## 1.Introdução

HTTPS é simplesmente o protocolo HTTP padrão coberto com uma camada generosa de **criptografia SSL/TLS**. A menos que algo dê terrivelmente errado, isso impede que pessoas visualizem ou modifiquem as solicitações que compõem sua experiência de navegação.

Embora o pequeno cadeado verde e as letras *https* na sua barra de endereço não signifique que ainda haja corda o suficiente para você e o site que você está visualizando se pendurarem em outro lugar, eles pelo menos ajudam você a se comunicar com segurança enquanto você faz isso.

Servidores e clientes ainda falam exatamente o mesmo HTTP entre si, mas por meio de uma conexão SSL segura que criptografa e descriptografa suas solicitações e respostas. A camada SSL tem 2 propósitos principais: verificar se você está falando diretamente com o servidor com o qual você pensa que está falando e garantir que apenas o servidor possa ler o que você envia e somente você possa ler o que ele envia de volta.

A parte realmente inteligente é que qualquer um pode interceptar cada uma das mensagens que você troca com um servidor, incluindo aquelas em que você concorda com a chave e a estratégia de criptografia a serem usadas, e ainda assim não consegue ler nenhuma das mensagens reais que você envia.

Uma conexão SSL entre um cliente e servidor é estabelecida por um *handshake*, cujos objetivos são:

- Para garantir ao cliente que ele está se comunicando com o servidor certo
- Para que as partes tenham acordado um “conjunto de cifras”, que inclua qual algoritmo de criptografia usarão para trocar dados
- Para que as partes tenham concordado com quaisquer chaves necessárias para este algoritmo

Assim que a conexão for estabelecida, ambas as partes podem usar o algoritmo e as chaves estabelecidas para enviar mensagens entre si com segurança.

O *handshake* funciona da seguinte maneira descrita abaixo.

- ***Hello:*** O *handshake* começa com o cliente enviando uma mensagem **ClientHello**. Contém todas as informações que o servidor precisa para se conectar ao cliente via SSL, incluindo os vários conjuntos de criptografia e a versão máxima de SSL que ele suporta. O servidor responde com um **ServerHello**, que contém informações semelhantes exigidas pelo cliente, incluindo uma decisão baseada nas preferências do cliente sobre qual conjunto de cifras e versão do SSL será usado

- ***Certificate exchange:*** Agora que o contato foi estabelecido, o servidor deve provar sua identidade aos clientes. Isso é conseguido usando seu certificado SSL, que contém vários dados, incluindo o nome do proprietário, a propriedade (domínio) à qual está anexado, a chave pública do certificado, a assinatura digital e informações sobre as datas de validade do certificado. O cliente verifica se confia implicitamente no certificado ou se ele é verificado e confiável por uma das várias autoridades de certificação (CAs) nas quais também confia implicitamente.
- ***Key exchange:*** A criptografia dos dados reais das mensagens trocadas entre o cliente e o servidor será feita pelo meio de um algoritmo simétrico, cuja natureza exata já foi acordada durante a fase **Hello**. Um algoritmo simétrico usa uma única chave para criptografia e descriptografia, em contraste com algoritmos assimétricos, que requerem uma chave pública e outra privada.

O cliente gera uma chave aleatória a ser usada para o algoritmo simétrico principal. Ele o criptografa usando um algoritmo também acordado durante a fase **Hello** e a chave pública do servidor. Ele envia essa chave criptografada para o servidor, onde é descriptografada usando a chave privada do servidor, e as partes interessantes do *handshake* são concluídas.

As partes estão suficientemente satisfeitas por estarem conversando com a pessoa certa e concordam secretamente com uma chave para criptografar simetricamente os dados que estão prestes a enviar um ao outro. Solicitações e respostas HTTP agora podem ser enviadas formando uma mensagem de texto simples e depois criptografando-a e enviando-a. A outra parte é a única que sabe como descriptografar esta mensagem e, portanto, os atacantes MitM não conseguem ler ou modificar quaisquer solicitações que possam interceptar.

**2.Certificados**

No seu nível mais básico, um certificado SSL é simplesmente um arquivo de texto e qualquer pessoa com um editor de texto pode criar um. A mágica que impede farsantes de criarem certificados SSL da Google, por exemplo, está na assinatura digital, que permite a uma parte verificar se o papel de outra parte é realmente legítimo.

Existem duas razões sensatas pelas quais você pode confiar em um certificado: Se estiver em uma lista de certificados nos quais você confia implicitamente e se for capaz de provar que é confiável para o controlador de um dos certificados da lista acima.

O primeiro critério é fácil de verificar. Seu navegador possui uma lista pré-instalada de certificados SSL confiáveis da CAs que você pode visualizar, adicionar e remover. Esses certificados são controlados por um grupo centralizado em

organizações extremamente seguras e confiáveis como a **Symantec**, **Comodo** e **GoDaddy**. Se um servidor apresentar um certificado dessa lista, você saberá que pode confiar nele.

