Reverse Proxy (NGINX)

Ein Reverse Proxy dient dazu einkommende Anfragen zu verwalten und diese an weitere Applikationen weiter zu leiten (Aivaliotis, 2013). Diese Anfragen folgen meist dem HTTP- bzw. HTTPS-Protokoll, aber sind nicht auf diese beschränkt. Im Allgemeinen wird zwischen 2 verschiedenen Reverse Proxy Arten unterschieden. Zum einen dem Protection Reverse Proxy und zum anderen dem Integration Reverse Proxy. Ein Protection Reverse Proxy ermöglicht es Anfragen gezielt zu filtern und nur gewünschte Anfragen durchzulassen. Ein Integration Reverse Proxy ermöglicht es Anfragen auf viele unterschiedliche Systeme zu verteilen. (Sommerlad, 2003) Im Folgenden wird nur der Intergration Reverse Proxy weiter beachtet, da ein Protection Reverse Proxy nicht zu Einsatz kam.

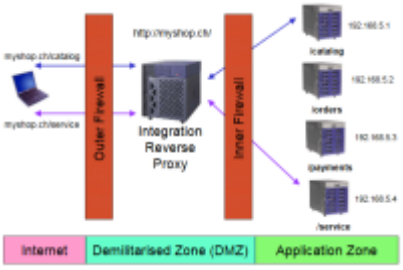
Probleme bei der Bereitstellung von Webapplikationen

Eine Webapplikation besteht meistens nicht nur aus einem Webserver beziehungsweise Webdienst, welcher alle Aufgaben übernimmt, sondern aus mehreren. Als Entwickler entstehen schwerwiegende Probleme, wenn versucht wird eine Applikation auf lediglich einem Server zu betreiben. Nach Sommerlad gibt es sieben Punkte die es zu beachten gilt:

- Man kann eine Webapplikation nicht auf einem Server implementieren, da dies die Komplexität erhöht, die Performance oder Wiederverwendbarkeit verschlechtert.
- Man möchte die allgemeine Netzwerkstruktur der Webapplikation verstecken, um bei Änderung jener beispielsweise Lesezeichen der Nutzer nicht zu zerstören.
- Das Backend der Webapplikation muss auch funktionieren, wenn sich die Netzwerkstruktur ändert. Wenn eine Applikation des Backends auf eine andere Maschine verschoben wird, darf das die Anderen Applikationen nicht beeinflussen.
- Man muss die Möglichkeit haben Teile der Netzwerkstruktur zu ändern, ohne andere Teile zu beeinflussen.
- Man muss die Möglichkeit haben neue Elemente und somit Funktionalität zur Webapplikation hinzufügen zu können.
- Man muss die Möglichkeit haben Anfragen zwischen mehreren Hosts aufteilen zu können.
- Das ganze System sollte lediglich über ein einziges SSL Zertifikat verfügen, da SSL Zertifikate einen hohen Preis haben und deren Erneurung koordiniert werden muss.

Die von Sommerlad genannten Punkte sind von hoher Relevanz. Jedoch ist ein Punkt mittlerweile nicht mehr aktuell. SSL-Zertifikate können mittlerweile auch kostenlos bezogen und automatisch erneuert werden. Siehe Verschlüsselung mit TLS. Ein weiterer Punkt mit hoher Relevanz ist die Ausfallsicherheit. Sollte eine Webapplikation lediglich auf einem Server betrieben werden und dieser fällt aus, dann ist die komplette Applikation folglich nicht mehr erreichbar.

Lösung für die Probleme



Nach Sommerlad ist die Lösung der oben genannten Probleme ein Intergration Reverse Proxy. In DARSTELLUNG SOMMERLAD wird eine beispielhafte Implementation eines solchen gezeigt. Ein Benutzer der Webapplikation ruft diese über das Internet auf. Dabei wird aus der URL klar, welchen bestimmten Bereich der Webapplikation der Benutzer aufrufen möchte, beispielsweise der Katalog unter “/catalog”. Die Anfrage geht dann jedoch nicht direkt an den Server, welcher für den Katalog zuständig ist, sondern zunächst einmal an den Reverse Proxy. Dieser befindet sich in einer Demiliarisierten Zone (DMZ). Diese DMZ ist durch zwei Firewalls von den anderen System und dem Internet getrennt.

Eine Firewall kann sowohl physisch in Form von Hardware, als auch digital in Form von Software vorliegen. Eine Kombination von beiden ist auch möglich. Die Hauptaufgabe einer Firewall ist es das dahinter liegende Netzwerk vor unauthorisierten Zugriffen zu schützen. Eine demiliarisierten Zone liegt weder in noch außerhalb einer solchen Firewall, wobei die Server in der Zone sowohl von internen, als auch externen Benutzern erreicht werden können. Sicherheitsrichtlinen verhindern dabei aber, dass externe Geräte und Nutzer direkt an die internen Server beziehungsweise Nutzer verbinden können. (Rababah, Zhou, Bader, 2018)

Wenn die Anfrage dann den Reverse Proxy erreicht, leitet dieser die Anfrage an den jeweilgen Backend-Server weiter. Antworten von einem der Backend-Server werden dann auch über den Reverse Proxy zurück an den Benutzer geleitet.

Vorteile der Lösung

Die wichtigsten Vorteile der Lösung mit einem Intergration Reverse Proxy, welche relevant im Rahmen dieser Arbeit sind, werden im folgenden erläutert.

