Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2019, 6(11):123-129

Research Article ISSN: 2394 - 658X

SSL Offloading and Load Balancers: Enhancing Security and Reducing Server Load

Nikhil Bhagat

Principal Network Engineer
Independent Scholar, Network Engineering
nikhil.bhagat90@gmail.com

ABSTRACT

The internet and web applications are now more complicated and scalable, and an effective load balancing strategy is increasingly important. It's necessary to use a load balancer with the ability to dynamically allocate the load across different backend servers to handle the scale of traffic. Using the HTTPS protocol to protect internet traffic further increase the computing overhead for web servers. SSL/TLS offloading, which redirects the SSL/TLS termination to the load balancer rather than the web servers, is one way to overcome these issues. This strategy not only preserves the computation power of the web servers but also allows the load balancer to allocate the traffic to the server pool in a more cost-effective manner. This paper summarizes the current challenges with terminating SSL on the web servers, advantages of using SSL offloading, various scenarios where SSL offloading would be beneficial, and the design considerations for using SSL offloading mechanism.

Keywords: Load Balancing, SSL/TLS Offloading, Web Server Performance, Security, Scalability, HTTPS

INTRODUCTION

Since the internet is growing rapidly and web applications are getting increasingly more complex, the amount of traffic a web server must support has increased exponentially. Web Application Providers often run a cluster of servers behind a load balancer to manage this demand and spread out requests across those servers. However, this comes with its challenges. Hardware-based load balancers are relatively inflexible and often feature-rich but are usually very costly, difficult to scale and might not provide the flexibility needed to handle the ever changing traffic dynamics [1]. Furthermore, the web-traffic encryption with HTTPS requires massive computational resources from the web servers since it has to be handled by SSL/TLS handshaking and encryption/decryption. To address these issues an application provider may use SSL/TLS offloading where the load balancer performs the SSL/TLS termination, and not the web servers [2]. This not only lowers the computational burden on the web servers but allows the load balancer to efficiently distribute traffic to the server pool as it can make better decisions of routing based on the decrypted request data.

CURRENT CHALLENGES WITH TERMINATING SSL DIRECTLY ON THE WEBSERVERS

There are various challenges to terminating SSL (Secure Sockets Layer) directly on the web server, especially in the performance, scalability, and security driven applications. SSL termination management at the webserver level makes some deployments easy, but it imposes limitations which can hamper an efficient infrastructure.

A. Performance Overhead

This is mainly because of the performance impact SSL termination imposes on web servers. The process of decryption and encryption of SSL are extremely computationally intensive [3]. Web servers do this work at a higher cost in CPU and this can affect whether or not the server can handle other requests. For large websites, this will result in a lagging response and decreased overall performance. A dedicated hardware or a load balancer for SSL termination can take care of this overhead more efficiently, leaving the web server to serve application data.

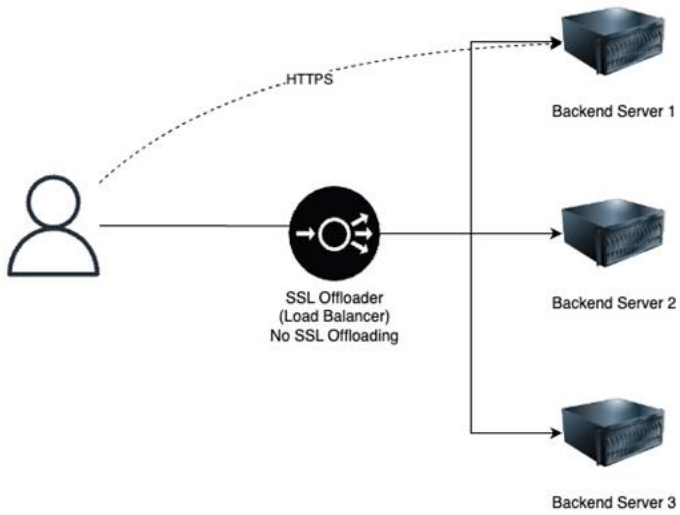

Fig 1. *SSL termination directly on web servers (No SSL offloading)*

B. Scalability concerns

As websites expand, scalability becomes a priority. Stopping SSL for all webservers means that all servers will have to maintain its SSL certificates, which might require more frequent updates and configurations [4]. In large-scale environments with many servers, it is hard and error-prone to preserve consistency from one server to another. Additionally, with the number of web servers processing traffic, completing SSL termination individually can create bottlenecks, requiring additional infrastructure investments to process the load.