O segundo critério é muito mais difícil. É fácil para um servidor dizer "sim, meu nome é Microsoft, você confia na Symantec e eles confiam totalmente em mim, então tudo bem". Um cliente um tanto inteligente pode então perguntar à Symantec, "*tenho aqui uma Microsoft que diz que você confia neles, isso é verdade?*" Mas mesmo que a Symantec diga "*sim, nós os conhecemos, a Microsoft é legítima*", você ainda não sabe se o servidor que afirma ser da Microsoft é na verdade a Microsoft ou algo muito pior. É aqui que entram as assinaturas digitais.

**3.Assinaturas digitais**

Como já observado, os certificados SSL possuem um par de chaves pública e privada associados. A chave pública é distribuída como parte do certificado, e a chave privada é mantida protegida de forma incrivelmente segura. Este par de chaves assimétricas é usado no *handshake* SSL para trocar uma chave adicional para ambas as partes criptografar e descriptografar dados simetricamente. O cliente usa a chave pública do servidor para criptografar a chave simétrica e enviá-la com segurança ao servidor, e o servidor usa sua chave privada para descriptografá-la. Qualquer pessoa pode criptografar usando a chave pública, mas somente o servidor pode descriptografar usando a chave privada.

O oposto é verdadeiro para uma assinatura digital. Um certificado pode ser "assinado" por outra autoridade, pelo que a autoridade efetivamente declara "verificamos que o controlador deste certificado também controla o domínio listado no certificado". Neste caso, a autoridade utiliza a sua chave privada para criptografar o conteúdo do certificado, e este texto cifrado é anexado ao certificado como sua assinatura digital. Qualquer pessoa pode descriptografar esta assinatura usando a chave pública da autoridade e verificar se ela resulta no valor descriptografado esperado. Mas apenas a autoridade pode criptografar o conteúdo usando a chave privada e, portanto, apenas a autoridade pode realmente criar uma assinatura válida.

Portanto, se aparecer um servidor alegando ter um certificado para Microsoft.com assinado pela Symantec (ou qualquer outra CA), seu navegador não precisará acreditar nisso, se for legítimo, a Symantec terá usado sua chave privada (ultra-secreta) para gerar a assinatura digital do certificado SSL do servidor e, portanto, o uso do seu navegador poderá usar sua chave pública para verificar se esta assinatura é válida.

**4.Autoassinatura**

Observe que todos os certificados de CA raiz são "autoassinados", o que significa que a assinatura digital é gerada usando a própria chave privada do

certificado. Não há nada intrinsecamente especial no certificado de uma CA raiz, você pode gerar seu próprio certificado autoassinado e usá-lo para assinar outros certificados, se desejar.

Mas como seu certificado aleatório não é pré-carregado como uma CA em nenhum navegador em qualquer lugar, nenhum deles confiará em você para assinar seus próprios certificados online ou outros certificados.

Isso coloca um fardo enorme para todos os editores de navegadores e sistemas operacionais confiarem apenas em CAs raiz totalmente limpas, já que essas organizações nas quais seus usuários acabam confiando para examinar sites e manter os certificados seguros.

Lavabit era o provedor de e-mail superseguro de Edward Snowden durante a loucura dos vazamentos da NSA em 2013. Como vimos, nenhuma quantidade de hackeamento padrão poderia permitir que o FBI visse quaisquer dados em seu caminho entre a Lavabit e seus clientes. Sem a chave privada do certificado SSL Lavabit, a agência estava ferrada. No entanto, um prestável juiz dos EUA disse ao fundador da Lavabit, Ladar Levison, que ele tinha de entregar esta chave, dando efectivamente liberdade ao FBI para bisbilhotar o tráfego à vontade. Levison fez uma corajosa tentativa de protelar entregando a chave de 2.560 caracteres em 11 páginas impressas de 4 pontos , mas foi recebido com uma ordem exigindo que ele entregasse a chave em um formato útil ou enfrentaria uma multa de US$ 5.000/dia até ele fez.

Assim que ele cumpriu, GoDaddy, a Lavabit CA, revogou o certificado, considerando-o (corretamente) comprometido. Isso adicionou o certificado Lavabit a uma Lista de Revogação de Certificados (CRL), uma lista de certificados desacreditados nos quais os clientes não devem mais confiar para fornecer uma conexão segura. Certificados comprometidos, autoassinados ou não confiáveis fazem com que os navegadores exibem uma grande mensagem de erro vermelha e desencorajam ou proíbam completamente outras ações do usuário. Infelizmente, os navegadores continuarão a confiar em um certificado quebrado até obterem as atualizações mais recentes da CRL, um processo que é aparentemente imperfeito na prática .