow response times for legitimate users. This can lead to degraded user experience and frustration.

To mitigate the SSL attack following steps can be taken:

1. **Rate Limiting**: Implement rate-limiting mechanisms to restrict the number of incoming SSL/TLS handshake requests from a single IP address or subnet. This can help mitigate the impact of the attack by limiting the rate at which the attacker can generate new handshakes.
2. **IP Reputation Services**: Employ IP reputation services or threat intelligence feeds to identify and block IP addresses associated with malicious activities or known attackers.
3. **Load Balancing and Redundancy**: Distribute incoming SSL/TLS traffic across multiple servers using load balancers. This can help distribute the load and reduce the impact of an attack on a single server.
4. **SSL Offloading**: Implement SSL offloading by terminating SSL/TLS connections at a dedicated load balancer or proxy before forwarding traffic to the backend servers. This can help reduce the processing burden on the backend servers.
5. **TLS Session Resumption**: Enable TLS session resumption to allow clients to reuse established sessions, reducing the overhead of repeated handshakes.
6. **DDoS Protection Services**: Consider using specialized DDoS protection services or appliances that can detect and mitigate various types of DDoS attacks, including SSL exhaustion attacks.