C. Complexity in Certificate Management

SSL certificate management is not easy to manage especially in dynamic/distributed environments [5]. Certificates must be updated regularly and in case a certificate is expired or not renewed, it can introduce security risks or cause the service downtime [6]. Furthermore, wildcard certificates or domain certificates may need to be spread out over multiple servers, making management even harder. By implementing SSL termination on load balancers or private devices, this can be centralized to reduce this work and facilitate.

D. Security Considerations

Web servers that break SSL are also more vulnerable to attacks. The web server might be accessed via SSL keys if an attacker breaks into it and then all encrypted traffic will be compromised [7]. Offloading SSL termination to special appliances or load balancers separates the SSL layer from the web application servers, eliminating the risk of key exposure. Furthermore, SSL termination at a single place simplifies security policy and auditing tasks.

SSL/TLS OFFLOADING

SSL offloading, or SSL termination is an approach whereby the SSL encryption and decryption tasks are performed on an external device, like a load balancer, instead of the web server. This practice aims to increase the speed of web servers by sending high-priority SSL calculations to lower-powered devices. This is a very detailed demonstration of the SSL offloading algorithm:

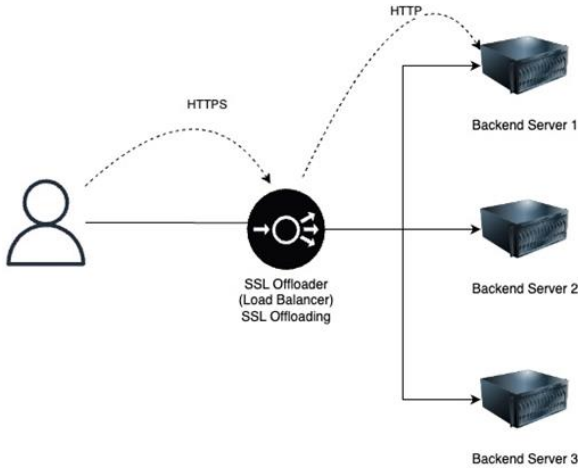

Fig 2. *SSL termination on the Load Balancer (with SSL Offloading)*

A. End user Initiates SSL Handshake

The process starts when an end user or a client usually via a web browser opens a secure HTTPS connection to a website [8]. The client makes a request to the server, asking for an SSL/TLS session. This message contains information about the capabilities of the client (such as what cipher suites, SSL/TLS versions, and compression are available).

B. SSL Offload Device Receives the Request

This request is intercepted instead of being sent directly to the web server, by the SSL offloading device, usually a load balancer or specialized SSL terminator [9]. This device performs the SSL handshake taking the load off the web server.

C. Server Certificate Presentation

The SSL offloading device presents the website's SSL certificate to the client. This certificate contains the public key and other website identification information [10]. Client verifies the authenticity of certificate by confirming the chain of trust (signed by a recognized Certificate Authority) and whether the certificate has not been expired or revoked.

D. Cipher Suite Negotiation

Both the client and SSL offloading device negotiate over which cipher suite to use for the session. A cipher suite is a sequence of encryption keys and hash functions, which specifies how information will be protected in transmission [11]. The best interoperable encryption algorithms are negotiated between the parties.

E. Key Exchange

During this step, the SSL offload device and the client perform key exchange, which creates a secret shared between them for the rest of the communication session. Depending on the cipher suite, this can involve either RSA (Rivest–Shamir–Adleman) or Diffie-Hellman key exchanges [12]. The client can exchange a "pre-master secret" with the public key of the SSL offload device.