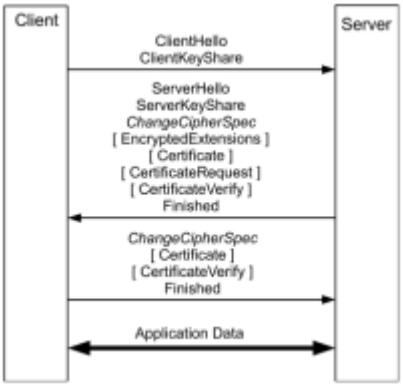
Nur ein Hostname

Einer der wichtigsten Vorteile laut Sommerlad ist es, dass es lediglich einen bekannten Host gibt, und zwar den Reverse Proxy. Für den Nutzer muss lediglich die Adresse von diesem bekannt sein, um eine Webapplikation zu erreichen, welche auf mehreren Servern betrieben wird. Dass lediglich ein Hostname benötigt wird bringt noch weitere Vorteile mit sich. Auch ist die allgemeine Netzwerkstruktur der Webapplikation versteckt. Dies macht es für potenzielle Angreifer schwerer Sicherheitslücken zu finden. Weiterhin ist es so möglich einen Teil der Applikation auf einen anderen Server zu verschieben, ohne dass dies für einen Nutzer auffallen würde. (Sommerlad, 2003)

Verschlüsselung mit TLS

TLS (kurz für Transport Layer Security) ist der Nachfolger des SSL-Protokolls. Die aktuelleste Version ist Version 1.3. Sowohl SSL als auch TLS sind Teil des HTTPS-Protokolls (Hypertext Transfer Protocol Secure). SSL kommt dabei heutzutage nicht mehr zum Einsatz, da es auf Grund von schweren

Sicherheitslücken (beispielsweise POODLE Attacke) seit 2015 von der “Internet Engineering Task Force” als veraltet eingestuft wird (Oppliger, 2016, S.15). TLS ist ein essentieller Bestandteil der verschlüsselten Verbindung zwischen Nutzer und Webserver. Im normalen Fall läuft ein verschlüsselter Verbindungsaufbau mit gerade einmal drei Anfragen ab.



Der Client (Nutzer) stellt zunächst eine Anfrage an den Server, sich zu authentifizieren. Der Server antwortet darauf hin mit den Verbindungsinformationen, einer Chiffre zur Verschlüsselung und einem Zertifikat. Der Client überprüft dann das Zertifikat. Ist das Zertifikat rechtens schickt er eine weitere Bestätigung an den Server zurück. Daraufhin können Daten mit der Chiffre verschlüsselt zwischen dem Server und dem Client ausgetauscht werden. Dieser Ablauf wird als TLS-Handshake bezeichnet. (Oppliger, 2016, S.139ff) Die Zertifikate, welche benötigt werden für TLS und somit https, können bei einer Zertifizierungsstelle wie “Let´s Encrypt” beantragt werden. Dies funktioniert nach Einrichtung auf einem Server voll automatisch, transparent und kostenlos. (Let´s Encrypt, o.J.) Besteht eine verschlüsselte Verbindung ist dies für den Nutzer anhand der URL erkennbar. Diese beginnt dann mit “https:” *statt mit “http:*”. Würden nun alle Backend-Server einzeln an das Internet angebunden werden, so müsste auch jeder sein eigenes Zertifikat besitzen. Dann müsste der Client auch mit jedem der Server einen TLS-Handshake durchführen, was zusätzlich Zeit benötigen würde. Ein Integration Reverse Proxy ermöglicht es nun alle Verbindungen mit dahinterliegenden Backend-Server mit leiglich einem Zertifikat zu verschlüsseln. Ein weiterer Vorteil ist, dass, sofern sowohl Reverse Proxy, als auch Backend-Server in einem Netzwerk liegen und nicht über das Internet verbunden sind, die Verbindungen zwischen Reverse Proxy und den Backend-Servern nicht verschlüsselt sein müssen. Dies ist eine weitere Zeitersparnis.

Load Balancing

Auch ermöglicht ein Reverse Proxy die Einrichtung mit sogenanntem Load Balancing. Dabei werden Anfragen auf unterschiedliche Backend-Sever verteilt, welche aber die selbe Applikation betreiben. Dies ermöglicht besonders bei System mit vielen Nutzern, die Last zu verteilen und dem Nutzer so eine bessere Nutzung zu ermöglichen. Es gibt mehrere Verfahren zur Verteilung von Anfragen. Eine einfache Lösung ist beispielsweise die “Round Robin”-Methode. Dabei wird lediglich eine Warteschlange an freien Backend-Servern abgearbeitet. Effezientere Methoden beinhaltet auch das Einbeziehen von Statistiken wie Antwortzeit und momentane Last von Backend-Servern, zur Verteilung von Anfragen auf die jeweiligen Backend-Server. Probleme entstehen lediglich, wenn die Webapplikation mit Nutzer-Sessions arbeitet. Diese müssen dann auf jedem Backend-Server mit der selben Applikation verfügbar sein. Dies fällt dann aber nicht mehr in den Zuständigkeitsbereich von Reverse Proxys. (Aivaliotis, 2013)

From:
https://wiki.eolab.de/ - **HSRW EOLab Wiki**

Permanent link:
https://wiki.eolab.de/doku.php?id=user:jan001:ba:reverseproxy

Last update: **2021/08/24 17:35**