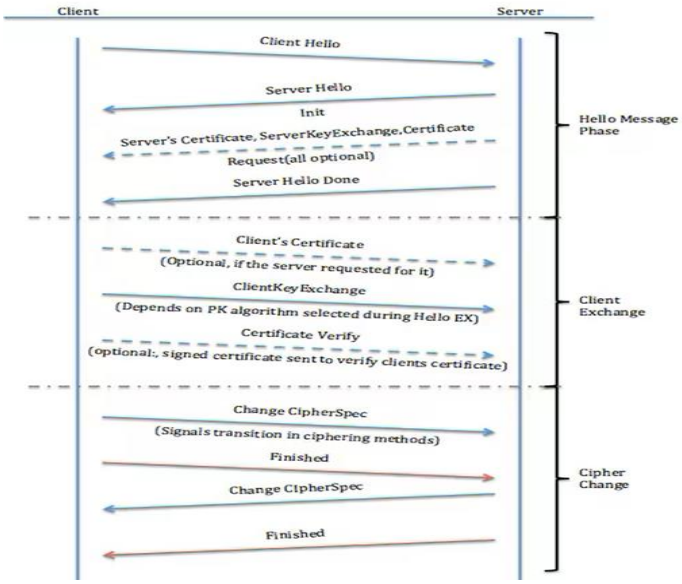

Fig. 3 *SSL process between the client and the server [13]*

F. Session Keys are created

Based on the data transferred, both the client and SSL offload device independently create session keys. The keys are symmetric, that is to say, they can be encrypted and decrypted as data passes between the client and the server. Symmetric encryption techniques such as AES (Advanced Encryption Standard) are less costly computationally than asymmetric encryption [14].

G. SSL Handshake Completion

After session keys are generated, SSL handshake completes, and client and SSL offload device are prepared to transmit data in a secure way [15]. Then, the SSL offload device issues a last acknowledgment to the client and the client replies indicating that the handshake is over.

H. Secure Communication Established

After a handshake, the client starts making HTTP requests with the session key. This is done by SSL offloading device which decrypts incoming data and sends the plain text to web server [5]. In a similar way, it encrypts the responses from the web server and returns them to the client. This makes the data safe when transmitting despite the fact that web server is not doing the encryption directly.

I. Session Termination

The SSL session continues to exist until the client or the server terminate the connection. After session closes, session keys are deleted. The client will reopen a connection later, unless session resumption mechanism such as SSL session caching or session tickets is applied.

BENEFITS OF USING SSL OFFLOADING

SSL offload on load balancers offers many key advantages, particularly in performance, security, and scalability of the web applications. By moving SSL encryption and decryption to a dedicated device (e.g., load balancer), businesses can streamline infrastructure and enhance the user experience. Here are the most important advantages of SSL offloading to load balancers:

A. Enhanced Web Server Performance

The most noticeable benefit of SSL offloading is a faster web server. SSL encryption and decryption use large amounts of CPU and memory. When a web server processes SSL termination, it needs to spend these resources on what they do best, serving content [15]. SSL offloads that load onto the load balancer and frees up web servers to deal with processing and the delivery of application data. This leads to quicker response times, lower latency and a general performance increase in the application when it comes to heavy traffic.

B. Simplified Certificate Management

Managing SSL certificates on multiple web servers can be cumbersome and time-consuming, especially on large-scale networks [2]. Load balancers centralize SSL termination, so SSL certificates only need to be held in one place for easier updates, renewals, and fault detection. This means no installing and setting up certificates for each web server, a less chance of configuration errors, and consistent encryption across all connections.

C. Improved Scalability

As web applications become popular, it would increase the overall traffic on the web servers. Web application provider will have to use scale-up strategies involve spawning additional servers to cope with this load. If SSL termination is done on every server individually, scaling will become complicated [16]. SSL offloaded load balancers make it easy to scale horizontally since they distribute traffic across different backend servers without the need to have those hosts terminate SSL. The result is more efficient resource consumption and easy scaling enabling the provider to add capacity without overloading individual servers.

D. Increased Security

Load balancers contain a dedicated hardware or a software designed to process SSL traffic high level security features like HSMs (hardware security modules) for key management. Additionally, by centralizing SSL termination, it's easier to maintain security policies and better audit and monitor SSL traffic to avoid misconfigurations or security attacks.

E. Advanced Load Balancing options

SSL offloading allows load balancers to add additional traffic management techniques like Layer 7 inspection, URL based routing, content switching and so on. Since the SSL layer is done on the load balancer, it will be able to decrypt and scan traffic, thus making better routing decisions based on the content of the request [17].

RECOMMENDED SCENARIOS FOR SSL OFFLOADING

SSL offloading is performance optimizing strategy used in the telecommunications industry by organizations that require a secure communication, improved security and enhanced scalability. The following use cases illustrate various kinds of businesses who can utilize SSL offloading.

A. High-Traffic E-Commerce Platform

Scenario: A large online retail company sells millions of products per day especially on Black Friday or holidays. They use multiple web servers that support both HTTP traffic and HTTPS traffic for encrypted transactions (credit card details, etc.).

Challenge: SSL encryption and decryption is very computationally demanding and it reduces the capacity of web server to process the request. The administration of SSL certificates on hundreds of servers imposes an administrative overhead as well.

Solution: The company can use terminate SSL on load balancers instead of the web servers. The load balancers take care of the SSL handshake and encrypt the incoming packets before redirecting it to the backend servers [1]. That way, the web servers can focus on application logic processing, better response times, and scale for heavy traffic. Moreover, centralizing SSL certificate management to load balancers simplifies the task of certificate maintenance across multiple servers.

B. Financial Services and Online Banking

Scenario: A financial company with online banking services requires a high-security encryption to ensure the customer data (such as account details, transaction data) is secured and encrypted.

Challenge: The company needs to use end-to-end encryption for all user flows and meet strict regulations like PCI-DSS. However, the computational cost of running SSL encryption locally on the application servers adds latency and delays transaction processing in real time.

Solution: In this situation, the company can terminate SSL at the load balancer, which would perform the encryption and decryption. This frees up the backend servers from processing SSL and instead focus on making the transactions fast and safe. The load balancers are also equipped with hardware security modules (HSMs) to protect encryption keys and stay in line with industry guidelines [11].

C. Healthcare Systems and Patient Portals

Scenario: A healthcare provider has a patient portal where users can view medical records, book appointments, and talk to providers. As healthcare information is very sensitive, every communications in the business should be encrypted.

Challenge: Patient data requests requires encryption that can strain the web application servers, especially in high-usage times where records need to be accessed by many users at the same time. Additionally, the healthcare provider must also meet regulations such as HIPAA, which requires encryption and high level of security measures.

Solution: By applying SSL offloading, the healthcare provider moves the SSL termination away from the application servers and on to an appliance. This allocates computational power for the backend servers to respond to request for patient data faster [18]. It also facilitates a centralized certificate management as per the regulatory compliance requirements, making auditing and security inspections easier.

D. Content Delivery Networks (CDNs)

Scenario: An enterprise-class content delivery network (CDN) provides websites that need fast and secure delivery of videos, images, and static pages. Most of these websites also need secure data exchange through HTTPS when it comes to subscriptions or user accounts.

Challenge: Maintaining SSL certificates and deleting SSL from each edge server would be inefficient and time-consuming especially for a large CDN provider that has a large distributed infrastructure.

Solution: SSL offloading is done at CDN edge servers, where SSL termination is done by load balancers. It does this by offloading SSL on edge servers, which speeds up content loading and secure connections. Secured SSL certificate management at the edge can also enable deploying updates across the CDN without impacting service delivery [19].

E. Cloud-Based SaaS Providers

Scenario: A SaaS provider provides cloud-based software to a series of customers, most of whom require secure HTTPS connectivity to safeguard their data.

Challenge: All end clients require an encrypted connection As the SaaS company grows, the more secured connections they would receive.

Solution: The SaaS business implements SSL offloading in the load balancers, which enables them to close thousands of HTTPS sessions without stressing the backend infrastructure. This helps reduce latency, enhance user experience, and leaves the application servers free to focus on providing the software core functions.

DESIGN CONSIDERATIONS

When a system needs to utilize SSL offloading, there are some design considerations to take into account to ensure optimized performance, security, and scalability. Here are the main design considerations around SSL offloading:

A. Performance and Scalability

Capacity Planning: SSL encryption and decryption consumes a good amount of CPU resources. An application provider needs to make sure that the load balancer or SSL terminator being used is powerful enough to process the desired number of SSL handshakes and concurrent connections without any bottlenecks [5].

Hardware Acceleration: Use hardware SSL offloading devices that leverage hardware security processors referred to as hardware security modules (HSMs) [11]. These can handle massive amounts of SSL connections while consuming less CPU and providing increased throughput.

Connection Pooling: In order to further minimize the burden on backend servers, enable connection reuse or pooling from the load balancer to the application servers. This reduces the overhead in setting up new connections on every request.

Load Balancing Algorithm: Select a suitable load balancer algorithm (e.g., round-robin, least connections, etc.) to balance traffic across the backend servers.

B. Security

Key Management: Secure key management is important when terminating SSL. Maintain SSL certificates and private keys on a tamper-proof hardware module (HSM) as much as possible. Maintain key management, for example rotating keys and keeping unauthorized users away.

SSL Cipher Suite Configuration: Select secure and newest cipher suites that comply with current security standards. Disable fragile protocols (SSLv3, older TLS versions) and ciphers to avoid POODLE, BEAST or Heartbleed [21].

Forward Secrecy: It is recommended to enable forward secrecy in the cipher suite settings to ensure security of prior sessions if a private key is compromised.
Certificate Management: Design effective SSL certificate management, renewal and update automation. Plan on using certificate management systems or ACME standards to facilitate this. If necessary, use wildcard certificates or SAN certificates to save administrative burden.

C. High Availability and Redundancy

Multiple redundant SSL Offloaders: Set up multiple SSL offloaders (load balancers) in an active-active or active-passive architecture [22]. This reduces single points of failure and makes the infrastructure more reliable.
Failover Mechanisms: Create effective failover mechanisms to migrate to back-up SSL offloaders (load balancers) in case of a device/service failure. Provide the ability to transition between devices without disrupting user experiences or creating a new SSL handshake.
Session Persistence (Sticky Sessions): For stateful apps, session persistence must be enabled on the load balancer. This means the user always communicates with the same backend server for the rest of the session.

D. Latency and Geographic Considerations

Distributed SSL Offloading: If a provider has an application with worldwide users, it might be a good idea to split SSL offloaders across multiple locations to minimize latency. The closer a provider can terminate the SSL from the user, the faster the response time and user experience will be.
TCP and SSL Optimization: Speed up the transport layer using options such as TCP fast open, session resumption, or SSL session caching to minimize the overhead of SSL handshakes [11].

E. Integration with Backend Systems

Clear-Text Communication Between Offload Device and Backend: SSL offloading locks SSL traffic at the load balancer and transfers decrypted traffic to the backend servers [11]. It is recommended to secure internal connection between the offloading device and backend servers via VPNs, or isolated network links to stop any traffic being intercepted.
Re-encrypting Traffic: In some cases, especially when it comes to heavily regulated environments (e.g., banking, healthcare), the load balancer and backend systems may need to be re-encrypted for further protection. In this case, the provider of the application might want to apply TLS between the load balancer and the backend servers.
Logging and Monitoring: Logging & Monitoring software needs to be integrated with the load balancer to monitor SSL traffic, expired certificates, and other security risks. Decrypted traffic logs can be used for auditing but should be done in a secure manner.

F. Compliance

Regulatory Compliance: The SSL offloading architecture must meet all rules and industry best practices like PCI-DSS for financial data or HIPAA for health data [5]. These can include high-level specifications on key security, encryption strength, and auditing.
Auditing and Monitoring: Create audit tools that record the SSL traffic, use of certificates, encryption algorithms. This is a way to follow rules and detect security loopholes before it gets missed.

G. Cost Considerations

Operational Costs: Consider and evaluate the costs of using SSL offload devices or services, like hardware, software, licensing and certificate management costs. HSM-equipped hardware SSL offloaders are a bit more expensive but provide better security and performance.

CONCLUSION

SSL offloading is a popular method used in today's web applications for increased performance, security and scalability. This paper explored the key design aspects of SSL offloading and shared some real-world advice to get started with a solid foundation for SSL offloading. While planning for performance, security, availability, latency, federation with backends, compliance, and cost, there are some key things to consider. Utilizing these best practices, companies can develop a customized SSL offloading solution for their own specific needs, which offers a smooth, safe and scalable user experience.

REFERENCES

[1]. "Load Balancing for High Performance & Availability in the Cloud," 2013.
[2]. C. Coarfa, P. Druschel, and D. S. Wallach, "Performance analysis of TLS Web servers," in Proceedings of the 11th Network and Distributed System Security Symposium (NDSS), Feb. 2002.
[3]. "Analyzing the Energy Consumption of Security Protocols," 2010.
[4]. J. Kim, G. S. Choi, and C. R. Das, "A Load Balancing Scheme for Cluster-based Secure Network Servers," in Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS), Apr. 2000.

[5]. L. Zhao, R. K. Iyer, S. Makineni, and L. N. Bhuyan, "Anatomy and Performance of SSL Processing," in Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Mar. 2005.
[6]. V. Ungureanu, "Formal support for certificate management policies," 2012.
[7]. "On the Security of the TLS Protocol," RFC. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6101, 2015.
[8]. G. Apostolopoulos, V. Peris, and D. Saha, "Transport layer security: how much does it really cost?," in Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Mar. 1999.
[9]. M. McDowell, "SSL/TLS Interception Proxies and Transitive Trust," 2014.
[10]. "Secure Socket Layer," 1999.
[11]. W. Chou, "Inside SSL: accelerating secure transactions," in IEEE Internet Computing, vol. 4, no. 4, pp. 47-52, Jul. 2000.
[12]. "Authenticated Key Exchange (in TLS)," 2006.
[13]. "Cisco Secure Socket Layer (SSL) Performance Overview," Cisco. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-socket-layer-ssl/116181-technote-product-00.html.
[14]. A. N. I. T. N. Institute, "Federal Information Processing Standard (FIPS) 197, Advanced Encryption Standard (AES)," 2001.
[15]. C. Coarfa, P. Druschel, and D. S. Wallach, "Performance analysis of TLS Web servers," in Proceedings of the 11th Network and Distributed System Security Symposium (NDSS), Feb. 2002.
[16]. J. Kim, G. S. Choi, and C. R. Das, "Improving Performance of Cluster-Based Secure Application Servers with User-Level Communication," in Proceedings of the IEEE International Conference on Cluster Computing, Sep. 2002.
[17]. "Web Applications: Securing high availability Web applications," 2007.
[18]. V. Gupta and S. Gupta, "Securing the wireless internet," in IEEE Communications Magazine, vol. 39, no. 12, pp. 68-74, Dec. 2001.
[19]. H. Chang, A. Hari, S. Mukherjee, and T. V. Lakshman, "Bringing the cloud to the edge," in Proceedings of the 37th IEEE International Conference on Computer Communications (INFOCOM), Apr. 2018.
[20]. W. Tsai, Y. Huang, X. Bai, and J. Gao, "Scalable Architectures for SaaS," in Proceedings of the 2008 IEEE International Conference on e-Business Engineering, Oct. 2008.
[21]. "This POODLE Bites: Exploiting The SSL 3.0 Fallback Security Advisory," 2014.
[22]. S. E. Frankel, P. Hoffman, A. Orebaugh, and R. Park, "Guide to SSL VPNs," in NIST Special Publication 800-113, Jul. 2008.